December 2020

Detection of Delaminations in Carbon Fiber Reinforced Polymers Embedded with Terfenol-D Particles Using Machine Learning

Christopher Nelon
Clemson University, cnelon@clemson.edu

Detection of Delaminations in Carbon Fiber Reinforced Polymers Embedded with Terfenol-D Particles using Machine Learning

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Mechanical Engineering

---

by
Christopher Michael Nelon
December 2020

Accepted by:
Dr. Oliver Myers, Committee Chair
Dr. Garrett Pataky
Dr. Suyi Li

Abstract

The characterization of the damage state of a system provides insight into its performance and safety during operation. In composite materials, specifically, fiber-reinforced polymers, delaminations form from the evolution of cracks in a matrix that leads to adhesion failure between adjacent laminae. Nondestructive evaluation (NDE) seeks to characterize the state of a material or system during non-operational times. A previously proposed NDE method employs embedded magnetostrictive particles between laminae of carbon fiber reinforced polymer (CFRP) for damage sensing. The phenomenon of magnetostriction couples the mechanical state of a material with its magnetic state so that a change in the local stress field alters its magnetic susceptibility. The change in magnetic susceptibility is measured using an induced sensing voltage.

This work aims to provide a preliminary exploration of machine learning to predict the presence of a delamination using the embedded magnetostrictive particle NDE method with CFRP laminates. Machine learning algorithms' ability to decipher and develop relationships among input features attracted its use since the visual examination of the experimental induced sensing voltage plots yielded inconsistent delamination predictions. This work investigated the feasibility of an analytical model based on the Euler-Bernoulli beam theory to generate data. This model utilized functional relationships to characterize the nonlinear behavior of the magnetostrictive material Terfenol-D. Fourier series relationships reduced the error in the characterization of the properties over the previously proposed functions. The previously proposed derived model failed to converge for the calculation of stress within the magnetostrictive material.

Eight machine learning algorithms were employed using a Python script to classify the presence of a delamination within a unidirectional HexPly AS4/3501-6 CFRP embedded with Terfenol-D particles. The maximum accuracy achieved was approximately 80%, whereas the average accuracy for all the models was just below 71%. The multi-layer perceptron (MLP) models, a neural network

algorithm, produced the highest prediction accuracy for this two-class classification problem because of their ability to account for nonlinear relationships. A parametric study involving the architecture and activation function was necessary for the MLP models because of the range of obtained prediction accuracies. A direct relationship was observed between the number of hidden layers and the accuracy outliers. Despite the accuracy of the examined machine learning models' being less than that of other NDE applications, this preliminary investigation demonstrated that machine learning could be paired with previously indiscernible experimental data to detect delaminations.

Dedication

For my mother and father.

Soli Deo Gloria.

Acknowledgments

I want to thank my family for their support during my studies at Clemson University. I especially would like to thank my parents, Michael and Ashley Nelon, for their gracious support of me financially and emotionally. Their guidance and encouragement have molded me into the person I am today, and I am forever grateful for them. Special thanks also go to my sister, Julia, and grandmother, Anne Scott, who dealt with my frustrations and were always available to listen.

I want to thank my advisor, Dr. Oliver Myers, for supporting me academically, providing me opportunities to share my work, and aiding me to get to this point through the evolution of several projects. Thank you, Dr. Garrett Pataky and Dr. Suyi Li, for assisting in my work and challenging me in my times at Clemson University as a student. Thank you to the Army Research Lab and Dr. Asha Hall for providing the funding for this work.

Finally, I would like to thank those who have crossed my path at Clemson University. My professors challenged me during my studies and were helpful when seeking advice. To the friends I have made, thank you. I would especially like to thank Christopher Knippenberg, Rebecca MacPherson, and Alex McGee for supporting me in various ways as I navigated through my undergraduate and graduate years.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The increased level of research and development of materials has resulted in incorporating specifically designed materials into a component or system's design. Material designers continually seek new materials that exhibit combinations of desirable properties; for instance, aerospace companies endeavor to reduce weight without significantly compromising the strength of aircraft components. Homogeneous materials, such as metals, have undergone extensive research and development to examine their microstructure and how it influences material properties. For centuries, these homogeneous materials were selected for designs based on prior experiences. By blending different metals, researchers produced alloys with characteristics of the combined materials. After further development, alternative composite materials emerged into the commercial design space, especially in the aerospace industry. The popularity of composites has increased because they combine constituent materials to yield new materials with material properties that differ from those of the constituents' properties. By altering the constituent materials, volume fractions, and microstructural composition of a composite, its overall properties can be tailored to design a material for a particular purpose. Carbon fiber composites, specifically, have been exploited in designs necessitating a high specific strength, i.e., the strength-to-weight ratio, as compared to metals [1]. Carbon fiber reinforced polymers (CFRPs) have specific strength values approximately ten times greater than aluminum alloys [2]. Research and development of CFRPs have increased with the rise of the aviation industry.

The utilization of composite materials includes, but is not limited to, designs in the aerospace, automotive, and athletic industries. Aerospace designers employ composites in elements such as

ailerons, elevators, and rudders in airplanes in addition to rotor blades of helicopters and other vertical-takeoff vehicles. An aircraft experiences fluctuating loads during a single operational use due to pressure differences and changing atmospheric conditions at the various altitudes the vehicle operates. Operational usage and environmental conditions affect the structure of a design. Evaluation of a vehicle's structure grants insight into the response and performance of an element or system of a structure. Structural health monitoring (SHM) and nondestructive evaluation (NDE) are two techniques commonly employed to examine a structure's state after enduring environmental and operational loads. These evaluation techniques can characterize a system based on the presence, or perceived presence, of damage in a material. The damage state of a system or material is vital for its operation to avoid potential failure that could lead to financial burdens, not to mention possible causalities. Evaluating a material or system can be time-intensive and result in extensive downtime, so developing innovative evaluation techniques could reduce turnaround time. Additionally, these new methods could result in more accurate evaluation techniques that identify pre-existing internal damage and damage precursors. The prediction of possible damage sites via damage precursors would allow operators to determine if a component should be replaced or offer a potential lifespan estimation for when to replace the component [3].

A benefit of composite materials, as previously mentioned, is the ability to manipulate the structure of the material. CFRPs consist of carbon fibers embedded in a matrix material. Layers, also referred to as laminae or plies, of these embedded fibers are stacked on top of one another to form a composite laminate. Particles can be embedded between adjacent plies or suspended in a ply of host material to evaluate the damage state of a material. Magnetotagging employs embedded magnetostrictive particles in a composite material to detect, or "sense", damage [4]. The theory of magnetostriction couples the mechanical state of a material with its magnetic state [5]. Damage within a material alters the local stress field in the material around the damage state. Therefore, damage near a region of magnetostrictive material induces a change in the magnetostrictive material's magnetic susceptibility.

Previous investigations have demonstrated the feasibility of this proposed NDE technique via analytical, numerical, and experimental methods [3, 6, 7]. An actuating coil first applied an actuation magnetic field to the CFRP beam with embedded magnetostrictive particles. A sensing coil then converted the picked up magnetic flux density with an induced voltage to quantify the change in magnetic susceptibility of the magnetostrictive material, Terfenol-D, per the theory of

magnetostriction. This embedded technique allowed for an in-place examination of a component, hence, decreasing a system's downtime. Evaluating with this approach required a user to have access to either one or both sides of a component; different apparatuses have been employed for actuating and sensing based on the situation and operation of the component [8]. Damage in the form of embedded delaminations was intentionally introduced in experimental laminates at established locations to observe the effect of the presence of a delamination on the induced sensing voltage. The examination and interpretation of the obtained induced sensing voltage plots aimed to provide evidence of the location and presence of a delamination along the length of a CFRP beam. In many cases, no patterns were discernable that highlighted the presence of a delamination. The presence of known damage failed to yield any noticeable effect on the induced sensing voltage, which further complicated this technique's accuracy.

The main objective of this thesis lies in improving the application of the embedded magnetostrictive particle NDE method, specifically, the prediction of detecting the presence of delaminations within a composite beam. Terfenol-D particles will be the magnetostrictive particle embedded in CFRP beams for sensing. The implementation of numerical expressions to represent the material property curves will be investigated in Chapter 3 to provide improved modeling capabilities of Terfenol-D for given conditions. Though literature exists on the characterization of Terfenol-D, the material properties and performance of the material are nonlinear and coupled with various parameters such as the applied magnetic field intensity and prestress state. Characterizing the properties of an embedded Terfenol-D lamina requires consideration of the magnetostrictive particles and the host material enveloping those particles. Accounting for the host material and volume fraction will better recreate experimental conditions rather than assuming the bulk properties of Terfenol-D to describe the embedded sensing lamina. Effective material property calculations will be presented in Chapter 4 for a particulate composite. Machine learning will be applied in Chapter 5 to raw data from the embedded magnetostrictive particle NDE technique in hopes of improving its prediction ability. By supplying machine learning algorithms with varying conditions and datasets, the algorithms can decipher trends and relationships to improve the detection ability of this NDE technique. Ultimately, a trained algorithm could be incorporated into the technique to determine whether or not damage exists at a given location.

Chapter 2

Background and Literature Review

2.1 Introduction to Composite Materials

A composite material combines two or more constituent materials to form a new material with unique properties [1,9–13]. The composite properties combine those of its constituent materials; therefore, a composite may be tailored by a designer to perform in a specific manner. Properties such as strength, stiffness, weight, and fatigue life can be improved through the combination of materials. For example, suspending a stiff material (e.g., metallic particles) in a more compliant material (e.g., epoxy resin) can result in a stiff, light composite. The contribution of the constituents' material properties is related to the volume fraction of the respective material.

There are four common types of composite materials: fibrous composites, particulate composites, laminated composites, and combinations of these three other types. Fibrous, or fiber-reinforced, composites consist of fibers embedded in a host material illustrated in Fig. 2.1. The possible geometries of the fibers include long, continuous fibers and short, discontinuous fibers, also known as whiskers. The diameter of the embedded fibers can be on the scale of micrometers, as with carbon fibers. The strength of a fiber-reinforced composite is derived from its fiber selection. Because of the geometry of a slender fiber, the material properties such as the elastic modulus and strength are greater in magnitude in the longitudinal direction of the fiber, i.e., in the direction parallel to the fiber, rather than the radial direction. Particulate composites consist of particles suspended in a matrix or other host material. The particles and matrix material can be metallic or nonmetallic. The geometry of the particles can include spheres, discs, and flakes. A challenge with

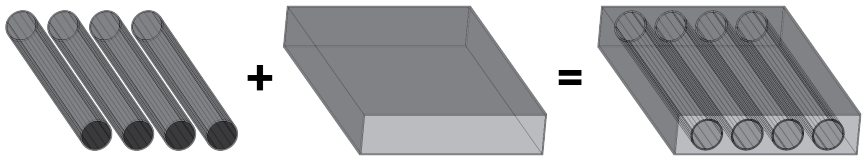


Figure 2.1: Fiber-reinforced composites combine fibers with a matrix material to yield a new material.

particulate composites lies in their manufacturing. If seeking a uniform composite, the distribution of the particles in the matrix material must be controlled to ensure that the concentration of the particles is uniform throughout the composite. The stacking of plies of different materials produces a laminated composite. Homogeneous materials can be combined in layers to result in a laminate. Examples of laminated composites include plywood, safety glass, and kitchen countertops.

Carbon fiber reinforced polymers (CFRPs) are one type of fiber-reinforced composite. Stiff, strong carbon fibers act as reinforcing elements in a more compliant thermoset or thermoplastic polymer matrix. A single layer of a CFRP is referred to as a lamina or ply; stacking multiple laminae forms a laminate. CFRPs have grown in popularity in design because of their high specific strength and specific stiffness compared to metals and technical ceramics [2]. By changing the orientation of its fiber-reinforced laminae, the overall properties of a laminate can be tailored to specific values.

Composite materials are initially considered anisotropic and inhomogeneous in nature because they combine two or more constituent materials in random orientations. An inhomogeneous composite has properties that depend on the location within the composite. For instance, the material properties in a CFRP on a microscopic level are different depending on if the point of interest lies in the fiber or polymer matrix domain. No planes of symmetry exist for an anisotropic material; therefore, the material properties differ in all directions at a given point. The orientation of the point determines the values of the material properties. The assumption of symmetry allows for the simplification of the composite. When a composite has three mutually perpendicular planes of symmetry, it is considered orthotropic. The material properties of an orthotropic material differ only along the three mutually perpendicular directions. If at a given point there exists only one plane of symmetry about which the material properties are equal in all directions, the material is considered transversely isotropic. The properties of a transversely isotropic material differ in only two

directions. Fiber-reinforced composites can be modeled as transversely isotropic materials because of the geometry of the fibers. The properties along the radial direction of the fibers are assumed equal and different from those along the longitudinal axis of the fibers due to the slenderness of the fiber. The differences in the through-direction properties and those in the transverse direction to the laminate's plane are considered negligible. An isotropic material exists when material properties are equivalent in all directions, often true of homogeneous materials; hence, composite materials are rarely considered isotropic. Some exceptions may be made based on the structure of the composite. A particulate composite containing spherical particles could be considered isotropic based on assumptions made concerning its constituents. A matrix material is often considered isotropic, and the spherical geometry of the embedded particles lends to an isotropic assumption.

Damage mechanisms of composite materials differ from those of the traditionally studied crystalline, homogeneous solids because of their micromechanical structure. In a crystalline solid, damage forms and grows in the lattice structure of the material. Each constituent material must be considered when discussing damage in a composite. Matrix cracking, fiber breakage, and delaminations are common damage mechanisms for composites [1]. Damage growth often begins at the interface between a filler element, i.e., a fiber or particle, and the matrix material as the magnitude of the stress field increases around the embedded interstitial body. Damage can occur in the crystal structure of the constituent materials; however, this type of damage is often not the aim of study because of the numerous other failure methods. Early damage involves cracking in the composite matrix region as the matrix is often the more compliant constituent material. The matrix cracks continue to grow under loading, specifically tensile loading, and may eventually accumulate to form either a more extensive crack or initiate a delamination site. A delamination exists when internal constituent materials separate from each other, creating a region of discontinuity. Further debonding of laminae or the development of void spaces is possible with the growth of delaminations. In the case of a fiber-reinforced polymer (FRP) composite, fiber breakage occurs when the applied loading achieves or exceeds the fiber's ultimate tensile, compression, or shear strength(s). Fiber breakage typically corresponds to the failure of an FRP because the fiber is the stiffest constituent of the composite.

Table 2.1: Elastic constants for a transversely isotropic material.

Elastic constant	Description
E_1	Elastic modulus along longitudinal direction (1) of fiber
E_2	Elastic modulus along transverse direction (2) of fiber
G_{12}	Shear modulus in the plane (12) of the lamina
ν_{12}	Poisson's ratio in the plane (12) of the lamina

2.2 Mechanics of Composite Materials

2.2.1 Assumptions for Elastic Properties

Assumptions are made when modeling the behavior of a composite material to simplify the inherent complexity associated with the combination of multiple materials with a random distribution. A fiber-reinforced laminate, specifically a CFRP, is considered throughout the subsequent discussion. The embedded fibers of the composite are assumed to be linearly elastic and homogeneous. Per their geometry, they are considered transversely isotropic with the material properties equal in the radial direction of the fibers. The matrix is considered linearly elastic, homogeneous, and isotropic having equivalent material properties in all directions throughout its domain. The fibers are assumed to be embedded at evenly spaced intervals and parallel in the longitudinal direction. The assumption of perfect bonding between the fibers and matrix eliminates the modeling of voids and discontinuities within the material. The overall fiber-reinforced lamina may then be considered a linear elastic, transversely isotropic material. Four elastic constants define the lamina with respect to its orientation as listed in Tab. 2.1.

2.2.2 Elastic Mechanics of a Lamina

The general form for the elastic stress-strain relationship for a material is given by the generalized Hooke's Law (in index notation):

$$\sigma_i = C_{ij}\epsilon_j, \tag{2.1}$$

where C_{ij} is the stiffness matrix. The stress-strain relationship for a linear elastic, anisotropic material can be written in matrix form in terms of the individual elastic constants,

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \tau_{23} \\ \tau_{13} \\ \tau_{12} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} \\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} \end{bmatrix} \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ \gamma_{23} = 2\epsilon_{23} \\ \gamma_{13} = 2\epsilon_{31} \\ \gamma_{12} = 2\epsilon_{12} \end{bmatrix} \tag{2.2}$$

which contains 36 independent elastic constants. In contraction notation, the indices on the stress and strain terms may be simplified such that,

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \tau_{23} \\ \tau_{13} \\ \tau_{12} \end{bmatrix} = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{bmatrix} \qquad \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ \gamma_{23} \\ \gamma_{13} \\ \gamma_{12} \end{bmatrix} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \end{bmatrix} \tag{2.3}$$

Symmetry can reduce the number of independent elastic constants in Eq. (2.2) from 36 to 21:

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{12} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{13} & C_{23} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{14} & C_{24} & C_{34} & C_{44} & C_{45} & C_{46} \\ C_{15} & C_{25} & C_{35} & C_{45} & C_{55} & C_{56} \\ C_{16} & C_{26} & C_{36} & C_{46} & C_{56} & C_{66} \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \end{bmatrix} \tag{2.4}$$

For an orthotropic material with three mutually perpendicular planes of symmetry which correspond to the main coordinate system of the laminate, simplifications to Eq. (2.4) can be made to reduce

the number of independent elastic constants to nine:

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{21} & C_{22} & C_{23} & 0 & 0 & 0 \\ C_{31} & C_{32} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{66} \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \end{bmatrix} \tag{2.5}$$

For a material with a plane of symmetry in the 2-3 plane such as a fiber-reinforced composite with the 1-direction corresponding to the longitudinal direction of the fiber, the stress-strain relationship for a transversely isotropic material contains only five independent elastic constants:

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{21} & C_{22} & C_{23} & 0 & 0 & 0 \\ C_{31} & C_{32} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & (C_{11} - C_{22})/2 \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \end{bmatrix} \tag{2.6}$$

An isotropic material contains an infinite number of planes of symmetry regarding its material properties. The stress-strain relationship simplifies the number of independent elastic constants as given in Eq. (2.5) from nine to two:

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{11} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{12} & C_{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & (C_{11} - C_{12})/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & (C_{11} - C_{12})/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & (C_{11} - C_{12})/2 \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \end{bmatrix} \tag{2.7}$$

A problem involving a composite laminate can often be considered a plane stress problem under the assumption the composite is thin and no out-of-plane loads are applied to the laminate.

For many composites, especially CFRPs, the thickness is often negligible when compared to the length or width of the sample of interest. The stress state for a plane stress problem is given as:

$$\sigma_3 = 0, \qquad \tau_{23} = 0, \qquad \tau_{31} = 0 \tag{2.8}$$

$$\sigma_1 \neq 0, \qquad \sigma_2 \neq 0, \qquad \tau_{12} \neq 0 \tag{2.9}$$

The three-dimensional relationship in Eq. (2.1) is now reduced to a two-dimensional relationship. The stress-strain relationship for an orthotropic material is given in terms of the reduced stiffness terms $[Q_{ij}]$,

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_{12} \end{bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} & 0 \\ Q_{12} & Q_{22} & 0 \\ 0 & 0 & Q_{66} \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \gamma_{12} \end{bmatrix}, \tag{2.10}$$

or in terms of the compliance matrix, defined by

$$[S] = [C]^{-1}, \tag{2.11}$$

$$\epsilon_i = S_{ij}\sigma_j, \tag{2.12}$$

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \gamma_{12} \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} & 0 \\ S_{12} & S_{22} & 0 \\ 0 & 0 & S_{66} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_{12} \end{bmatrix} \tag{2.13}$$

The reduced stiffness constants in Eq. (2.10) are related to the compliance matrix $[S]$ coefficients or expressible in terms of the elastic properties of the lamina:

$$Q_{11} = \frac{S_{22}}{S_{11}S_{22} - S_{12}^2} = \frac{E_1}{1 - \nu_{12}\nu_{21}} = \frac{E_1^2}{E_1 - E_2\nu_{12}^2} \tag{2.14}$$

$$Q_{12} = Q_{21} = \frac{S_{12}}{S_{11}S_{22} - S_{12}^2} = \frac{\nu_{12}E_2}{1 - \nu_{12}\nu_{21}} = \frac{E_1E_2\nu_{12}}{E_1 - E_2\nu_{12}^2} \tag{2.15}$$

$$Q_{22} = \frac{S_{11}}{S_{11}S_{22} - S_{12}^2} = \frac{E_2}{1 - \nu_{12}\nu_{21}} = \frac{E_1E_2}{E_1 - E_2\nu_{12}^2} \tag{2.16}$$

$$Q_{66} = \frac{1}{S_{66}} = G_{12} \tag{2.17}$$

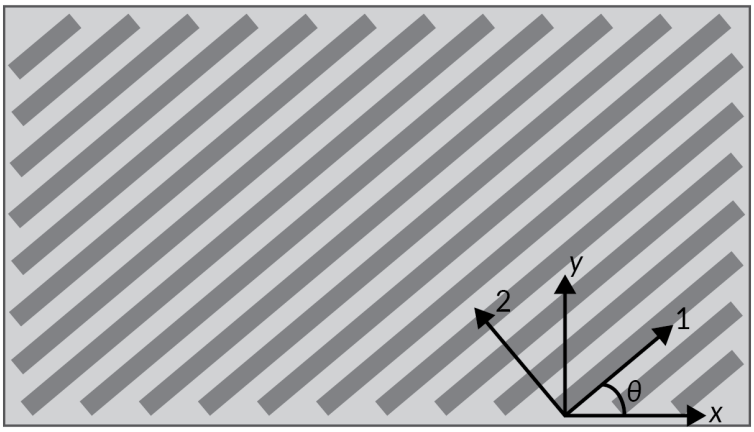


Figure 2.2: The local coordinate system differs from the global coordinate system in a fiber-reinforced polymer when the fibers are rotated an angle θ from the global coordinates.

The compliance terms in Eq. (2.13) are calculated using the elastic constants for the lamina:

$$S_{11} = \frac{1}{E_1} \tag{2.18}$$

$$S_{12} = -\frac{\nu_{12}}{E_1} = -\frac{-\nu_{21}}{E_2} \tag{2.19}$$

$$S_{22} = \frac{1}{E_2} \tag{2.20}$$

$$S_{66} = \frac{1}{12} \tag{2.21}$$

As previously mentioned, one benefit of employing a fiber-reinforced laminate is the ability to manipulate its behavior by altering the angle of the plies with respect to a global axis. For a fiber-reinforced laminate, two coordinates system exist: the local coordinate system of the fiber laminae $(1, 2, 3)$ and the global, or principal, directions associated with the laminate (x, y, z). An orientation angle θ is defined as the angle between the global and local axes as illustrated in Fig. 2.2. A mechanics approach for transforming stress from one coordinate system to another permits converting stress in the local coordinate system to the global system. In matrix form, this transformation matrix $[T]$ is expressed as

$$[T] = \begin{bmatrix} \cos^2\theta & \sin^2\theta & 2\cos\theta\sin\theta \\ \sin^2\theta & \cos^2\theta & -2\cos\theta\sin\theta \\ -\cos\theta\sin\theta & \cos\theta\sin\theta & \cos^2\theta - \sin^2\theta \end{bmatrix} \tag{2.22}$$

The transformation matrix can be applied to transform both stress,

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_{12} \end{bmatrix} = [T] \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix}, \qquad \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} = [T]^{-1} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_{12} \end{bmatrix}, \tag{2.23}$$

and strain,

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_{12} \end{bmatrix} = [T] \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_{xy} \end{bmatrix}, \qquad \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_{xy} \end{bmatrix} = [T]^{-1} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_{12} \end{bmatrix} \tag{2.24}$$

The shear strain $(\gamma_{12}, \gamma_{xy})$ is equivalent to twice the engineering strain $(\epsilon_{12}, \epsilon_{xy})$, or

$$\gamma = 2\epsilon \tag{2.25}$$

The $[R]$ matrix is defined to account for the relationship in Eq. (2.25),

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} \tag{2.26}$$

Transforming the strain from the local to the global coordinate system using the $[R]$ matrix is accomplished by

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \gamma_{12} \end{bmatrix} = [R]\,[T]\,[R]^{-1} \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix} \tag{2.27}$$

The relationship for attaining the local stress from the local strain is expressed as

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} = [T]^{-1}\,[Q]\,[R]\,[T]\,[R]^{-1} \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix} \tag{2.28}$$

The transformed reduced stiffness matrix is defined as

$$\left[\overline{Q}\right] = [T]^{-1}[Q][R][T], \tag{2.29}$$

and relates the reduced stiffness constants in the local coordinate system to the global system; therefore, the $\left[\overline{Q}\right]$ is a global matrix in the principal directions of the laminate. The transformed reduced stiffness constants in $\left[\overline{Q}\right]$ are defined as follows:

$$\overline{Q}_{11} = Q_{11}\cos^4\theta + Q_{22}\sin^4\theta + 2\left(Q_{12} + 2Q_{66}\right)\sin^2\theta\cos^2\theta \tag{2.30}$$

$$\overline{Q}_{12} = \left(Q_{11} + Q_{22} - 4Q_{66}\right)\sin^2\theta\cos^2\theta + Q_{12}\left(\cos^4\theta + \sin^4\theta\right) \tag{2.31}$$

$$\overline{Q}_{22} = Q_{11}\sin^4\theta + Q_{22}\cos^4\theta + 2\left(Q_{12} + 2Q_{66}\right)\sin^2\theta\cos^2\theta \tag{2.32}$$

$$\overline{Q}_{16} = \left(Q_{11} - Q_{12} - 2Q_{66}\right)\cos^3\theta\sin\theta - \left(Q_{22} - Q_{12} - 2Q_{66}\right)\cos\theta\sin^3\theta \tag{2.33}$$

$$\overline{Q}_{26} = \left(Q_{11} - Q_{12} - 2Q_{66}\right)\cos\theta\sin^3\theta - \left(Q_{22} - Q_{12} - 2Q_{66}\right)\cos^3\theta\sin\theta \tag{2.34}$$

$$\overline{Q}_{66} = \left(Q_{11} + Q_{22} - 2Q_{12} - 2Q_{66}\right)\cos^2\theta\sin^2\theta + Q_{66}\left(\sin^4\theta + \cos^4\theta\right) \tag{2.35}$$

The global stress-strain relationship using $\left[\overline{Q}\right]$ is expressed as

$$\begin{bmatrix}\sigma_x\\ \sigma_y\\ \tau_{xy}\end{bmatrix} = \begin{bmatrix}\overline{Q}_{11} & \overline{Q}_{12} & \overline{Q}_{16}\\ \overline{Q}_{12} & \overline{Q}_{22} & \overline{Q}_{26}\\ \overline{Q}_{16} & \overline{Q}_{26} & \overline{Q}_{66}\end{bmatrix}\begin{bmatrix}\epsilon_x\\ \epsilon_y\\ \gamma_{xy}\end{bmatrix}, \tag{2.36}$$

or more generally,

$$[\sigma] = \left[\overline{Q}\right][\epsilon] \tag{2.37}$$

2.3 Classical Lamination Theory

The classical lamination theory (CLT) analytically predicts the strain and curvature of a composite material. The theory's approach follows similarly to that of the Euler-Bernoulli beam theory and closely resembles the classical plate theory [14]. The laminate strains and stresses are estimated for applied loads using the laminae's material properties. The elastic constants E, G, and ν are primarily used in the analysis along with information about the individual laminae, including

the orientation angle θ of the laminae and their thicknesses t. If a thermal difference occurs, i.e., a temperature change, thermal loads induce additional stress and strain on the laminate. The coefficients of thermal expansion α for the laminae determine how temperature affects the magnitude and direction of the induced strain.

2.3.1 Assumptions

Because composites are defined as an aggregate of two or more distinct materials, simplifying their behavior is vital for modeling with the elastic relationships presented in Section 2.2.2 [15]. The laminae are simplified from anisotropic to orthotropic materials. Assumptions also permit for the CLT to be implemented with a laminate.

1. The thickness of each ply within the laminate is minimal compared to the other dimensions ($t \ll w$ and $t \ll l$).
2. The layers within the laminate are completely and perfectly bonded to one another.
3. Lines perpendicular to the surface remain straight and perpendicular to the surface following deformation.
4. The laminate and individual plies are considered to be linear elastic.
5. The stresses and strains through the thickness of the material are negligible.

Assumption 2 implies that no damage can exist within the laminate; damage can include delaminations, fiber or matrix cracking, and voids [1]. No slippage can occur at the interfaces of laminae since they are perfectly bonded to one another. Additionally, this assumption implies that all the laminae must deform together. One lamina can not deform in a manner unlike the other n laminae that form a laminate. An important assumption which simplifies the behavior of modeling the laminate is assumption 5 which constitutes a plane stress problem; the stress and strain in the direction perpendicular to the plane of the laminate are zero as illustrated by Eq. (2.9). Symbolically, assumption 5 states that $\sigma_3 = 0$ and $\epsilon_3 = 0$. This assumption can be made by assuming the thickness of the material to be negligible compared to other dimensions, as stated by assumption 1. Assumptions of the CLT specifically relate to the overall laminate rather than the individual laminae, so homogeneous behavior is necessary when using the CLT.

2.3.2 Application of Assumptions

As previously mentioned, the classical lamination theory assumes the individual laminae to display orthotropic material properties. The theory also assumes a plane stress state to eliminate any through-thickness stress effects. The generalized stress-strain relationship in the principal coordinate system for an arbitrarily oriented laminate was given in Eq. (2.37). For a laminate with k laminae, the stress-strain relationship is

$$\{\sigma\}_k = \left[\overline{Q}\right]_k \{\epsilon\}_k \tag{2.38}$$

where k denotes the position of the lamina of interest within the laminate. The Kirchhoff hypothesis asserts that a line perpendicular to the original, undeformed surface remains straight and normal to the deformed surface as shown in Fig. 2.3 (assumption 3). Mathematically, this assumption states that the shear strains in the direction perpendicular to the xy-plane of the laminate are ignored,

$$\gamma_{xz} = \gamma_{yz} = 0 \tag{2.39}$$

Furthermore, the strain components in the z-direction are assumed to be negligible since the laminate thickness does not change, nor do the laminae become unbonded.

2.3.3 Plate Mechanics

For completeness, the strain experienced within a laminate is defined. Let the displacement in the x-, y-, and z-directions be denoted by u, v, and w, respectively. The strain is defined as

$$\epsilon_x = \frac{\partial u}{\partial x} \tag{2.40}$$

$$\epsilon_y = \frac{\partial v}{\partial y} \tag{2.41}$$

$$2\epsilon_{xy} = \gamma_{xy} = \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{2.42}$$

When flexed, the slope of a plate is equal to the change in displacement in the z-direction divided by the change in displacement in the direction of interest.

$$\frac{\partial w}{\partial x}: \quad \text{slope in the } x\text{-direction} \tag{2.43}$$

$$\frac{\partial w}{\partial y}: \quad \text{slope in the } y\text{-direction} \tag{2.44}$$

The CLT assumes a laminate experiences perfect bonding between its constituent materials and all the laminae that combine to form the laminate (assumption 2); therefore, a laminate may be considered and modeled as a single body. Ideally, the material is modeled as a homogeneous material. The midplane, or middle surface, can then be used as a reference within the laminate since a line straight and perpendicular to the midplane remains straight and perpendicular to the midplane following deformation [16]. The displacements associated with the midplane are denoted as u_0 and v_0 for the x- and y-directions, respectively. Per the Kirchhoff hypothesis, a set of linear functions describe the displacement at any point in a cross-section of the laminate:

$$u = u_0 - z\frac{\partial w}{\partial x} \tag{2.45}$$

$$v = v_0 - z\frac{\partial w}{\partial y} \tag{2.46}$$

The strains in the x- and y-directions can be related to Eqs. (2.45) and (2.46) through differentiation with respect to the direction of the strain term.

$$\epsilon_x = \frac{\partial u}{\partial x} = \frac{\partial u_0}{\partial x} - z\frac{\partial^2 w}{\partial x^2} \tag{2.47}$$

$$\epsilon_y = \frac{\partial v}{\partial y} = \frac{\partial v_0}{\partial y} - z\frac{\partial^2 w}{\partial y^2} \tag{2.48}$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = \frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x} - 2z\frac{\partial^2 w}{\partial x \partial y} \tag{2.49}$$

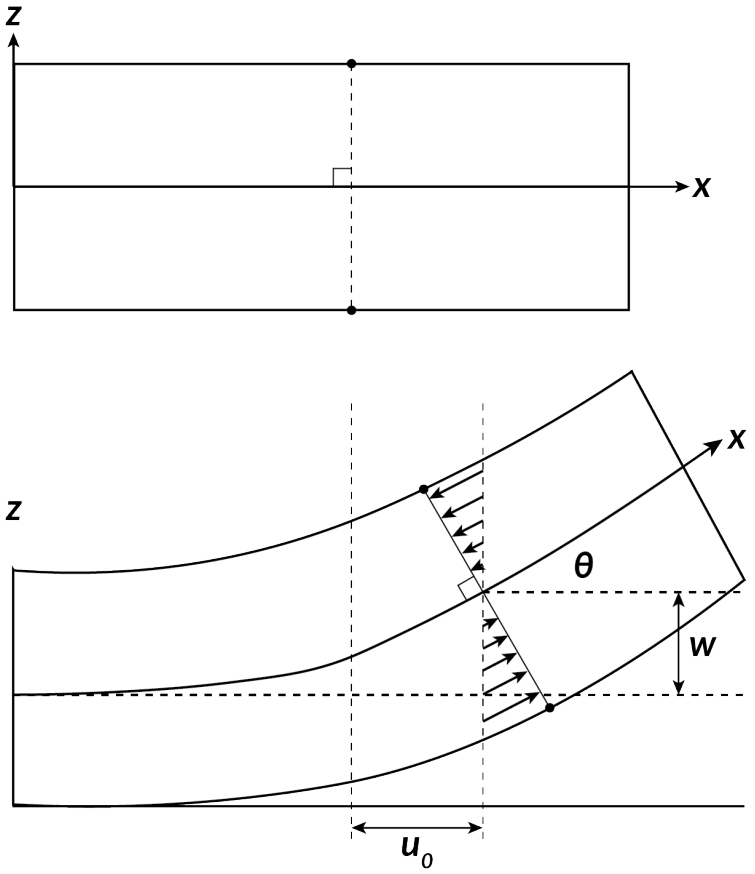


Figure 2.3: In the classical lamination theory, the line perpendicular to the midplane when undeformed remains straight and perpendicular to the midplane when deformed.

The midplane strains are then defined as

$$\epsilon_x^0 := \frac{\partial u_0}{\partial x_0} \tag{2.50}$$

$$\epsilon_y^0 := \frac{\partial v_0}{\partial y_0} \tag{2.51}$$

$$\gamma_{xy}^0 := \frac{\partial u_0}{\partial y_0} + \frac{\partial v_0}{\partial x_0} \tag{2.52}$$

and the midplane curvatures are defined as

$$\kappa_x^0 := -\frac{\partial^2 w}{\partial x^2} \tag{2.53}$$

$$\kappa_y^0 := -\frac{\partial^2 w}{\partial y^2} \tag{2.54}$$

$$\kappa_{xy}^0 := -2\frac{\partial^2 w}{\partial x \partial y} \tag{2.55}$$

The defined terms in Eqs. (2.50) – (2.55) can be substituted into Eqs. (2.47) – (2.49) to yield expressions to find the strain field at any point within the cross-section of a laminate.

$$\epsilon_x = \epsilon_x^0 + z\kappa_x^0 \tag{2.56}$$

$$\epsilon_y = \epsilon_y^0 + z\kappa_y^0 \tag{2.57}$$

$$\gamma_{xy} = \gamma_{xy}^0 + z\kappa_{xy}^0 \tag{2.58}$$

In matrix form, Eqs. (2.56)–(2.58) are expressed as

$$\begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} \epsilon_x^0 \\ \epsilon_y^0 \\ \gamma_{xy}^0 \end{bmatrix} + z \begin{bmatrix} \kappa_x^0 \\ \kappa_y^0 \\ \kappa_{xy}^0 \end{bmatrix} \tag{2.59}$$

Having defined the strain in an arbitrary, global coordinate system for a lamina, the stress in the same coordinate system can be calculated. Equation (2.59) can be substituted into Eq. (2.38) to calculate the stress in the global system using the midplane stress and strain.

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} = \begin{bmatrix} \overline{Q}_{11} & \overline{Q}_{12} & \overline{Q}_{16} \\ \overline{Q}_{12} & \overline{Q}_{22} & \overline{Q}_{26} \\ \overline{Q}_{16} & \overline{Q}_{26} & \overline{Q}_{66} \end{bmatrix}_k \begin{bmatrix} \epsilon_x^0 \\ \epsilon_y^0 \\ \gamma_{xy}^0 \end{bmatrix} + z \begin{bmatrix} \overline{Q}_{11} & \overline{Q}_{12} & \overline{Q}_{16} \\ \overline{Q}_{12} & \overline{Q}_{22} & \overline{Q}_{26} \\ \overline{Q}_{16} & \overline{Q}_{26} & \overline{Q}_{66} \end{bmatrix}_k \begin{bmatrix} \kappa_x^0 \\ \kappa_y^0 \\ \kappa_{xy}^0 \end{bmatrix} \tag{2.60}$$

2.3.4 Stress and Moment Resultants

While the strain experienced by the laminate is linear and continuous through its thickness, the stress is a function of the elastic properties of each lamina. The stress can be considered a piecewise function as the reduced stiffness matrices $\left[\overline{Q}\right]$ differ between laminae due to differing

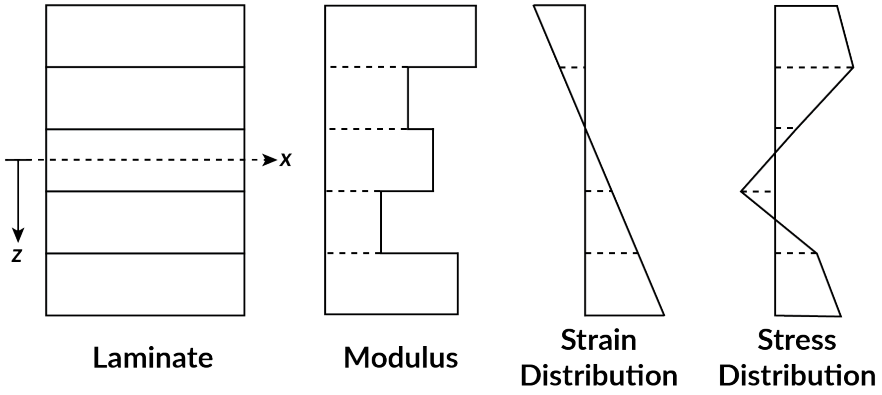


Figure 2.4: Even though the moduli of the individual laminae may differ, the strain is continuous and linear across the thickness of a laminate when using the classical lamination theory. The stress, however, can be discontinuous between laminae with changes in slope at the interface coordinates.

elastic properties or orientation angles with fiber-reinforced laminae. Therefore, the stress follows a continuous, linear relationship in the domain of each lamina, but not across the entire thickness of the laminate as illustrated in Fig. 2.4. Discontinuities may exist due to the piecewise behavior at the interfaces of each lamina.

The CLT relates the forces and moments applied to a laminate to the stress and strain experienced, specifically those at the midplane. For a cross-section of a laminate, the total force in one direction for an infinitesimal element can be expressed as (the x-direction is used for this derivation)

$$\sum F_x = \sum \sigma_x \, (dz) \, (y) \tag{2.61}$$

Assuming that the size of the element decreases such that $dz \to 0$, Eq. (2.61) is expressible in integral form as

$$N_x = \int_{-h/2}^{h/2} \sigma_x \, dz \tag{2.62}$$

where N_x is the stress resultant in the x-direction and has units of force per length, and h represents the thickness of the laminate. The $h/2$ term denotes the locations respective to the outer surfaces of the laminate as the integration is centered about the midplane of the laminate. The stress resultants

are expressible as

$$N_x = \int_{-h/2}^{h/2} \sigma_x \, dz \tag{2.63}$$

$$N_y = \int_{-h/2}^{h/2} \sigma_y \, dz \tag{2.64}$$

$$N_{xy} = \int_{-h/2}^{h/2} \tau_{xy} \, dz \tag{2.65}$$

Moment resultants are calculated based on that z is the perpendicular distance from the midplane to an arbitrary point in the laminate cross-section along the thickness direction. The moment resultants are:

$$M_x = \int_{-h/2}^{h/2} \sigma_x z dz \tag{2.66}$$

$$M_y = \int_{-h/2}^{h/2} \sigma_y z dz \tag{2.67}$$

$$M_{xy} = \int_{-h/2}^{h/2} \tau_{xy} z dz \tag{2.68}$$

The units of the moment resultants are torque per unit length. The resultants M_x and M_y cause the laminate to bend, whereas M_{xy} causes twisting. Equations (2.63) – (2.68) are expressed in matrix form by

$$\begin{bmatrix} N_x \\ N_y \\ N_{xy} \end{bmatrix} = \int_{-h/2}^{h/2} \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} dz, \tag{2.69}$$

$$\begin{bmatrix} M_x \\ M_y \\ M_{xy} \end{bmatrix} = \int_{-h/2}^{h/2} \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} z dz \tag{2.70}$$

2.3.5 Thermal Resultants

The thermal expansion coefficients are defined for a material in its local coordinate system. As with the stress and strain, the locally defined coefficients of thermal expansion may be transformed

into the principal directions of the laminate using Eq. (2.22),

$$\begin{bmatrix} \alpha_x \\ \alpha_y \\ \alpha_{xy} \end{bmatrix} = \begin{bmatrix} \cos^2\theta & \sin^2\theta & -2\cos\theta\sin\theta \\ \sin^2\theta & \cos^2\theta & 2\cos\theta\sin\theta \\ \cos\theta\sin\theta & -\cos\theta\sin\theta & \cos^2\theta - \sin^2\theta \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ 0 \end{bmatrix} = [T]^{-1} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ 0 \end{bmatrix} \tag{2.71}$$

A coefficient of thermal expansion is created in the shear direction for the case of $\theta \neq 0^\circ$ or $\theta \neq 90^\circ$. The thermal strains in each lamina are then expressible as [17]

$$\epsilon_x^T = \alpha_x \Delta T \tag{2.72}$$

$$\epsilon_y^T = \alpha_y \Delta T \tag{2.73}$$

$$\epsilon_{xy}^T = \frac{\gamma_{xy}^T}{2} = \alpha_{xy} \Delta T \tag{2.74}$$

where $\Delta T = T_1 - T_2$ represents the change in temperature experienced by the material from state one T_1 to state two T_2. The T superscript denotes that the strains are thermal strains. The thermal resultants are found using the same methodology employed for calculating the mechanical resultants which combines the elastic properties, orientation angle, and location of the individual lamina to calculate the thermal resultants:

$$\begin{bmatrix} N_x^T \\ N_y^T \\ N_{xy}^T \end{bmatrix} = \sum_{k=1}^{n} \begin{bmatrix} \overline{Q}_{11} & \overline{Q}_{12} & \overline{Q}_{16} \\ \overline{Q}_{12} & \overline{Q}_{22} & \overline{Q}_{26} \\ \overline{Q}_{16} & \overline{Q}_{26} & \overline{Q}_{66} \end{bmatrix}_k \begin{bmatrix} \epsilon_x^T \\ \epsilon_y^T \\ \gamma_{xy}^T \end{bmatrix}_k (h_k - k_{k-1}) \tag{2.75}$$

$$\begin{bmatrix} M_x^T \\ M_y^T \\ M_{xy}^T \end{bmatrix} = \sum_{k=1}^{n} \begin{bmatrix} \overline{Q}_{11} & \overline{Q}_{12} & \overline{Q}_{16} \\ \overline{Q}_{12} & \overline{Q}_{22} & \overline{Q}_{26} \\ \overline{Q}_{16} & \overline{Q}_{26} & \overline{Q}_{66} \end{bmatrix}_k \begin{bmatrix} \epsilon_x^T \\ \epsilon_y^T \\ \gamma_{xy}^T \end{bmatrix}_k \frac{1}{2}\left(h_k^2 - k_{k-1}^2\right) \tag{2.76}$$

2.3.6 Total Resultants

The total stress and moment resultants are found by adding the mechanical and thermal resultants together. The hygroscopic resultants are calculated in the same manner as the thermal resultants in Section 2.3.5 using the hygroscopic coefficients and a moisture difference Δm rather

than the temperature change; this effect was disregarded in this discussion.

$$\begin{bmatrix} N_x^{total} \\ N_y^{total} \\ N_{xy}^{total} \end{bmatrix} = \begin{bmatrix} N_x \\ N_y \\ N_{xy} \end{bmatrix} + \begin{bmatrix} N_x^T \\ N_y^T \\ N_{xy}^T \end{bmatrix} \tag{2.77}$$

$$\begin{bmatrix} M_x^{total} \\ M_y^{total} \\ M_{xy}^{total} \end{bmatrix} = \begin{bmatrix} M_x \\ M_y \\ M_{xy} \end{bmatrix} + \begin{bmatrix} M_x^T \\ M_y^T \\ M_{xy}^T \end{bmatrix} \tag{2.78}$$

2.3.7 Constitutive Relations for a Laminate

The CLT examines the laminate on a macromechanical scale, though, it considers the effects of the individual laminae by including the elastic properties, orientation angle, and thickness of each lamina. Equations (2.69) and (2.70) are expressible in terms of k, or the ply number, through summation:

$$\begin{bmatrix} N_x \\ N_y \\ N_{xy} \end{bmatrix} = \sum_{k=1}^{n} \int_{h_{k-1}}^{h_k} \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix}_k dz \tag{2.79}$$

$$\begin{bmatrix} M_x \\ M_y \\ M_{xy} \end{bmatrix} = \sum_{k=1}^{n} \int_{h_{k-1}}^{h_k} \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix}_k z dz \tag{2.80}$$

Integrating the expressions in Eqs. (2.79) and (2.80) results in two matrix equations with coefficient matrices defined by the elastic properties, thickness, and orientation angle of the laminae.

$$\begin{bmatrix} N_x \\ N_y \\ N_{xy} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{16} \\ A_{12} & A_{22} & A_{26} \\ A_{16} & A_{26} & A_{66} \end{bmatrix} \begin{bmatrix} \epsilon_x^0 \\ \epsilon_y^0 \\ \gamma_{xy}^0 \end{bmatrix} + \begin{bmatrix} B_{11} & B_{12} & B_{16} \\ B_{12} & B_{22} & B_{26} \\ B_{16} & B_{26} & B_{66} \end{bmatrix} \begin{bmatrix} \kappa_x^0 \\ \kappa_y^0 \\ \kappa_{xy}^0 \end{bmatrix} \tag{2.81}$$

$$\begin{bmatrix} M_x \\ M_y \\ M_{xy} \end{bmatrix} = \begin{bmatrix} B_{11} & B_{12} & B_{16} \\ B_{12} & B_{22} & B_{26} \\ B_{16} & B_{26} & B_{66} \end{bmatrix} \begin{bmatrix} \epsilon_x^0 \\ \epsilon_y^0 \\ \gamma_{xy}^0 \end{bmatrix} + \begin{bmatrix} D_{11} & D_{12} & D_{16} \\ D_{12} & D_{22} & D_{26} \\ D_{16} & D_{26} & D_{66} \end{bmatrix} \begin{bmatrix} \kappa_x^0 \\ \kappa_y^0 \\ \kappa_{xy}^0 \end{bmatrix} \tag{2.82}$$

where

$$A_{ij} = \sum_{k=1}^{n} \left[\overline{Q}_{ij}\right]_k (h_k - k_{k-1}) \tag{2.83}$$

$$B_{ij} = \frac{1}{2} \sum_{k=1}^{n} \left[\overline{Q}_{ij}\right]_k \left(h_k^2 - k_{k-1}^2\right) \tag{2.84}$$

$$D_{ij} = \frac{1}{3} \sum_{k=1}^{n} \left[\overline{Q}_{ij}\right]_k \left(h_k^3 - k_{k-1}^3\right) \tag{2.85}$$

The matrix equations in (2.81) and (2.82) can be combined and written in a single matrix equation:

$$\begin{bmatrix} N_x \\ N_y \\ N_{xy} \\ M_x \\ M_y \\ M_{xy} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{16} & B_{11} & B_{12} & B_{16} \\ A_{12} & A_{22} & A_{26} & B_{12} & B_{22} & B_{26} \\ A_{16} & A_{26} & A_{66} & B_{16} & B_{26} & B_{66} \\ B_{11} & B_{12} & B_{16} & D_{11} & D_{12} & D_{16} \\ B_{12} & B_{22} & B_{26} & D_{12} & D_{22} & D_{26} \\ B_{16} & B_{26} & B_{66} & D_{16} & D_{26} & D_{66} \end{bmatrix} \begin{bmatrix} \epsilon_x^0 \\ \epsilon_y^0 \\ \gamma_{xy}^0 \\ \kappa_x^0 \\ \kappa_y^0 \\ \kappa_{xy}^0 \end{bmatrix} \tag{2.86}$$

In contracted form, Eq. (2.86) expresses the individual elements of each submatrix as its parent matrix and drops the index notation such that

$$\begin{bmatrix} \mathbf{N} \\ \mathbf{M} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \epsilon^{\mathbf{0}} \\ \kappa^{\mathbf{0}} \end{bmatrix} \tag{2.87}$$

which illustrates the matrix denoted as the $[ABD]$ matrix, or **ABD**, in composite terminology. The resultant vector on the left side of Eq. (2.87) is equal to the sum of resultants given by Eqs. (2.77) and (2.78).

2.3.8 Explanation of ABD Matrices

The defined matrix in Eq. (2.83), the **A** matrix, is called the extensional stiffness matrix. From (2.83), the term surrounded by parentheses can be expressed as t_k, the thickness of the

respective lamina.

$$A_{ij} = \sum_{k=1}^{n} \left[\overline{Q}_{ij}\right]_k t_k \tag{2.88}$$

$$t_k := h_k - h_{k-1} \tag{2.89}$$

This matrix relates normal stress and midplane strain. The A_{16} and A_{26} terms relate shear strains to normal stresses and normal strains to shear stresses rather than relating normal stress to normal strain. If these terms are non-zero values, normal stresses will occur when a shear strain is applied.

The **B** matrix is referred to as the coupling stiffness matrix, and it relates midplane curvature and normal stresses. Like the terms in the **A** matrix, the B_{16} and B_{26} constants relate twisting strains to normal stresses and shear strains to bending stresses. An alternative expression to Eq. (2.84) is [15]

$$B_{ij} = \frac{1}{2} \sum_{k=1}^{n} \left[\overline{Q}_{ij}\right]_k (h_k - h_{k-1})(h_k + h_{k-1}) \tag{2.90}$$

$$= \sum_{k=1}^{n} \left[\overline{Q}_{ij}\right]_k t_k \frac{(h_k + h_{k-1})}{2} \tag{2.91}$$

where the $(h_k + h_{k-1})/2$ term represents the distance from the midplane of the laminate to the midplane of ply k.

Lastly, the **D** matrix is termed the bending stiffness matrix. It relates the curvature of the laminate to the experienced bending moments.

$$D_{ij} = \frac{1}{3} \sum_{k=1}^{n} \left[\overline{Q}_{ij}\right]_k \left(h_k^3 - k_{k-1}^3\right)$$

$$= \sum_{k=1}^{n} \left[\overline{Q}_{ij}\right]_k \left(\frac{t_k^3}{12} + t\overline{z}_k^2\right) \tag{2.92}$$

$$\overline{z}_k := \frac{h_k + h_{k-1}}{2} \tag{2.93}$$

2.3.9 Inversion of ABD Matrix

Equation (2.87) relates the midplane values to the force and moment resultants using the **ABD** matrix. In most cases, the midplane strain and curvature are unknown and desired quantities. The applied, external loads on the laminate are, however, typically known, so the expression in Eq.

(2.87) requires inverting to express the strain and curvature as functions of the force and moment resultants,

$$\begin{bmatrix} \epsilon^{\mathbf{0}} \\ \kappa^{\mathbf{0}} \end{bmatrix} = \begin{bmatrix} \mathbf{A}^* & \mathbf{B}^* \\ \mathbf{C}^* & \mathbf{D}^* \end{bmatrix} \begin{bmatrix} \mathbf{N} \\ \mathbf{M} \end{bmatrix} \tag{2.94}$$

where the starred terms are defined as follows [15]:

$$[A^*] = [A]^{-1} - \left[-[A]^{-1}[B]\right]\left[[D] - [B][A]^{-1}[B]\right]^{-1}\left[[B][A]^{-1}\right] \tag{2.95}$$

$$[B^*] = \left[-[A]^{-1}[B]\right]\left[[D] - [B][A]^{-1}[B]\right]^{-1} \tag{2.96}$$

$$[C^*] = -\left[[D] - [B][A]^{-1}[B]\right]^{-1}\left[[B][A]^{-1}\right] \tag{2.97}$$

$$[D^*] = \left[[D] - [B][A]^{-1}[B]\right]^{-1} \tag{2.98}$$

2.4 Magnetostriction

A magnetostrictive material couples magnetism with mechanics and results in deformation under an applied magnetic field. Ferromagnetic materials such as iron, cobalt, and nickel display this behavior. Two ideas have been defined to categorize a magnetostrictive material's response to its environment: the Joule and Villari effects [5, 18]. The Joule effect describes the change in material shape under an externally applied magnetic field. The external magnetic field induces strain in the magnetostrictive material, and this strain is referred to as magnetostriction. The Villari effect describes the inverse of the Joule effect. When a mechanical load is applied to a magnetostrictive material, the magnetic susceptibility of the material changes. Sensors employing magnetostrictive materials utilize the Villari effect as mechanical energy is correlated to magnetic energy.

A two-dimensional constitutive relationship describes the coupling of the mechanical and magnetic states [19]:

$$\epsilon = s^H \sigma + d_c H \tag{2.99}$$

$$B = d_c \sigma + \mu^\sigma H \tag{2.100}$$

where ϵ is the strain, s^H the elastic compliance measured at constant H, σ the stress, H the magnetic field intensity, B the magnetic flux density, and μ^σ the permeability measured at a constant stress

σ [20]. In matrix form, the two-dimensional constitutive equations are expressed as [17]

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} = \begin{bmatrix} s_{11}^H & s_{12}^H & s_{13}^H & s_{14}^H & s_{15}^H & s_{16}^H & d_{11} & d_{21} & d_{31} \\ s_{21}^H & s_{22}^H & s_{23}^H & s_{24}^H & s_{25}^H & s_{26}^H & d_{12} & d_{22} & d_{32} \\ s_{31}^H & s_{32}^H & s_{33}^H & s_{34}^H & s_{35}^H & s_{36}^H & d_{13} & d_{23} & d_{33} \\ s_{41}^H & s_{42}^H & s_{43}^H & s_{44}^H & s_{45}^H & s_{46}^H & d_{14} & d_{24} & d_{34} \\ s_{51}^H & s_{52}^H & s_{53}^H & s_{54}^H & s_{55}^H & s_{56}^H & d_{15} & d_{25} & d_{35} \\ s_{61}^H & s_{62}^H & s_{63}^H & s_{64}^H & s_{65}^H & s_{66}^H & d_{16} & d_{26} & d_{36} \\ d_{11} & d_{12} & d_{13} & d_{14} & d_{15} & d_{16} & \mu_{11}^T & \mu_{12}^\sigma & \mu_{13}^\sigma \\ d_{21} & d_{22} & d_{23} & d_{24} & d_{25} & d_{26} & \mu_{21}^T & \mu_{22}^\sigma & \mu_{23}^\sigma \\ d_{31} & d_{32} & d_{33} & d_{34} & d_{35} & d_{36} & \mu_{31}^T & \mu_{32}^\sigma & \mu_{33}^\sigma \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \\ H_1 \\ H_2 \\ H_3 \end{bmatrix} \tag{2.101}$$

The coupling of stress, strain, magnetic field intensity, and magnetic flux density is linearly expressible through examining the coefficient matrix in Eq. (2.101).

2.5 Terfenol-D

Terfenol-D was developed at the Naval Ordinance Laboratory in 1971 through alloying terbium (Te), dysprosium (Dy), and iron (Fe) [21]. The name of the material was derived from its elements and the laboratory which developed it. The chemical composition of Terfenol-D varies based on the amounts of iron present such that the composition is $Tb_{1\text{-}x}Dy_xFe_{1.9\text{-}2.0}$ [17,22–24]. The manufacturer [25] cites the composition of the material as $Tb_{0.3}Dy_{0.7}Fe_{1.92}$. The rare-earth metals contribute to Terfenol-D's magnetostrictive properties, whereas the iron concentration contributes to the Curie temperature [21]. The Curie temperature, or Curie point, signifies the temperature at which a material's magnetic properties experience a notable change [26]. Typical ferromagnetic magnetostrictive materials have saturation magnetostriction values on the magnitude of $\lambda_s \approx 10^{-6}$. Terfenol-D, however, shows saturation magnetostriction ranging from $\lambda_s = 1.5 \times 10^{-3}$ to $\lambda_s = 2 \times 10^{-3}$ [17, 25, 27]. The difference in saturation magnetostriction deems Terfenol-D as a "giant magnetostrictive" material [28]. Select properties of Terfenol-D are often listed in literature as a range of values due to dependence on the crystal structure, geometry, magnetic conditions, and mechanical conditions. For example, TdVib (formally ETREMA), the manufacturer of Terfenol-D, lists the value of the elastic modulus as 25–35 MPa for the bulk material [25]. Characterization of

the properties of Terfenol-D listed in Tab. 4.1 has been limited due to the complex coupling of its properties with its environment.

2.6 Machine Learning

The processing and decision-making ability of an organism can be attributed to the presence of a brain structure. Inside the brain, neurons transmit information between one another across synaptic clefts, which is the region between the axon terminals and dendrites of two neurons [29]. Figure 2.5 displays the structure of an anatomical neuron consisting of an axon, axon terminals, and dendrites. Neurons are the basis of the brain and allow for information to be transmitted, interpreted, and perceived. Electrical signals, called spikes or action potentials, transmit across synapses that occur at the connection between the axon terminals of a transmitting neuron and the dendrites of a receiving neuron. The receiving neuron interprets the spikes based on some of its attributes, including timing and strength [29]. Further manipulation occurs inside the neuron where multiple spikes accumulate to form a single signal to transmit to a neighboring neuron.

As computational abilities increased, a desire emerged to develop an artificial system capable of mimicking the decision-making and processing ability of a mammalian brain. Following the creation of *The Bombe* that deciphered the Enigma code used by the German army in World War II, Alan Turing published an article titled "Computing Machinery and Intelligence" [30] in 1950 to explain how intelligent machines could be created [31]. The aptly named Turing Test was proposed to determine if a machine could be considered intelligent. The Turing Test deemed a machine "intelligent" if an external, observing machine could not distinguish the interaction of a human with

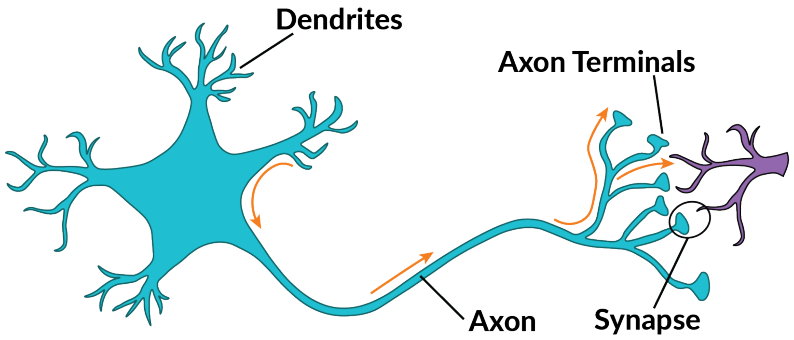


Figure 2.5: Diagram of an individual biological neuron. [29]

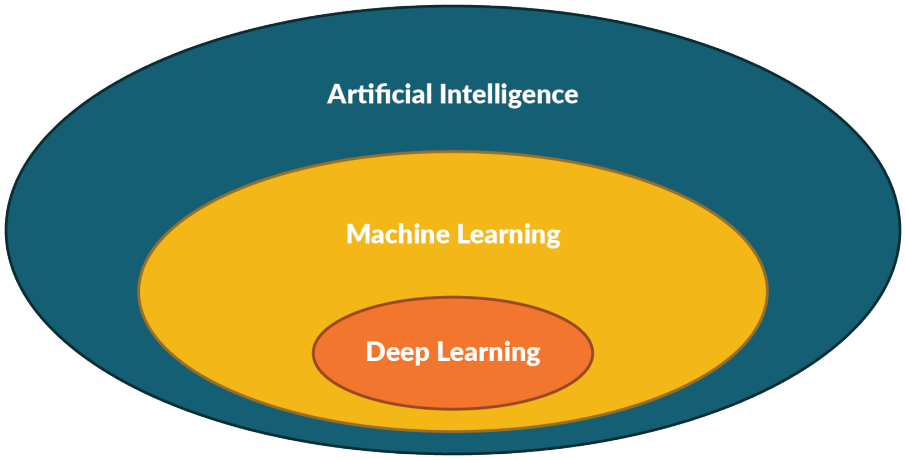


Figure 2.6: Artificial intelligence has become a general term to encompass and include more specific applications.

another human and the machine [31]. The term "artificial intelligence" was coined six years later in 1956 by a group of mathematicians and scientists at the Dartmouth Summer Research Project on Artificial Intelligence (DSRPAI) [31,32]. At this gathering, mathematicians and scientists discussed the current and future computer intelligence systems trends and laid the groundwork for future machine learning advances. The development of intelligent machine algorithms has flourished since 1956 with an increase in computational resources [33].

Initial work with artificial intelligence (AI) investigated its application to game strategy, such as that involved with checkers, chess, and Go, because a known, desired outcome is identified in each game—to win [31,34]. Healthcare, household commodities, and cell phones have made AI a standard, everyday technology through its use for decision-making, providing daily reminders, and acting as a personal secretary completing tasks via only verbal commands [34, 35]. AI technology has been impregnated into almost every technology to learn about and optimize a user's experiences. These intelligent systems need to interpret the system's surroundings utilizing quantitative or qualitative data for decision-making and learning. However, AI has come to serve as a generalized label for related concepts that invoke methods to mimic the ability of the mammalian brain to solve problems.

Machine learning (ML) and deep learning (DL) extend the generalized idea of machine intelligence by applying learning methods and highly connected networks to resemble further the neural structure of a mammalian brain [34, 35]. Figure 2.6 illustrates the relationships between AI, ML, and DL. In ML, algorithms analyze data to "learn" trends and make predictions based on the

input data provided; large datasets are essential to training an algorithm effectively [34]. The use of large datasets ("big data") aims to reduce the likelihood of supplying an ML algorithm with an insufficient amount of data. DL, a subset of ML, utilizes a structured neural network (NN) architecture for prediction making. While data processing may be applied, human intervention is reduced in DL compared to ML as artificial neural networks (ANNs) can extract useful information from data using pattern recognition techniques [31, 34]. Layers of artificial neurons, called perceptrons or nodes, are connected in an ANN resembling the mammalian neuron interaction between axons and dendrites [35]. Figure 2.7 illustrates the operation of a linear perceptron where inputs are combined with a weight function to output a binary value based on a threshold function [36]. A linear perceptron calculates a comparison value y as

$$y = \sum_{i=1}^{n} w_i x_i + w_0 \tag{2.102}$$

where w_i is a weight associated with each input x_i, and w_0 is an intercept value taken from a bias value applied to each perceptron [21]. The weight and bias terms are determined using an iterative scheme. The linearly combined comparison value y is mapped to a new value using a threshold, or activation, function f. Activation functions may take many forms, and common functions include the Heaviside and sigmoid function:

$$\text{Heaviside function:} \qquad s(a) = \begin{cases} 0, & a > 0 \\ 1 \end{cases} \tag{2.103}$$

$$\text{Sigmoid function:} \qquad y = \text{sigmoid}(\alpha) = \frac{1}{1 + \exp(-\alpha)} \tag{2.104}$$

Data passes into a perceptron from a previous layer, undergoes appropriate modification, and is then passed to the next layer as illustrated in Fig. 2.8. While an ANN can also be considered an ML method, the distinction between an ANN used for ML and DL is typically the number of hidden layers in the network. One hidden layer in the ANN is generally associated with ML as in the NN in of Fig. 2.8. However, a DL ANN includes multiple hidden layers and may consist of thousands of nodes [21, 33]. An ANN, however, has no accepted architecture in terms of the number of hidden layers or nodes within those layers, and parametric studies are needed to determine the best combination for a given problem. The performance of an ML algorithm is determined by

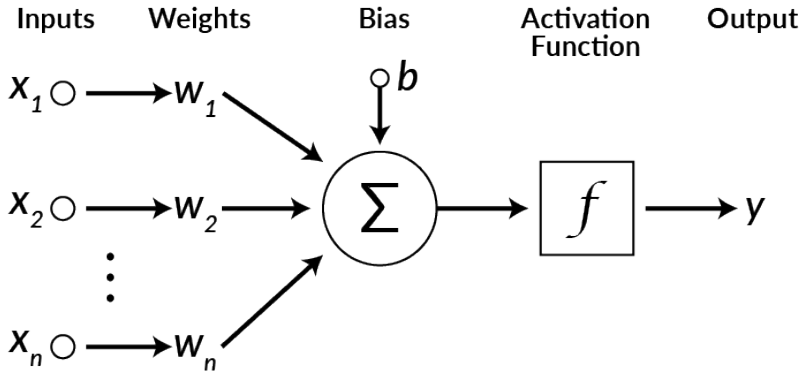


Figure 2.7: A perceptron combines a series of inputs in addition to a bias to produce an output signal.

metrics such as accuracy, precision, and F1-score. The shared goal of both ML and DL lies in making predictive models for a problem given a set amount of data.

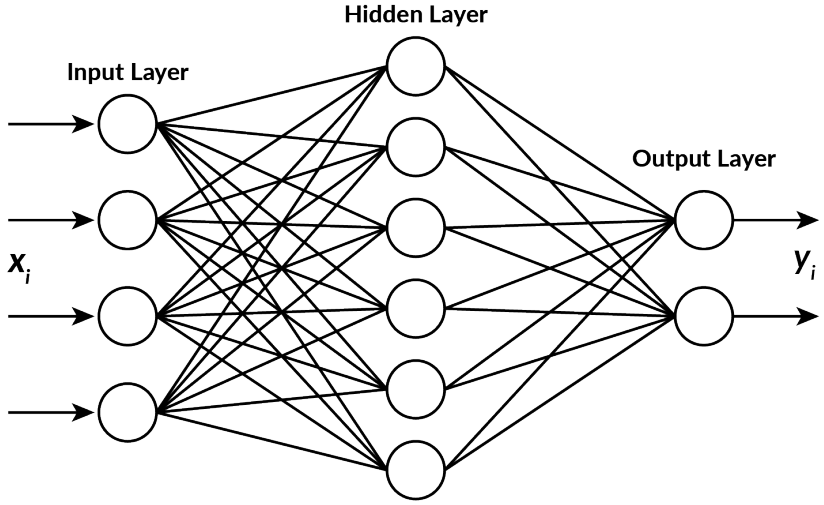


Figure 2.8: Example of an ANN with a 3-4-2 architecture (three input, four hidden layer, and two output nodes) demonstrating the connectedness of perceptrons and the flow of information.

2.6.1 Datasets

The learning process associated with machine learning consists of two steps: a training and testing phase. Each phase introduces a unique set of data to the ML algorithm; a large dataset is

typically divided as shown in Fig. 2.9 into subsets for use in the training and testing phases. The appropriately named training phase focuses on introducing data to the algorithm for "learning" purposes. A standard percentage for dividing data into training and testing sets does not exist, but training sets should be greater in number than the testing set to aid in "learning." The training set should encompass the breadth of the problem-at-hand to ensure that the algorithm can make accurate predictions for the scope of the problem [21]. The training set may be further divided into a specified training and validation set used to test the generalized model with known instances before testing [21].

Cross-validation is one method used to divide the overall training set into training and validation sets. The premise of this method is introducing the algorithm to multiple repetitions of the same dataset but in distinctive combinations such as the k-fold cross-validation scheme presented in Fig. 2.9 [33]. A validation step allows for a performance check of the algorithm before implementing the algorithm in its test environment. If the validation step produces undesirable results, parameters such as the initial weights, bias values, and activation functions for an ANN can be adjusted to alter performance. Once the training set has been introduced, and the "best" model has been found, the testing phase of machine learning begins using never-before-seen data. If the ML model fails to perform as expected at this stage, the learning process can be restarted from the training phase to improve the hyperparameters. Underperformance can suggest that either the hyperparameters were improperly set or that the training set was inadequate for the problem-at-hand.

2.6.2 Supervised learning

One type of machine learning is supervised learning. Per its name, the algorithm "learns" via a controlled, supervised approach. A set of known inputs x_i with respective outputs y_i are introduced to the algorithm for pattern association between the given instances with the respective outcomes [33,38]. Because the input data must be correlated with its respective output, the data used in supervised learning is called *labeled data* [38]. The use of labeled data occurs only in the training phase, and the algorithm trains by reducing the training error between the calculated outcome and actual value,

$$\text{Training Error} = \mathbb{E}_{\text{training}} = \frac{1}{n}\sum_{i=1}^{n} c\left(\mathbf{x}_i, \mathbf{y}_i, f\left(\mathbf{x}_i\right)\right) \tag{2.105}$$

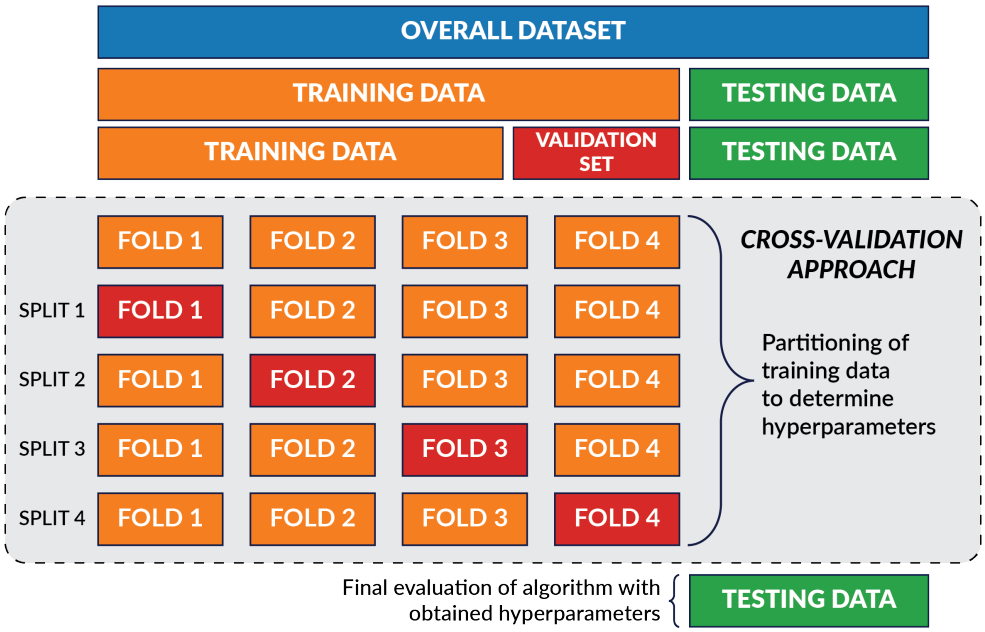


Figure 2.9: Two sets of data are necessary when using a machine learning algorithm: training and testing data. A cross-validation approach can be implemented to increase the amount of data introduced to the algorithm during training. [37]

where the vector $\mathbf{x}_i$ is the instance (input), $\mathbf{y}_i$ is the respective outcome, and $f(\mathbf{x}_i)$ is the trained model [38]. Empirical error, mean square error (MSE), root mean square error (RMSE), and least square estimates are frequently used to quantify the performance of the predicted results with the actual results [21]. When *overfitting* occurs, the model trains to accurately replicate results for the trained data but performs less accurately when new, unseen testing data is introduced to the algorithm [38]. This decrease in prediction accuracy is due to minimizing the training error to an undesirable minimum.

Supervised learning problems are divided into two categories: classification and regression. In classification problems, information about an instance (e.g., customer, location, item) is inputted into the model. The classification algorithm then assigns a discrete class to the data point based on associations learned in the training phase between the presented information and the desired output classes. Algorithms can utilize a hyperplane to divide the region between the classes as illustrated by the red line in Fig. 2.10a for a two-class problem. Classes are often modeled within algorithms as integers to reduce complexity; the use of binary systems improves computational efficiency [38]. Regression problems analyze introduced data to produce a function that outputs a

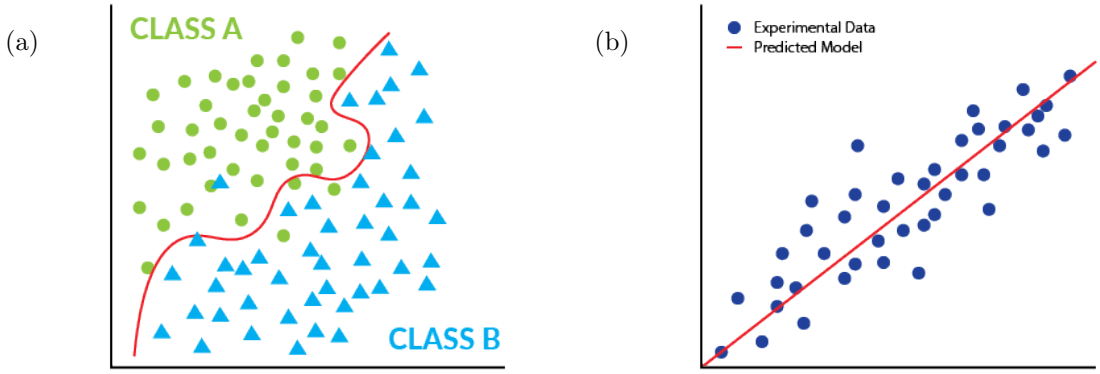


Figure 2.10: Two types of supervised learning problems: (a) classification; (b) regression.

continuous value. The predicted relationships can follow linear (as shown in Fig. 2.10b), polynomial, or nonlinear functions based on the chosen architecture of the algorithm [21]. Data fitting can occur where the predicted outcomes closely correspond to the training data and differ from relationships observed in the testing data, i.e., *overfitting*.

2.6.3 Unsupervised learning

Unlike supervised learning, which uses *labeled data*, unsupervised learning uses *unlabeled data* to train a model. Two applications of unsupervised learning are association and clustering [36]. Clustering tasks, similar to classification problems in supervised learning, separate instances into k groups or clusters. Algorithms can either be fed the desired number of clusters or be allowed to discern patterns within the data and determine the appropriate number of clusters [38]. Figure 2.11 illustrates the idea of clustering data into $k = 2, 3, 4$ clusters. The algorithm determines the cluster to which a point is assigned by its distance from other data points. The Euclidean distance is one measure employed to calculate the distance between two points, $\mathbf{x}_i$ and $\mathbf{x}_j$ [38]:

$$d\left(\mathbf{x}_i, \mathbf{x}_j\right) = ||\mathbf{x}_i - \mathbf{x}_j|| = \sqrt{\sum_{s=1}^{D} \left(x_{is} - x_{js}\right)^2} \tag{2.106}$$

Semi-supervised learning combines the concepts of supervised and unsupervised models [33, 38]. This technique employs a mixed set of *labeled* and *unlabeled data* during training. The *labeled data* identifies trends, and the algorithm then applies the knowledge "learned" from the *labeled data* to the *unlabeled data* in addition to combining unsupervised learning strategies to develop

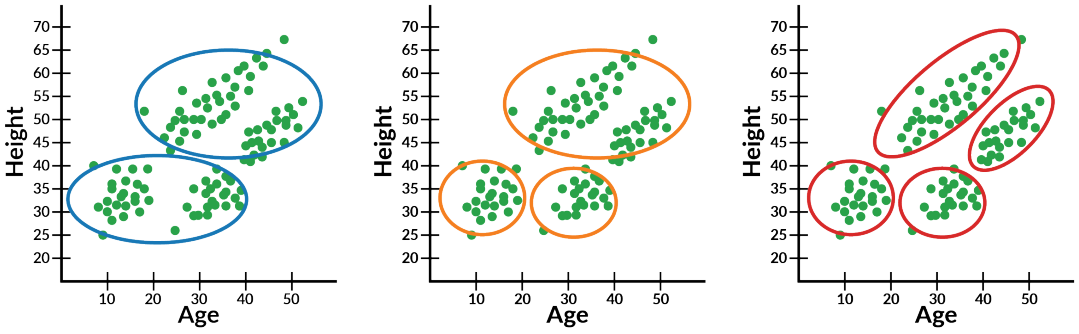


Figure 2.11: Example of a clustering problem with two, three, and four classes. [38]

a model. Applications of semi-supervised learning combine supervised and unsupervised learning; ergo, classification, regression, clustering, and dimensionality reduction are possible problems for a semi-supervised learning algorithm.

2.6.4 Reinforcement learning

Reinforcement, or reinforced, learning differs from the previously discussed types of machine learning through the use of real-time information to make decisions. This type of algorithm utilizes a feedback loop, as shown in Fig. 2.12, between the agent (decision-making algorithm) and the environment [21,39]. The agent interacts with the environment via issued actions based on analysis of the current state of the environment. The action aims to accomplish a set task or goal, and a given policy predefines the possible actions issued by the agent. The action alters the state of the environment, and a reward is sent to the agent along with information about the new state of the environment. The agent seeks to produce an action that maximizes the reward. By achieving a maximum reward, the agent has issued the best action. This feedback loop occurs until the task or goal is achieved [39]. A critic can be introduced to the algorithm to judge only the past decisions made by the agent and give feedback [21].

Applications of reinforcement learning primarily concern problems with changing situations and large state spaces [33]. Game playing (e.g., chess, backgammon, video games) and autonomous driving are two examples of problems with these situations. Because of its functionality and methods, reinforcement learning has yet to find application in research focusing on the application ML to SHM or NDE. The ability to analyze real-time data would prove useful in SHM; however, most work concerning component monitoring or evaluation does not require continually altering the state of the

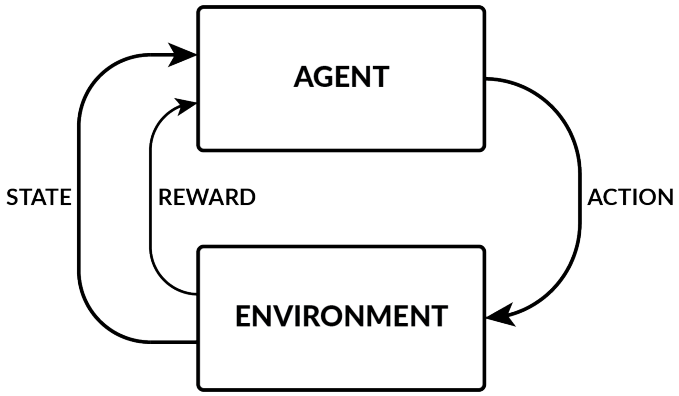


Figure 2.12: Reinforcement learning acts as a feedback loop using information the environment for the algorithm to make its next action. [38]

environment via actions. The current state of the SHM/NDE field only seeks to obtain information about a system to comprehend and develop predictions rather than manipulating the environment of the system to reduce possible damage.

2.7 Structural Health Monitoring and Nondestructive Evaluation

All materials inherently accumulate damage during their lifetime due to usage. Ideally, materials are assumed perfect, meaning that their lattice structures are perfectly intact and not altered in any manner [40]. Farrar and Worden [41] generalized the definition of damage to encompass any introduced alterations in a system that affects its performance at either the current or future state. Regarding a mechanical system, damage describes changes in the material or geometric properties that could prove unfavorable in current or future performance [41, 42]. A material is susceptible to damage at any point in its lifespan, and the length-scale of damage ranges from the microscale to macroscale. Damage initiates in the crystal structure of a material when the elastic limit is surpassed and crystal plasticity begins. Damage in the lattice, often referred to as defects or flaws, transpires due to changes in the arrangement of atoms within the crystal lattice structure. Figure 2.13 illustrates examples of defects which include vacancies (the absence of an atom in the lattice), interstitials (the addition of an extra atom in the lattice), dislocations, and grain boundaries [43].

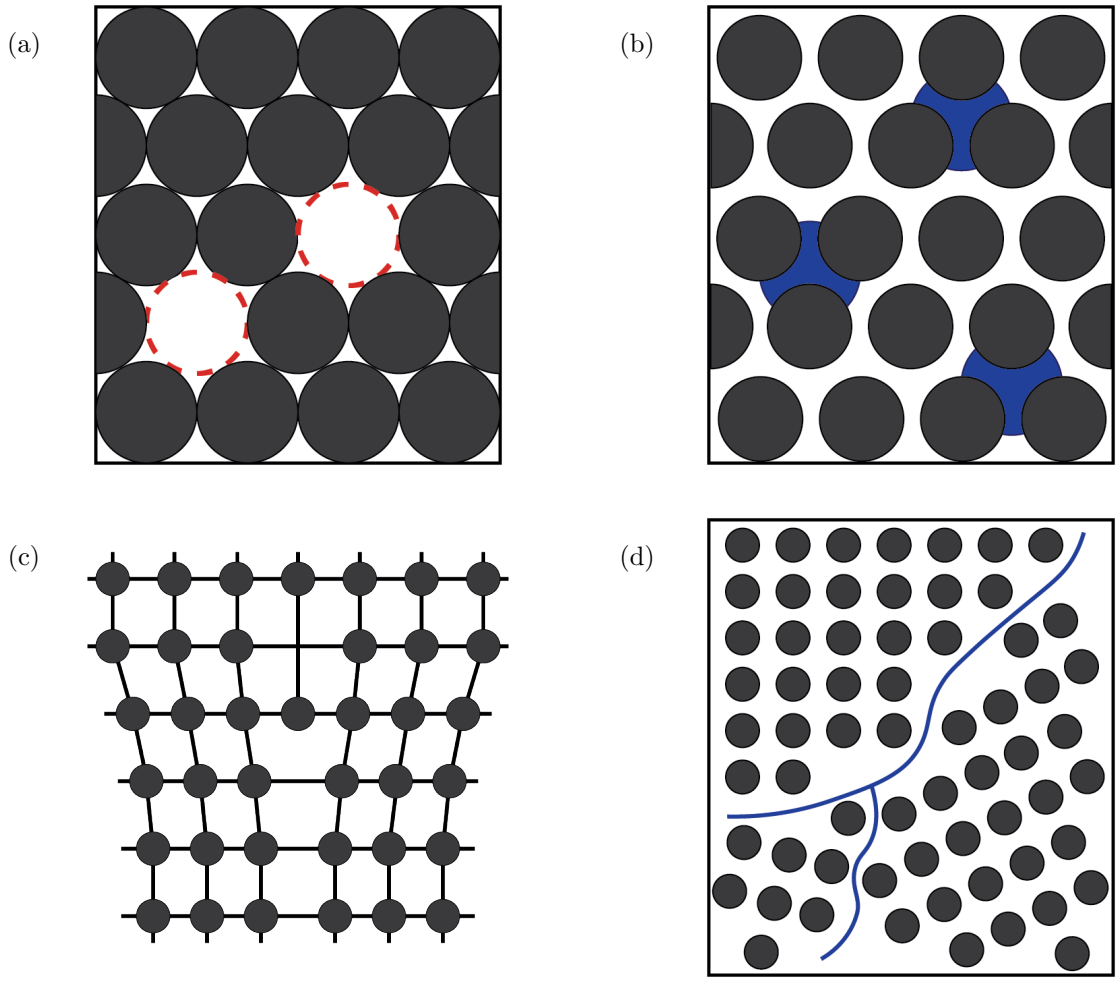


Figure 2.13: Four types of defects that occur on the microscale of a material and can eventually lead to macrolevel damage: (a) vacancy of atoms (point defect); (b) interstitial of foreign atom (point defect); (c) edge dislocation (line defect); (d) grain boundary (planar defect);

Material-level defects accumulate as a material or system encounters continued loading. These defects eventually turn into damage such as cracks or voids at the system-level. At this point in a structure's life, the system-level damage does not typically result in a total system failure, but this damage level can influence the performance of the system. Mechanical properties such as stiffness and hardness can deteriorate as the presence of system-level damage increases in a system [44]. Like material-level defects, system-level damage grows as loading continues and may join together to form larger-sized damage. For instance, accumulated dislocations can form into cracks, which become stress concentrations that weaken a material. "Failure" of a system occurs when its performance has reached a level deemed no longer safe or acceptable because of an accumulation of adverse damage.

Damage detection processes are implemented to observe a structure for damage. The ultimate goal of applying damage detection methods is reducing the probability of a system unknowingly reaching a state of failure. The operation of a system and the type of materials used in the design determine the appropriate and applicable methods for detecting and identifying damage. Specialized damage detection disciplines include structural health monitoring (SHM), condition monitoring (CM), nondestructive evaluation (NDE), health and usage monitoring system (HUMS), statistical process control (SPC), and damage prognosis (DP) [45]. SHM involves collecting periodic data about a mechanical system to analyze its performance. In-situ measurements are often recorded using a real structure during operation to obtain accurate data; therefore, SHM can be regarded as an "on-line" method [46]. On the other hand, NDE is considered an "off-line" technique because evaluation occurs during a non-operational state. In a non-operational state, components can be removed and isolated from a structure for evaluation using methods such as acoustic emission (AE), X-ray scanning, and computerized tomography (CT) scanning. Unlike other damage detection methods, NDE examines a system with techniques that do not impact its future performance. Intentional damage, e.g., holes, are not introduced into the component to examine the internal structure [47]. Farrar and Worden [45] suggested that NDE utilizes an *a priori* knowledge of damage. NDE focuses on characterizing and evaluating the extent of damage based on a known amount of operation [41]. NDE occurs more sporadically and often on larger intervals of time, cycles, or other chosen measurable parameters than SHM.

A generalized four-step process was proposed by Farrar et al. [48] to define any SHM/NDE process:

1. operational evaluation,
2. data acquisition and cleansing,
3. feature selection, and
4. statistical model development.

Information and results are passed along in this generalized process as illustrated in Fig. 2.15. Farrar et al. [41] suggested that machine learning techniques are ideal for the fourth step, *statistical model development*, to combine and statistically analyze collected data. Rytter [49] proposed five damage state descriptions that coincide with the aims of the *statistical modeling development* step of a SHM process:

1. Detection/existence: is there damage in the structure?
2. Location: where is the damage in the structure?
3. Type/classification: what kind of damage exists in the structure?
4. Extent/assessment: how severe is the damage?
5. Prognosis: how much useful life remains for the structure?

Any combination of these questions that describe the damage state of a system may be addressed in a SHM/NDE process. The amount of information about the damage state increases for each question based on the order listed above. For instance, the prognosis of a damage state requires information about the type of damage in the structure and the severity of the damage—assuming damage exists (existence).

2.7.1 Embedded Particle Sensing Methods

Ryan and Miller [50] first proposed designing a system with embedded particles for material evaluation in their 1973 patent. For the next 15 years, subsequent work investigated the application of introducing particles to detect explosive material. The application of embedded particles extended to detecting damage in the form of voids and bond failure in the 1990s [51, 52]. Weisensel [4] discussed a new approach to nondestructive evaluation utilizing magnetostrictive particles embedded within a material deemed "magnetotagging". The constitutive relationships of magnetostriction, Eqs. (2.99) and (2.100), couple the mechanical stress state of a material with its magnetic state. Krishnamurthy et al. [53] compared the sensing voltages obtained from an Euler-Bernoulli-derived

model and experimental work. A deformation compatibility model stated that the deformation of a infinitesimal element was a function of the free elongation of the magnetostrictive material, the deformation caused by the reaction force(s), and the strain of the magnetostrictive region [54–56]:

$$\delta_{\Lambda} = \delta_f + \delta_{dx} \tag{2.107}$$

A loading line related the free strain, Λ, to the mechanical stress in the magnetostrictive particles,

$$\Lambda\left(H,\sigma\right) = \frac{\nu_p E_p\left(H,\sigma\right) + \left(1-\nu_p\right) E_m}{\left(1-\nu_p\right) E_m E_p\left(H,\sigma\right) K_0\left(H_0,\nu_p\right)} \left(\sigma - \sigma_0\right) \tag{2.108}$$

where the elastic modulus E_p is a nonlinear function of the magnetic field intensity and stress state. Recent investigations of this NDE method employed experimental testing to correlate induced sensing voltages with the damage state of a material [6, 7, 57]. Haile et al. [3] found that embedding particles in a CFRP negatively affected the breaking stress (a 15% reduction) though increased the elastic modulus by nearly 20% under fatigue loading. Williams [27] examined the magnetic flux of experimental laminates statically loaded to predict the presence of damage. Results were compared using acoustic emission testing to correlate the flux and sensing voltage.

Recent studies utilized a standard fabrication and testing method to experimentally study this NDE approach [3, 6, 7, 57]. CFRP beams with dimensions of 9.75 $\times$ 1 inch (247.65 $\times$ 25.4 mm) made from Hexply AS4/3501-6 were fabricated of various stacking sequences ranging from two to 14 plies. Terfenol-D particles ranging in diameter from 38 to 300 μm were distributed over the middle six-inch (152.4 mm) domain of the beams. Two techniques distributed the particles across the region. A hand distribution technique used a straight-edge tool to spread the particles across the plane of the laminate. A trenching method sought to improve the distribution and yield a more uniform spread. Delamination damage was simulated using embedded Teflon tape squares.

The experimental setup in Fig. 2.14 employed a sensing circuit to generate and pickup magnetic fields. The sensing module structure surrounded the CRFP beam being evaluated. An actuation coil was connected to a power amplifier and waveform generator to produce an actuating magnetic field. A second coil in the sensing module, a sensing coil, converted the magnetic flux density picked up from the magnetostrictive material to an induced sensing voltage. The sensing voltage data was measured and stored using an attached data acquisition system for later analysis.

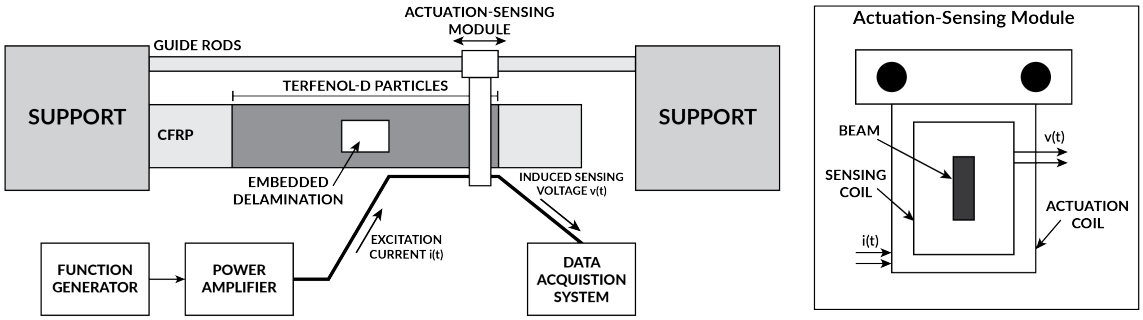


Figure 2.14: Experimental setup of the embedded magnetostrictive particle NDE method.

The sensing voltage data was plotted against its position along the length of the beam to observe the change in voltage. The method illustrated that the presence of Terfenol-D influenced the magnitude of the induced sensing voltage.

2.8 Machine Learning Application for SHM/NDE

The concept of *statistical modeling development* encompasses numerous mathematical methods that can be applied for prediction-making using data analysis trends. Machine learning has become a useful technique because of its basis in statistical principles. Algorithms have been successfully implemented for addressing each of the five damage state descriptions [49]; however, certain types of algorithms are better suited for addressing particular damage descriptions. Unsupervised learning algorithms can address questions regarding the existence and location of damage in a system. These are the most fundamental descriptions of a damage state due to their lack of need for prior, contributing information [41]. Determining the existence and location of possible damage can involve examining a dataset for anomalies, i.e., outlier detection. The remaining three questions (the type of damage, the extent of damage, and an estimation of the remaining life for a system—prognosis) are best determined using a supervised learning algorithm as they are more complex questions requiring more knowledge about the system [41]. Because these three questions require the most knowledge about a damage state, introducing information into the algorithm for training is necessary to aid in "learning".

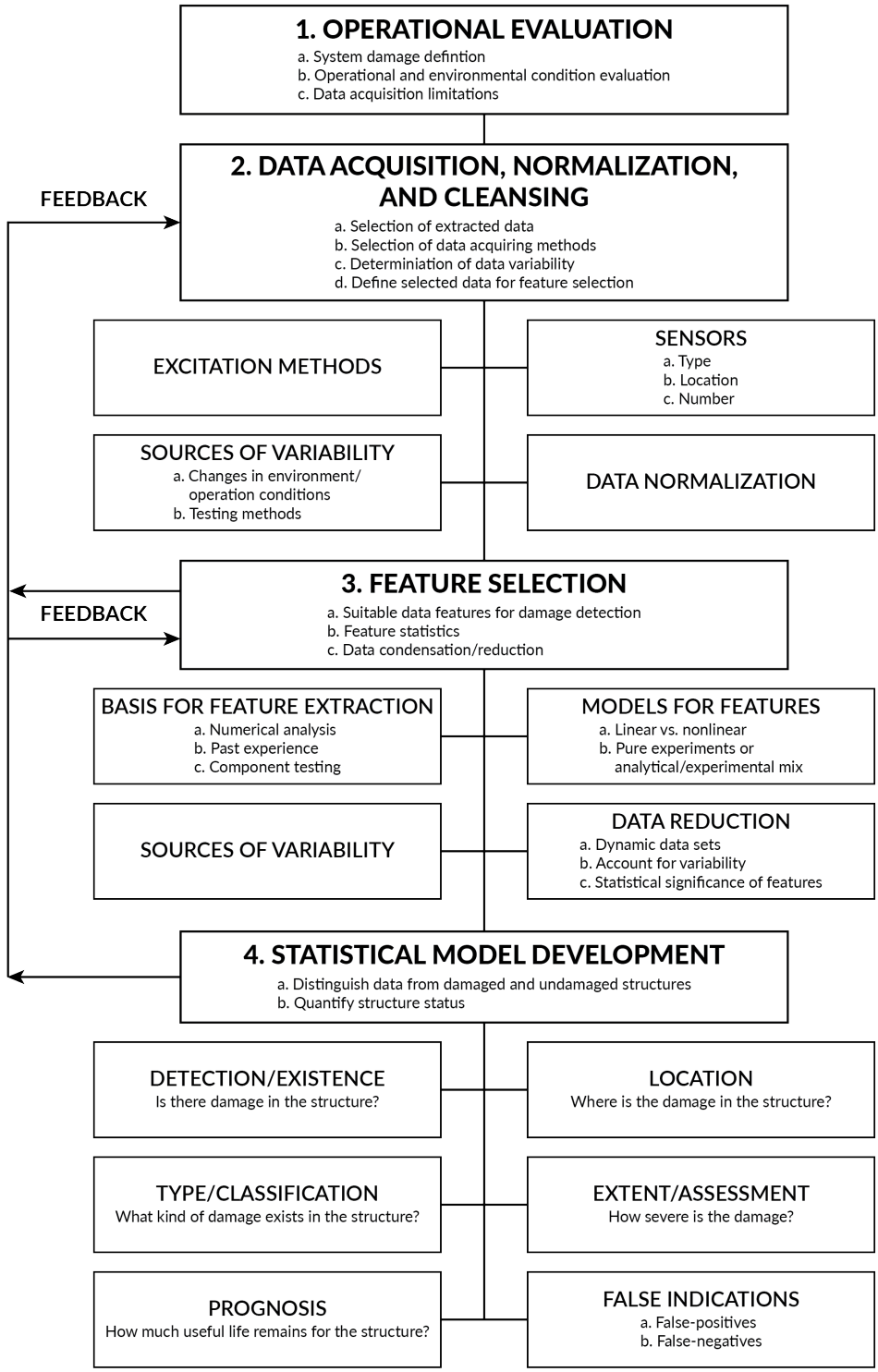


Figure 2.15: Farrar et al. proposed a SHM approach using ML that involved four steps: operational evaluation, data acquisition and cleansing, feature selection, and statistical model development. [48]

2.8.1 Non-composite Civil Structures

Civil structures commonly employ SHM techniques for damage state analysis to reduce the probability of catastrophic failure. Monitoring these structures is paramount because of the constant operational and environmental variabilities to which the structures experience [42]. Failure of these systems at any point in their lifespan can result in significant casualties and financial loss. Damage detection and classification are the predominate damage descriptions/questions from [49] addressed in monitoring civil structures. Basic structural elements ranging from plates [58–60] and beams [61] to large-scale systems like buildings and trusses [62–64] applied machine learning algorithms for monitoring. Classification and identification algorithms were used to detect types of damage and classify the structures of railways [65] and bridges [62, 66] Prognosis and predictive modeling with ML was utilized to reduce the downtime of machinery [67].

2.8.2 Composite Materials

2.8.2.1 General Composites

The anisotropic behavior of a composite material exists due to the many factors associated with its structure; the complexity lends to utilizing machine learning algorithms for analysis. A two-dimensional composite design was investigated using an element-based approach that utilized "stiff" or "soft" elements [68, 69]. Gu et al. [69] used ML to classify random arrangements of 8x8 and 16x16 element grids produced using finite element methods (FEM) as either "good" or "bad" based on the ranking of the grid's predicted toughness. The location of the soft and stiff elements was ultimately found to determine the overall toughness (strength) of the composite. Soft material was found to surround introduced cracks to mitigate the localized stress increase around the crack; this increased the overall toughness of the composite. For maximum stiffness, the soft elements were aligned in parallel to the unidirectional loading direction. In contrast, alignment of the soft elements in series with the loading direction resulted in the softest, or most compliant, laminates [68].

Tensile testing of open-hole CFRP laminates paired with a probabilistic neural network resulted in a feasible method for predicting the open-hole tensile strength [70]. Unlike other algorithms that require large datasets, a probabilistic neural network was shown to predict the tensile strength to within 4% of the experimental value with a limited amount of data—52 fabricated specimen were studied, and only 36 were used to train. Additional data was created by manipulating

the experimental data based on observed trends in the experimental results. Jiang et al. [71] used experimental data in a NN to predict mechanical and wear properties of a composite. A parametric study on the number of hidden layers and output nodes found that the complexity of the investigated problem determines the number of each of these parameters. A saturation point can be achieved regarding the number of hidden nodes in a NN. Complex problems performed better using a one-output architecture to reduce the consequences of associating multiple inputs with multiple outputs. Architectures with multiple outputs were found to apply to more straightforward problems. The feature inputs and nature of the prediction contribute to the complexity of the problem and its classification as "complex" or "simple." For instance, a complex problem could involve a fatigue analysis as it requires large datasets to fully describe the problem, and fatigue is nonlinear in nature, especially for composites.

2.8.2.2 SHM/NDE of Composites

Composite materials suffer from progressive failure that originates with the cracking or breaking of individual constituent materials. Multiple factors require consideration to determine the proper damage detection technique employed. For example, surface cracks can be detected using magnetic particles applied to the surface of a ferromagnetic material based on the idea that cracks affect the local magnetic field of a ferromagnetic material [47, 72].

Composite materials, however, often combine non-ferromagnetic constituents. A polymer matrix with graphite or glass materials is combined to produce a FRP; particulate composites can consist of various particulate fillers, including metals, ceramics, glasses, and polymers. Many evaluation techniques used for homogeneous materials can be applied to composites such as visual, acoustic, optical, imaging, electromagnetic, and modal analysis techniques [72, 73]. While expensive to implement, imaging techniques are particularly useful for composite damage detection as they provide a visual image of the internal structure of the material. Other methods such as acoustic emission and modal analyses interpret the response of a material to an applied signal. A benefit of composites in terms of damage detection is controlling their structure through the constituent materials selected and the stacking sequence.

Detection Detecting the presence of damage represents the first step to a SHM system, according to Rytter [49]. When investigating the use of ML for damage detection, damage is often intentionally

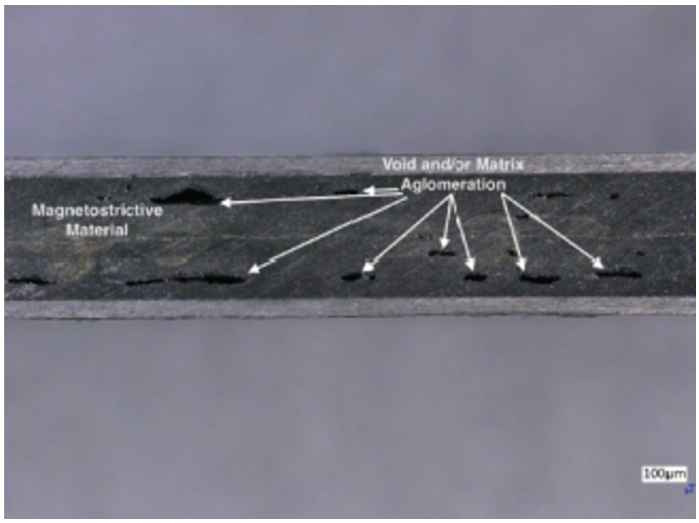


Figure 2.16: Myers and Banerjee [74] illustrated damage with a CFRP using a cross-section image. Damage, specifically void and agglomeration regions, can be seen by the darkened regions in the image.

introduced into a system to expedite the damage evolution process and yield repeatable experiments. Scanning techniques allow for the visual determination of the presence of damage. Marani et al. [75] investigated the use of thermography with glass fiber reinforced polymers (GFRP). Thermal images were captured of GFRPs containing drilled holes exposed to sinusoidal heat signals. The existence of damage was suggested to affect the rate at which heat transferred through a material, so the time delay involved with the transfer of the applied heat through the GFRP was visually captured by comparing a series of thermal images. Unsupervised clustering algorithms were employed to differentiate the homogeneous, i.e., undamaged, and damaged regions of the GFRP specimen.

Sammons et al. [76] employed computerized tomography (CT) scanning to detect the presence of damage. Simulated data was generated by adding delamination-shaped regions of varying size, shape, and intensity in the background of undamaged CFRP images. A convolutional neural network (CNN), used for image recognition, predicted the presence (and location) of 89.17% of the delaminations in the simulated testing data. The selected patch size employed in the CNN algorithm reduced the detection rate of large and small delaminations.

Acoustic emission was utilized by Vitola et al. [77] to detect "on-line" damage in a CFRP plate using piezoelectric transducers attached to its surface. One of the attached transducers generated a signal that traveled through the plate, and the other transducers acted as sensors—a common practice for AE methods. Of six k-nearest neighbor (k-NN) algorithms examined, the fine

and weighted k-NN algorithms yielded the highest classification accuracy of over 90% data points correctly predicted.

Rodriguez-Cobo et al. [78] investigated the differences in performance of embedded and surface-mounted fiber optic sensors. Fiber Bragg grating (FBG) was selected as the employed fiber optic sensor because of its ability to detect strain by a change in length of the physical sensor [79,80]. A hierarchical temporal memory (HTM) algorithm and an ANN were trained and tested using three datasets: data from only the embedded FBGs, data from only the surface-mounted FBGs, and data from all the FBG sensors. The HTM algorithms outperformed the ANNs as they are suited for large datasets.

Location With the existence of damage known, ML algorithms can be combined with SHM/NDE techniques to locate damage. Determining the location of delaminations was found to be prevalent in previous literature because of the discernible size of delaminations compared to cracks.

Modal analyses have been utilized to determine the location of damage because "natural frequencies of any structural component contain the information regarding the presence of delamination along with its size, shape, and location" [81]. Chakraborty [81] generated 201 20-ply CFRP laminates using finite elements that contained an embedded delamination located at their midplane. The delamination was modeled by reducing the elastic modulus of the elements in the simulated delamination region so that the region of the delamination was softer than the surrounding elements. Valoor and Chandrashekhara [82] attempted to determine the location of midplane delaminations using a thick composite beam analytical model. The beams were modeled with different boundary conditions (clamped-clamped, simply-supported, clamped-free), length-to-height ratios, and stacking sequences to determine how the boundary conditions affected the natural frequencies of the beams. Natural frequencies were calculated to input into a NN to predict the size and location of the delamination.

Zhang et al. [83] presented three algorithms (a graphical method and two ANN techniques) to determine the size and location of delaminations in a beam. The ANNs utilized the first seven frequency changes to predict the interface number containing the delamination, the x-location along the beam, and the delamination's size. The first ANN technique used only one ANN to make predictions. In contrast, the second ANN technique employed a surrogate ANN system consisting of seven ANNs that each generated one frequency using supplied information about the delamination:

interface number, x-location, and size. The seven frequencies outputted by the seven ANNs were then fed into a separate ANN to predict the delamination information. The method employing only one ANN had an average error for location prediction of 3.11%, whereas the surrogate method had an average error of 3.34%. The increase in error could be associated with the increased complexity of the data flow through the surrogate system.

Acoustic emission testing was implemented by Zhao et al. [84] to determine the abscissa and ordinate values of a delamination in a two-dimensional plate. Acoustic signals were applied at randomized, but known, points along the surface of experimental and FEA modeled laminates to represent damage. Predicting the delamination coordinates involved recording measurements along each axis to determine the specific coordinates. The FEA data yielded a maximum error between the actual and predicted values of 5.00% and 5.60% in the abscissa and ordinate directions. The experimental data, though, had greater error values of 7.40% and 9.40%, respectively. This increase could be attributed to the experimental composites being fabricated with an unknown number of plies and unknown orientation angles.

Liu et al. [85] presented an acoustic emission-based location prediction method using the length across the path of a delamination. CFRPs were embedded with 12 piezoelectric transducers in the configuration shown in Fig. 2.17 to act as a set of six actuators and six sensors, and a crack was introduced to encourage the growth of a delamination during fatigue loading. X-ray imaging quantified the size of the delamination caused by the introduced crack. Regression algorithms related the acoustic emission data to the predicted length of a delamination.

Classification Because various damage mechanisms can occur within a material, determining the specific type of damage offers insight into the response of a material. When experiencing loading for an extended period of time, damage naturally occurs and grows; fatigue testing has been utilized in experimental work to provoke damage growth and development in a material. Dabetwar et al. [87] used data obtained from the NASA Ames Prognostics Data Repository [88] which tested dogbone CFRPs embedded with piezoelectric transducers as acoustic emission sensors as shown in Fig. 2.17 under fatigue loading. Data was classified using a three-class problem based on the number of cracks in the laminate. X-ray images were taken to compare and validate the predicted class by ML algorithms to the correct class. The accuracy of damage classification for the three-class problem using frequency data averaged 80% for all but on one (SVM) of the tested algorithms. Larrosa et

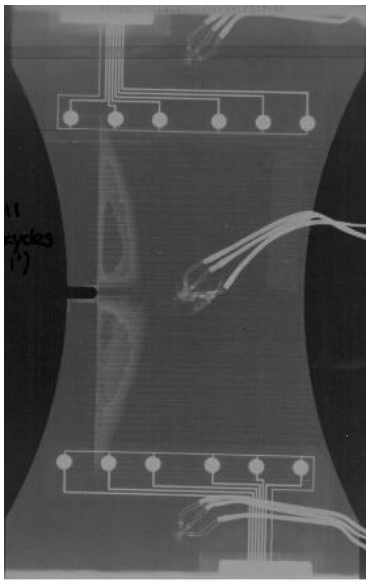
(a)  (b) 

Figure 2.17: NASA Ames Prognostics Data Repository [86] experimental fatigue testing setup with 12 embedded AE sensors used for collecting data during loading; (a) an image of the notched CFRP laminate with the sensor array and (b) an X-ray image of a CFRP laminate illustrating the 12 AE sensor array.

al. [89] also utilized CFRP fatigue data from the NASA Ames Prognostics Data Repository, but their work focused on statistical feature extraction of both time and frequency domain data. Six parameters were extracted and combined to yield 511 inputs for the tested ML algorithms.

Farinholt et al. [90] investigated damage classification of drilled holes in composite plates using an electrical impedance technique [91] paired with fatigue loading. The classification error for this experiment ranged from 6.6% using a random forest algorithm to 83.2% for a quadratic discriminant analysis. The second experiment in this work classified embedded delaminations under fatigue tested 25.4-millimeter wide specimen. The ML approach correctly identified 85.7% of the six damage mechanisms investigated. Misclassification occurred for the extreme delamination widths (6.35 and 19.05 millimeters) due to the size of the embedded patch relative to the width of the test specimen.

Tibaduiza et al. [92] investigated a composite sandwich structure and plate with various applied damages. Preprocessing tools generated "scores", values to be used as inputs, for the 20 ML algorithms implemented. A novel data processing method was proposed involving a nonlinear analysis. Tibaduiza et al. employed nonlinear components as they believed that nonlinear values were capable of representing data about a structural state in less information than traditional methods.

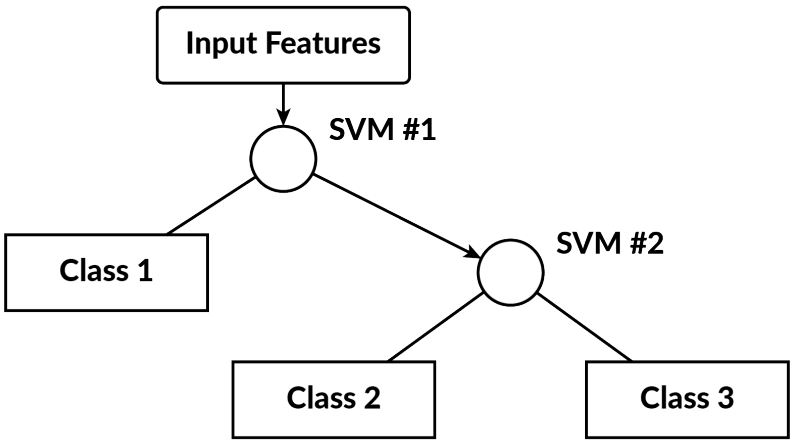


Figure 2.18: "One to others" approach using SVMs for problems involving more than two classes. [94]

The results confirmed this proposition as the fine and weighted k-NN algorithms correctly predicted the state of the CFRP sandwich structures 348 out of 350 times (99.4%). Further, the same types of ML algorithms had a prediction accuracy of 99.7% and 100%, respectively, in the case of the CFRP composite plate.

Doyle and Fernando [93] investigated impact damage classification in CFRP plates. A fiber-optic vibration sensor was mounted to the surface of each plate while impacted using a drop-weight testing machine or a steel ball bearing. The percentage of correct classifications was above 95% for all but one of the ANNs tested.

Pawar and Jung [94] applied ML to an analytical approach consisting of aeroelastic and damage accumulation models for classifying the damage level of helicopter rotor blades. Rather than classifying a specific type of damage such as a crack or delamination, the extent of damage was of interest. Three damage classes (negligible, moderate, and severe damage) were defined as given in Tab. 2.2. A "one to others" technique shown in Fig. 2.18 was implemented in the ML process such that two support vector machines (SVMs) were used in classifying the three classes. When all the hub forces and moments were inputted into the "one to others" SVM algorithm, the highest accuracy was obtained, whereas combinations of only two forces or two moments produced the lowest accuracy, especially at higher noise levels.

Extent The fourth damage description concerns the severity of the state of damage in a material. Answering this question provides information about the effect of damage mechanisms. Because

Table 2.2: Three-class damage classification proposed by Pawar and Jung [94] that relates the the damage class to the physical damage. A percentage of the used life (or the percentage of life remaining) can be correlated to the damage class to determine what actions are necessary to reduce the possibility of failure in-use.

Damage class	Physical damage level	Life consumption (%)	Prognostic Action
Negligible Damage	Up to matrix crack saturation (crack density ranges from 0 to the saturation crack density of 3.0)	0-15%	OK
Moderate Damage	Initial part of debonding/delamination zone (ESR 0.8-0.88)	15-56%	WATCH
Severe Damage	Last phase of debonding/delamination (ESR 0.92-0.94)	>56%	REMOVE

each damage mechanism grows differently, each type of damage impacts a structure differently. Since damage affects the microstructure of a material, the local property values around the damage location differ from those in a pristine, undamaged region. Regression algorithms have been used to quantify this damage detection question [95].

Gupta et al. [96] proposed a novel SHM technique utilizing mechanoluminescence to quantify the change in the elastic modulus of a polymer composite under cyclical loading. Organic mechanoluminescent phosphors were embedded into a polymer composite as sensing elements because they emit light under a mechanical load. The light intensity, I, emitted by the embedded particles along with the applied stress, σ, were measured and correlated using a multivariate linear regression model,

$$E = f(\sigma, I) = \theta_0 + \theta_1\sigma + \theta_2 I \tag{2.109}$$

where θ_0 was a bias term, and θ_1, θ_2 were weights associated with the stress and intensity, respectively. The elastic modulus was chosen as the parameter of interest because "structural deterioration can be observed by modulus relaxation as the component undergoes fatigue failure" [96]. After optimizing, the final multivariate linear model was found to be

$$E = 2.28 + (-0.071)\,\sigma + (0.037)\,I \tag{2.110}$$

The training and testing error was minimal at 1.25% and 1.32%, respectively, demonstrating the

viability of this approach for SHM. A drawback to this technique was the usage of mechanoluminescent particles; the host material must allow for light to pass through, i.e., the host material must be transparent to implement this method. A transparent polymer composite can allow for light emission, but polymers do not function well as structural materials [2].

Seo and Lee [97] presented an electrical resistance method for quantifying damage based on the degradation of the stiffness of a CFRP laminate. An electrical resistance damage parameter was defined and inputted into NNs to predict either the fatigue life or stiffness of the CFRP. Implementing electrical resistance for damage detection is feasible for CFRP laminates because of carbon fiber's conductivity (e.g., AS4 has a conductivity of approximately 6.54×10^4 siemens per meter [98]). However, the matrix material acts as an insulator hindering the usage of this approach for measurements in directions other than in the longitudinal direction of the fibers [99]. Todoroki [99], however, presented a finite element method for detecting delaminations in a CFRP using a set of five electrodes to determine its size and location.

Prognosis The final damage state description, prognosis, requires the most knowledge about the damage state to address. Information collected from the subsequent damage state descriptions can help forecast the health of a material or system. The prognostic work with CFRP laminates has focused on predicting failure before it transpires. There are several mechanisms for failure in a CFRP (or a generalized FRP) because of the structure and interaction between the fibers and matrix materials. Failure mechanisms exist for both constituent materials and must be considered as possible causes of failure. Matrix cracking and fiber breakage are predominantly investigated because loading conditions are typically distributed across a laminate cross-section. These mechanisms usually correspond to initial and total failure, respectively, of a FRP.

Bheemreddy et al. [100] investigated the use of machine learning with analytical modeling to predict a failure mechanism specific to FRPs: fiber pull-out. An ANN was trained to predict the coordinates of points on a load-displacement plot for a single fiber being pulled from an embedding host matrix. Results from an FEA and an analytical model were compared with the predicted load-displacement plots by ANNs. The predicted plot by the ANN was found to be a combination of the analytical and FEA curves due to their usage in training; therefore, the ANN curve fell between the other two models as shown in Fig. 2.19.

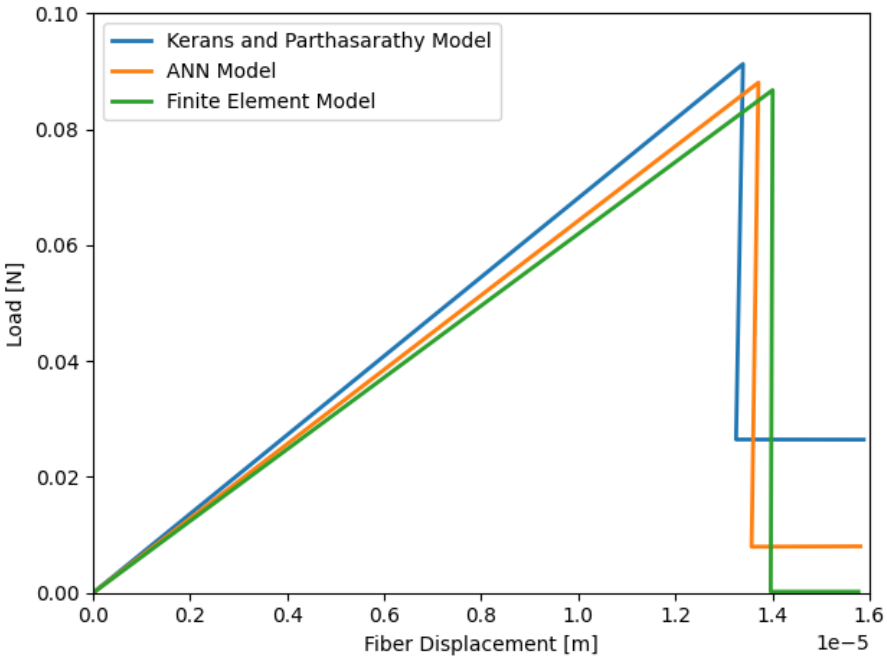


Figure 2.19: The use of data from analytical and FEA models in an ANN suggest the predicted load-displacement curve of the ANN is a combination of the two methods [100].

Fatigue The extensive amount of data necessary to fully describe fatigue suggests ML as a viable option for characterization [101]. Fatigue performance of a material is typically displayed on a plot of the stress amplitude and the number of cycles to failure (also called the fatigue life); other parameters for illustrating fatigue life have been implemented, such as acoustic emission data [102]. A material must be cyclically loaded to failure for various stress amplitudes to describe its fatigue behavior; therefore, a fatigue problem becomes a big data problem consisting of thousands of data points if using raw, complete datasets obtained from each experiment.

Aymerich and Serra [103] proposed using a NN to predict fatigue strength (the highest value of stress a material can sustain before failure) based on the number of cycles at failure and information about the stacking sequence of a CFRP. Neural networks were tested with various stacking sequences. The models better predicted a laminate with a trained stacking sequence than an unknown stacking sequence. Subsequent work focused on predicting the fatigue life—a useful approach for forecasting the life of a composite in a structure to determine how long it can be safely used without failure—of a composite rather than the fatigue strength.

El Kadi and Al-Assaf [104,105] investigated the use of neural networks to predict the fatigue

life of glass fiber composites. In [104], feedforward NNs (FNNs), modular NNs (MNNs), and radial basis function (RBF) networks were examined using data obtained from the cyclical loading of five unidirectional glass fiber composites. The fiber orientation angle, stress ratio, and maximum applied stress were inputted into each algorithm to predict fatigue life. The decomposition of a complex predictive model, such as that involved with fatigue life, into multiple NNs resulted in more accurate results than using a single NN for modeling. El Kadi and Al-Assaf [105] expanded their work in [104] by investigating the performance of a MNN using a strain energy term as an input. The strain energy term combined information about the composite and testing parameters:

$$\Delta W^{+} = \frac{1}{2}\overline{S}_{11}\Delta\sigma^{2}\left[\frac{1+R}{1-R}\right] \tag{2.111}$$

where $\overline{S}_{11}$ was an elastic term calculated from the stress-strain relationship and contained information about the fiber orientation angle, $\Delta\sigma$ was the applied stress range, and R was the stress ratio ($R = \sigma_{max}/\sigma_{min}$). Equation (2.111), however, was only applicable for uniaxial cyclical loading with positive stress ratio values due to applied assumptions.

Lee et al. [106] found that additional input parameters from those used in [104, 105] were useful when predicting fatigue life. Introducing the maximum and minimum stresses along with the tensile and compressive strengths of the material improved the accuracy of the ANNs. An attempt was made to predict the fatigue life of unknown, untrained materials using an ANN, but this investigation resulted in errors of over 100%. Figure 2.20 shows that this type of application of ML may not prove useful as the smallest RMSE was approximately 150%.

El Kadi and Al-Assaf [107] investigated the use of monotonic material properties to predict the fatigue life of unknown materials using neural networks. The longitudinal and transverse elastic moduli, longitudinal and transverse tensile strengths, fiber volume fraction, and fiber orientation angle were inputs based on previous findings in [104]. Polynomial combinations of the material properties were created to increase the number and variation of input parameters. The smallest RMSE obtained was 38.7% using a nonlinear combination of material properties: $S_{90}^{T} \times \theta \times (\log\sigma)^{3}$. Al-Assaf et al. [108] expanded the work of [107] by increasing the number of materials and loading conditions included in the training process. The RMSE for the unknown prediction models ranged from 6.1% to 40% when compared to experimental data. Ultimately, it was determined that the tested material's properties should fall within the range of material properties used in training the

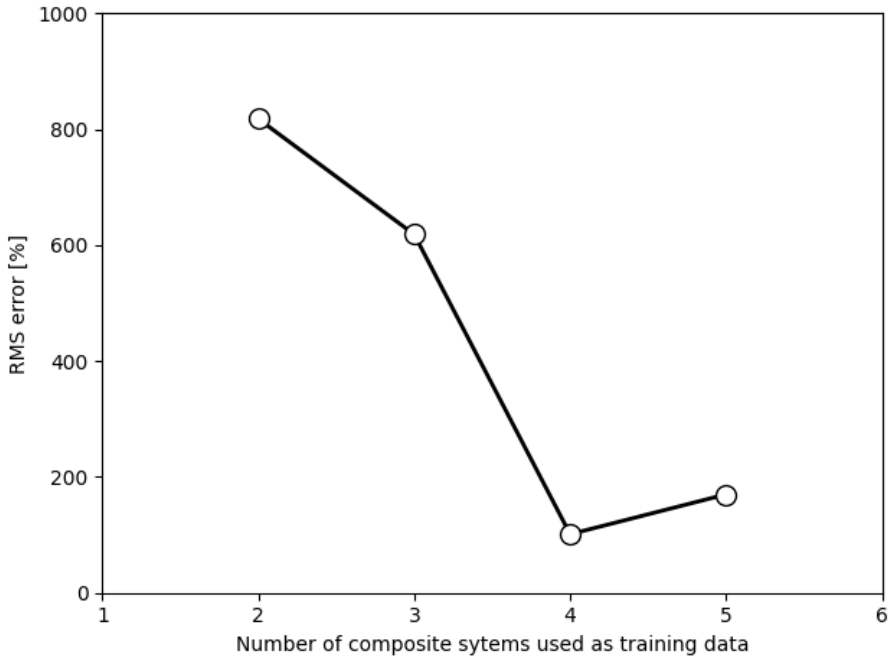


Figure 2.20: The RMSE in [106] decreased as more data (composite systems) were introduced to the ANN for fatigue life predicting. Four composite systems were found to produce the lowest RMSE.

ANN because of the high error obtained in [107, 108].

Failure Theoretical models for composite failure are based on the established and well-studied metallic failure theories. The response of each constituent material must be considered in composite failure theory. Additionally, the interactions of the constituents at their respective interfaces can be included in a theoretical model for completeness. Each constituent has the possibility of failing at different times in the life of a composite as displayed in Fig. 2.21. Composite failure theories are often empirical, utilizing physical information about the material and its microstructure to define parameters [109–111]. The complexity of these calculations can be numerically intensive. To reduce computational expenses, researchers have implemented ML algorithms for failure predictions because of their ability to model nonlinear relationships.

Because of the complexity of a multi-constituent composite material, anisotropic models are necessary to model a composite fully. Simplification of a composite to an orthotropic material is often implemented in failure predictions because of previously developed orthotropic models. Tsai-Wu [112] proposed a failure surface that utilized a six-dimensional stress field combined with

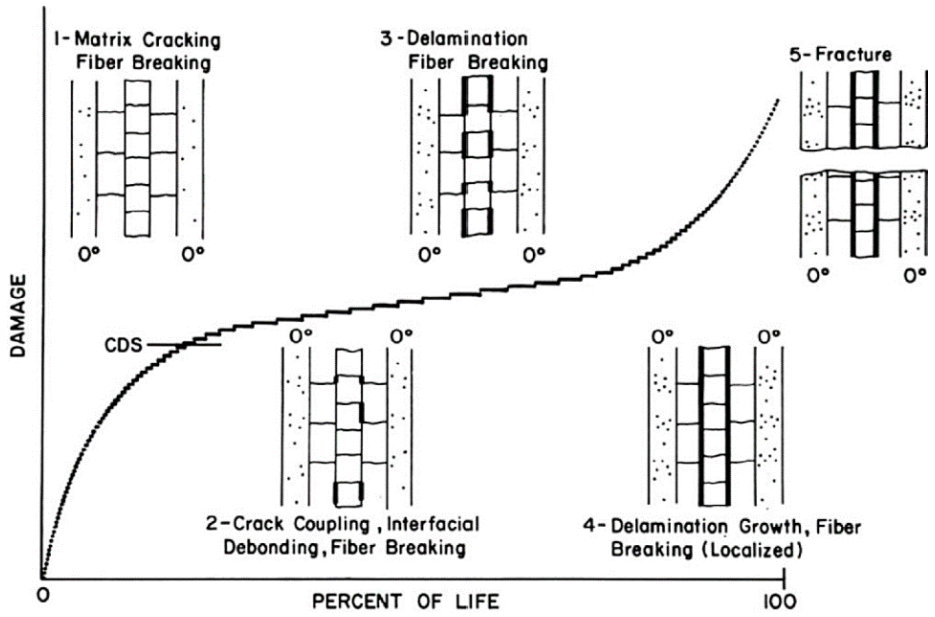


Figure 2.21: Over the lifetime of FRP composite under loading, damage mechanisms change and accumulate to form different mechanisms like cracks and delaminations. [102]

anisotropic strength parameters,

$$F_i\sigma_i + F_{ij}\sigma_i\sigma_j = 1, \qquad i,j = 1,\ldots,6 \tag{2.112}$$

where F_i and F_{ij} are strength tensors dependent on the composite layup. Assumptions have been developed with Eq. (2.112) to reduce the symmetry of the model to orthotropic and transversely isotropic natures. Jones [1] presented the relationships of the strength tensors with monotonic material properties for a plane stress condition of a transversely isotropic material:

$$F_1\sigma_{11} + F_2\sigma_{22} + F_{11}\sigma_{11}^2 + F_{22}\sigma_{22}^2 + F_{66}\sigma_{12}^2 + 2F_{12}\sigma_{11}\sigma_{22} = 1 \tag{2.113}$$

where F_i and F_{ij} are related to the monotonic strengths as

$$\begin{aligned} F_1 &= \frac{1}{X_T} - \frac{1}{X_C} \\ F_2 &= \frac{1}{Y_T} - \frac{1}{Y_C} \\ F_{11} &= \frac{1}{X_T X_C} \\ F_{22} &= \frac{1}{Y_T Y_C} \\ F_{66} &= \frac{1}{S^2} \end{aligned} \tag{2.114}$$

where X_T and X_C are the longitudinal tensile and compressive strengths in the fiber direction, Y_T and Y_C are the transverse tensile and compressive strengths, and S is the in-plane shear strength. Two forms of the Tsai-Wu criterion were utilized in [113–117]: the form presented in Eq. (2.113) and an optimized Tsai-Wu criterion. The optimized criterion sought to minimize the error between the experimental data and the analytical failure surface by optimizing the strength tensor coefficients, i.e., the monotonic material properties were not used to define the strength tensors. The Tsai-Wu and the optimized Tsai-Wu failure criterion had ellipsoidal failure envelopes, or surfaces, as illustrated in Fig. 2.22.

Labossière and Turkkan [113] first suggested the use of a NN for the prediction of the failure envelope using a softwood, fibrous composite material. Lee et al. [114] investigated the failure of CFRPs under biaxial loading using an ANN. Bhuiyan et al. [117] confirmed the Tsai-Wu, optimized Tsai-Wu, and NN failure envelopes obtained in [113,114] using biaxial loading experiments of CFRP plates. The CFRP plates were subjected to transverse and axial loads to induce failure. The Tsai-Wu criterion predicted a general symmetric ellipsoid failure surface independent of experimental data illustrated in Fig. 2.22. Because the optimized Tsai-Wu criterion considered experimental data in its failure envelope prediction, the failure envelope more closely resembled the experimental data than the traditional Tsai-Wu criterion.

Lopes et al. [115] proposed the use of a radial basis network with randomized loading conditions (axial, biaxial, in-plane shear) as inputs for the prediction of the Tsai-Wu factor, i.e., the sum of the left-hand side of Eq. (2.113). Elhewy et al. [107] investigated the architecture and types of functions used in an ANN to reduce computational costs over other methods such as finite elements. Both [107] and [115] demonstrated that ML algorithms could be used to reduce computational

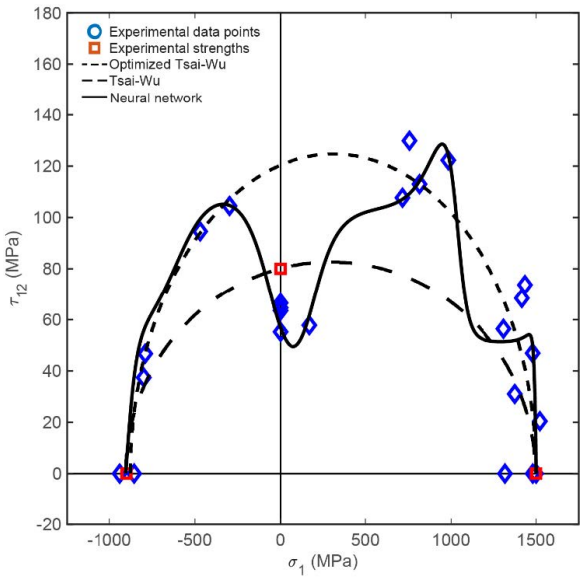


Figure 2.22: The NN predicted failure envelope minimizes the error between the actual (experimental) values and its predicted values, thus producing an unsymmetric, non-uniform failure envelope. [117]

expenses given sufficient data.

Şerban [116] proposed the implementation of classification and regression algorithms for predicting failure probability. A two-class problem was defined to include a "fail" and a "not fail" class for a set of $86,953$ simulated laminates with random ply orientations. The division of classes was determined by comparing a predicted failure criterion with a critical failure factor. Seven algorithms were explored, and the minimum and maximum classification errors were 1.33% (k-NN) and 9.08% (quadratic discriminant analysis), respectively. The regression algorithms predicted the value of the failure factor, and the results were compared against a value calculated using FEA models. Şerban projected confidence in both the classification and regression methods based on the error calculated between the ML and FEA predictions.

Chapter 3

Analytical Modeling

3.1 Motivation for Implementing a Model

Experimentation can be financially expensive to complete due to a lack of available resources; some experiments are not repeatable. The creation of Terfenol-D embedded laminates requires special care to reduce oxidation of the particles during fabrication [118, 119]. The examined NDE method utilizes a testing apparatus consisting of an actuating-sensing coil connected to a power amplifier and data acquisition system. A lack of availability to the testing equipment spurred an investigation into the implementation of an analytical model to predict the presence of a delamination within a CFRP using magnetostriction. Slight manipulations in testing parameters are also easier to make in modeling methods than experimentation so that parametric studies can examine the effect of multiple parameters on the results of the model.

Krishnamurthy et al. [53] proposed a model derived from the Euler-Bernoulli beam theory to calculate the induced stress in the embedded magnetostrictive material. With the induced stress found, the constitutive relationship correlated the magnetic susceptibility with the stress state to predict the induced sensing voltage. The model necessitated the characterization of Terfenol-D to relate the material's nonlinear stress-strain and strain-magnetic field intensity relationships. Implementing an analytical model has been previously demonstrated as a method for acquiring data for machine learning. This analytical model investigation aimed to generate a large dataset of induced sensing voltages for various stacking sequences to resemble experimental conditions.

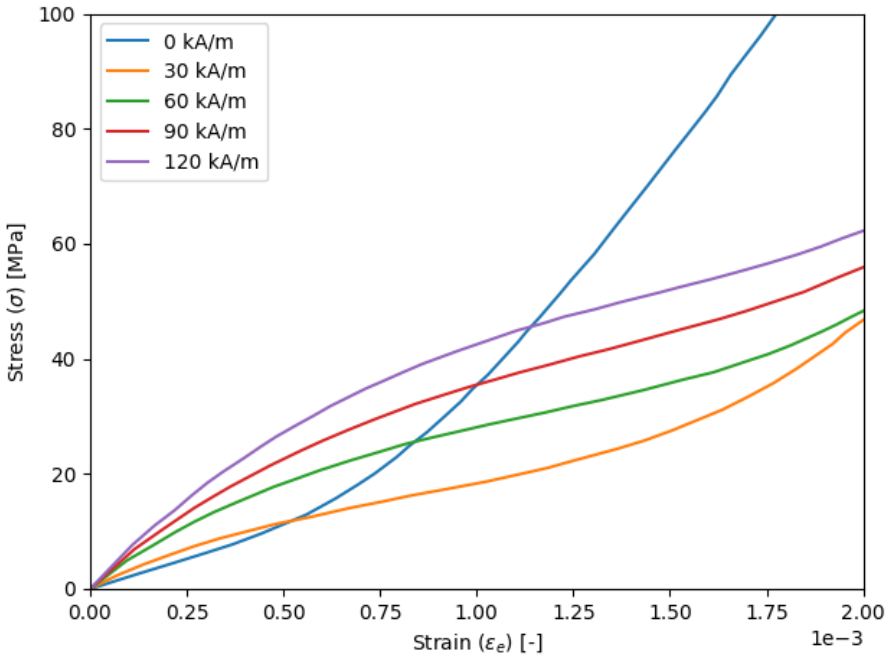


Figure 3.1: Extracted data from a stress-strain plot for Terfenol-D plot given by Butler [120]. Five bias magnetic fields intensities were experimentally applied to the material.

3.2 Characterization of Terfenol-D

3.2.1 Stress-Strain Relationship

Stress-strain data for Terfenol-D was presented by Butler [120] using experiments with varying bias magnetic field intensities, H_0. The bias magnetic field aligns the magnetostrictive material's dipoles along a specific direction during fabrication to increase the effect of magnetostriction. A reproduction of Butler's experimental stress-strain plot is shown in Fig. 3.1 where the curves were extracted using the GRABIT function [121] in MATLAB. Using the GRABIT function, data points were interactively selected for each curve from the original plot. These data points were then exported and plotted to create the curves using the Python 3.8.5 script in Appendix D.1.1.

3.2.1.1 Ramberg-Osgood Model

Ramberg and Osgood [122] previously showed that a nonlinear stress-strain curve could be defined by

$$\epsilon = \frac{\sigma}{E} + K\left(\frac{\sigma}{E}\right)^n \tag{3.1}$$

Table 3.1: Ramberg-Osgood constants for stress-strain curve [53].

Constant	Value
α_e	52.287×10^{-12}
β_e	$-1.218 \times 10^{-6(n+1)}$
n	1.818

where K and n are constants. Their proposed relationship incorporated both elastic and plastic strain terms to produce a single stress-strain curve. Krishnamurthy [53] presented a modified Ramberg-Osgood expression to quantify the stress-strain behavior of Terfenol-D:

$$\epsilon_e = \alpha_e \sigma + \beta_e \sigma^n \tag{3.2}$$

where α_e, β_e, and n are constants. The constants presented in [53] for Eq. (3.2) are listed in Tab. 3.1. Though not explicitly stated, the following relationships were drawn between Eq. (3.1) and (3.2):

$$\alpha_e = \frac{1}{E} \tag{3.3}$$

$$\beta_e = \frac{K}{E^n} \tag{3.4}$$

The proposed Ramberg-Osgood expression for the stress-strain relationship was then expressible with the constants substituted as:

$$\epsilon_e = \left(52.287 \times 10^{-12}\right) \sigma + \left(-1.218 \times 10^{-6(1.818+1)}\right) \sigma^{1.818} \tag{3.5}$$

$$= \left(52.287 \times 10^{-12}\right) \sigma + \left(-1.218 \times 10^{-16.908}\right) \sigma^{1.818} \tag{3.6}$$

3.2.2 Compliance Term

The compliance term was defined as the inverse of the slope of the stress-strain curve:

$$s = \frac{d\epsilon_e}{d\sigma} \tag{3.7}$$

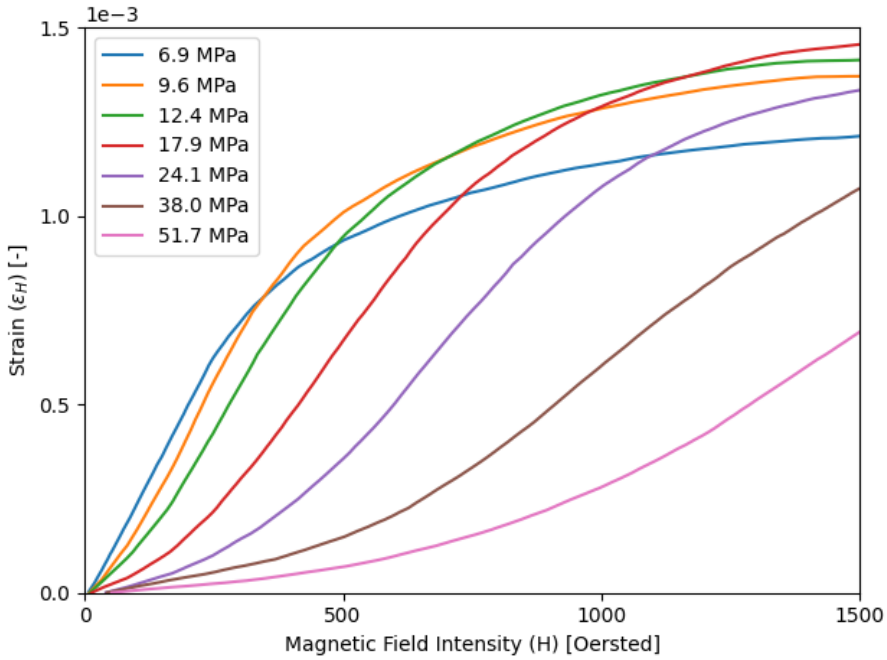


Figure 3.2: Extracted data from a strain-magnetic field intensity plot for Terfenol-D plot given by Butler [120] for seven prestress levels.

For the model proposed in [53], the compliance term was symbolically expressed as

$$s = \frac{d}{d\sigma}\left(\alpha_e \sigma + \beta_e \sigma^n\right) \tag{3.8}$$

$$= \alpha_e + \beta_e n \sigma^{n-1} \tag{3.9}$$

where the constants are listed in Tab. 3.1. The compliance term was assumed to be nonlinear because of its derivation utilizing the nonlinear stress-strain relationship given by Eq. (3.2).

3.2.3 Strain-Magnetic Field Intensity Relationship

Using the GRABIT function [121] in MATLAB, the strain-magnetic field intensity plot presented by Butler [120] was recreated in Fig. 3.2. The strain-magnetic field intensity relationship was experimentally dependent on the prestress, σ_0, applied to the Terfenol-D material. This prestress can develop during fabrication due to thermal contraction or applied external loads [17].

Table 3.2: α_{ij} constants found in [53] for respective prestress levels.

σ_j [MPa]	α_{0j}	α_{1j}	α_{2j}	α_{3j}	α_{4j}
6.9	-50.2987	3.7301	-4.7000e-03	2.8406e-06	-6.4174e-10
9.6	-81.3171	3.2017	-2.3000e-03	4.0316e-07	1.1572e-10
12.4	-58.7017	1.9530	9.7200e-04	-2.4375e-06	9.0950e-10
17.9	-2.1391	-0.0931	5.1000e-03	-5.2425e-06	1.5458e-09
24.1	22.0131	-0.5268	3.6000e-03	-2.5378e-06	4.9722e-10
38.0	-0.4968	0.2066	-3.7006e-04	1.3335e-06	-5.7574e-10
51.7	0.1733	0.0743	3.4900e-05	2.1599e-07	-4.4299e-11

3.2.3.1 Proposed Quartic Polynomial Relationship

Krishnamurthy et al. [53] proposed a fourth degree polynomial to quantify and express the nonlinear experimental data given in Fig. 3.2. The relationship correlated the intensity of an applied magnetic field, H, with the strain induced by the magnetic field, ϵ_H:

$$\epsilon_H = \alpha_0 + \alpha_1 H + \alpha_2 H^2 + \alpha_3 H^3 + \alpha_4 H^4 = \alpha_0 + \sum_{i=1}^{4} \alpha_i H^i = \sum_{i=0}^{4} \alpha_i H^i \tag{3.10}$$

where the α_i values in Tab. 3.2 were determined for each of the seven experimental prestress values.

3.2.4 Piezomagnetic Constant

The piezomagnetic constant was defined in [53] as the slope of the strain-magnetic field intensity curve:

$$d_c = \frac{d\epsilon_H}{dH} \tag{3.11}$$

For the quartic polynomial relationship given in Eq. (3.10), the piezomagnetic coefficient was symbolically expressible as

$$d_c = \frac{d}{dH}\left(\alpha_0 + \alpha_1 H + \alpha_2 H^2 + \alpha_3 H^3 + \alpha_4 H^4\right) \tag{3.12}$$

$$= \alpha_1 + 2\alpha_2 H + 3\alpha_3 H^2 + 4\alpha_4 H^3 \tag{3.13}$$

where the α_i constants in Tab. 3.2 for the prestress values still applied. An alternative to Eq. (3.13) was given as

$$d_c = a_1 + a_2 H + a_3 H^2 + a_4 H^3 \tag{3.14}$$

where $a_1 = \alpha_1$, $a_2 = 2\alpha_2$, $a_3 = 3\alpha_3$, and $a_4 = 4\alpha_4$.

3.2.5 Permeability

Butler [120] presented experimental data for the nonlinear relationship of the magnetic flux, B, and magnetic field intensity, H, as a function of the prestress on the Terfenol-D. The magnetic permeability was defined as

$$\mu = \frac{dB}{dH} \tag{3.15}$$

which is the slope of the magnetic flux-magnetic field intensity curve. Krishnamurthy et al. [53] suggested that a linear region existed for an interval around $H = 0$ Oersted. Outside that interval, the relationship was no longer linear as the slope was not constant. A piecewise function was proposed to model the magnetic flux-magnetic field intensity curves:

$$B = \begin{cases} b_0 H, & -H_1 \leq H \leq H_1 \\ b_0 H_1 + b_1 \left(H - H_1\right), & H > H_1 \\ -b_0 |H| + b_1 \left(H - |H_1|\right), & H < -H_1 \end{cases} \tag{3.16}$$

Likewise, a piecewise function was defined to model the permeability:

$$\mu = \begin{cases} b_0, & -H_1 \leq H \leq H_1 \\ b_1, & H > H_1, \quad H < -H_1 \end{cases} \tag{3.17}$$

The constants for the piecewise function were dependent on the prestress, as shown in Tab. 3.3.

Table 3.3: Constants for the piecewise permeability relationship given in [53].

σ_j [MPa]	b_0 [Gauss/Oersted]	b_1 [Gauss/Oersted]	H_1 [Oersted]
0.00	60.00	3.06	0.00
3.45	38.44	3.18	197.00
10.35	14.88	3.26	459.00
13.80	12.42	3.35	656.00
20.70	8.77	4.30	767.00

3.3 Previously Proposed Model

Krishnamurthy et al. [53] presented a three-step process for modeling the sensing capability of embedded magnetostrictive material in a composite laminate. The proposed model assumed the laminate to consist of unidirectional, fiber-reinforced laminae, and the magnetostrictive lamina was bounded between two of the fiber-reinforced laminae. Simplification of the fiber-reinforced laminae was made through an orthotropic assumption. The magnetostrictive lamina discussed consisted of Terfenol-D particles embedded in a resin matrix host; however, the influence of the resin system was ignored or considered negligible because the bulk properties of Terfenol-D represented the properties of the sensing lamina. The only source of stress applied to the laminate in [53] came from an actuation coil with a current in the form

$$I = I_0 \sin \Omega t \tag{3.18}$$

The associated magnetic field intensity for a coil was found using principles of magnetism:

$$H = H_0 \sin \Omega t \tag{3.19}$$

No mechanical loads were considered to focus on the effects of the actuating coil in the sensing process. The influence of axial and bending stresses on the performance of the sensing lamina was examined. The effect of axial stress was determined greater than that of bending stress due to the induced elongation constraint by the host matrix on the Terfenol-D particles. The magnitude of the bending stress was related to the symmetry of the laminate. If the magnetostrictive lamina was embedded along the midplane of a symmetric laminate, the bending stress was negligible as symmetric laminates display negligible bending according to the classical lamination theory. Bending moments, however, are necessary to consider for a laminate with an asymmetrically located magne-

tostrictive lamina. The problem was deemed nonlinear because of the nonlinearity associated with the Terfenol-D particles.

Krishnamurthy et al. formulated their analytical model from the principles of the Euler-Bernoulli theory. The elastic response of a beam was first applied to the composite laminate, and additional terms were introduced to expand the theory to incorporate the response of the magnetostrictive material. The following derivation was presented in [53]. The Euler-Bernoulli theory defines a linear elastic displacement field of

$$u = U(x) - yV(x)_{,x} \tag{3.20}$$

$$v = V(x) \tag{3.21}$$

$$w = 0 \tag{3.22}$$

The only non-zero strain term considered in the derivation in [53], ϵ_x, was given as

$$\epsilon_x = \epsilon - y\gamma = \frac{dU}{dx} - y\frac{d^2V}{dx^2} \tag{3.23}$$

The stress in the x-direction is found according to the elasticity concept of Hooke's law as

$$\sigma_x = E_i \epsilon_x \tag{3.24}$$

where the subscript i denoted the layer of the lamina. The x-direction corresponded to the longitudinal direction of the fibers, i.e., the axial direction of the fibers. For the magnetostrictive layer, the elastic modulus was related to the compliance term such that when $E_i = E_m$, the elastic modulus equaled the inverse of the derived compliance term, $E_m = 1/s$. Unlike the elastic modulus of the fiber-reinforced laminae, the elastic modulus of the magnetostrictive lamina depends on the magnetic field intensity applied to the particulate composite during fabrication [123]. The force and moment resultants of the laminate were found by integrating the stress component

$$N = \int \sigma_x dy \tag{3.25}$$

$$M = \int \sigma_x dy \tag{3.26}$$

which was the same methodology given in Section 2.3.4 for the classical lamination theory. In matrix form, the resultants were related to the strain and curvature as

$$\begin{pmatrix} N \\ M \end{pmatrix} = \begin{bmatrix} A_c & B_c \\ B_c & D_c \end{bmatrix} \begin{pmatrix} \epsilon \\ \kappa \end{pmatrix} \tag{3.27}$$

where the A_c, B_c, and D_c were elastic terms defined as

$$(A_c, B_c, D_c) = \sum_{i=1}^{n} \int_{y_i}^{y_{i+1}} E_i \left(1, y, y^2\right) dy \tag{3.28}$$

Note, this formulation of the **ABD** matrix terms differed from that given in the CLT by Eqs. (2.83) – (2.85).

3.3.1 Stress due to Force in x-axis

For a symmetric laminate, the number of plies above and below the midplane is equal. The magnetostrictive sensing lamina must be embedded along the midplane of a laminate if seeking to preserve symmetry. Three force resultants existed in the laminate illustrated in Fig. 3.3 due to the applied magnetic field. Unconstrained, the magnetostrictive lamina would ideally expand under the externally applied magnetic field. However, the surrounding laminae (regions 1 and 2) constrained the magnetostrictive lamina and constricted any deformation caused by magnetostriction. The host matrix material would further constrain the magnetostrictive particles, but [53] did not consider this effect.

The compressive force resultant induced in the magnetostrictive layer equaled

$$N_m = \sigma h_m \tag{3.29}$$

whereas the tensile force resultants in the surrounding regions equaled

$$N_1 = A_1 \epsilon, \qquad A_1 = \sum_{i=1}^{p_1} \int_{y_i}^{y_{i+1}} E_i dy \tag{3.30}$$

$$N_2 = A_2 \epsilon, \qquad A_2 = \sum_{i=1}^{p_2} \int_{y_i}^{y_{i+1}} E_i dy \tag{3.31}$$

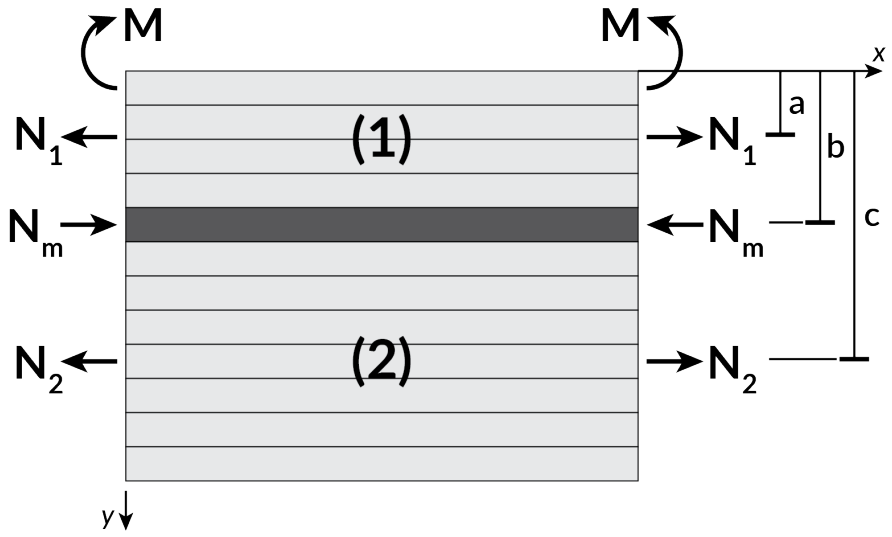


Figure 3.3: Force and moment resultants for a asymmetric composite laminate with an embedded magnetostrictive lamina. [53]

Static equilibrium along the x-direction required

$$N_m = N_1 + N_2 \tag{3.32}$$

Equation (3.32) mathematically stated that the surrounding laminae constrained the magnetostrictive material since the force resultants had to equal to ensure no dynamic effects were present. Substituting the force resultants of the surrounding regions, Eqs. (3.30) and (3.31), into the equation of equilibrium, Eq. (3.32), yielded the overall strain of the laminate in terms of the force resultant for the magnetostrictive lamina

$$\epsilon = \frac{N_m}{A_1 + A_2} \tag{3.33}$$

The overall strain ϵ was equal for all the laminae because perfect bonding between layers was assumed; thus, the strain in the magnetostrictive material equaled the strain in the surrounding regions 1 and 2. The strain in the magnetostrictive layer was determined through the integration of the nonlinear, differential form of Eq. (2.99):

$$d\epsilon = ds^H \sigma + d_c dH \tag{3.34}$$

$$\int_0^\epsilon d\epsilon = \int_0^\sigma s d\sigma + \int_0^H d_c dH \tag{3.35}$$

Eq. (3.35) was rewritten in terms of individual strain terms as

$$\epsilon = \epsilon_e + \epsilon_H \tag{3.36}$$

where ϵ_e was the elastic strain determined from the stress-strain curve, and ϵ_H was the strain relative to the applied magnetic field intensity. Both of these terms required the consideration of external conditions applied to the lamina during fabrication. Substituting the formulations proposed by [53] for the strain terms, Eqs. (3.2) and (3.10), the strain in the magnetostrictive material was symbolically represented by

$$\epsilon = (\alpha_e \sigma + \beta_e \sigma^n) + \sum_{i=0}^{4} \alpha_i H^i \tag{3.37}$$

Note that the lower limit of summation in Eq. (3.37) differed from that in [53] which was set to $i = 1$. Combining Eq. (3.33) with (3.37) resulted in an expression relating the total strain of the laminate (equal to that of the magnetostrictive lamina) to the force resultant of the magnetostrictive lamina and the elastic properties of regions 1 and 2:

$$\frac{N_m}{A_1 + A_2} = (\alpha_e \sigma + \beta_e \sigma^n) + \sum_{i=0}^{4} \alpha_i H^i \tag{3.38}$$

The induced stress in the magnetostrictive material was related to the force resultant for the magnetostrictive lamina by Eq. (3.29). The two nonlinear strain terms were related to the stress in the magnetostrictive material by substituting Eq. (3.29) into (3.38):

$$\frac{\sigma h_m}{A_1 + A_2} - (\alpha_e \sigma + \beta_e \sigma^n) = \sum_{i=0}^{4} \alpha_i (\sigma) H^i \tag{3.39}$$

Krishnamurthy et al. [53] suggested that an iterative scheme be used to solve Eq. (3.39) for the induced stress in the magnetostrictive material because of the nonlinear effects involved with the material properties of the magnetostrictive material. The iterative expression to find the induced stress for a given magnetic field intensity H was given as

$$\frac{\sigma_{j+1} h_m}{A_1 + A_2} - \left(\alpha_e \sigma_{j+1} + \beta_e \sigma_{j+1}^n\right) = \sum_{i=0}^{4} \alpha_i (\sigma_j) H^i \tag{3.40}$$

3.3.2 The Effect of the Induced Bending Moment

When the magnetostrictive layer is located asymmetrically, a moment, M, is induced under loading and should be considered to capture the behavior of the laminate fully. The equation of equilibrium along the x-direction for this situation per Fig. 3.3 was

$$M = N_1 a + N_2 c - N_m b \tag{3.41}$$

where a, b, and c were the distances from the top of the laminate to the neutral axes of the respective regions. Per the $\mathbf{A_c B_c D_c}$ matrix relationship, Eq. (3.27), the moment resultant and curvature were related by

$$\gamma = \frac{M - B_c \epsilon}{D_c} \tag{3.42}$$

The force resultant matrix expression was adjusted to account for bending,

$$N_m = A' \epsilon' + B_c \gamma \tag{3.43}$$

where ϵ' was an axial strain term that accounted for bending, and $A' = A_1 + A_2 + E_m h_m$ with $E_m = 1/s$. The elastic modulus of the magnetostrictive material E_m, though nonlinear, was assumed to be linear in consideration of the bending effects because the magnitude of the effect of the moment was believed to be small compared to the effect of the axial stress. Equation (3.43) was rewritten in the form of Eq. (3.33):

$$\epsilon' = \frac{N_m - B_c \gamma}{A'} \tag{3.44}$$

Substituting Eqs. (3.37) and (3.29) into (3.44) yielded an expression similar to Eq. (3.39) that included the effect of the moment:

$$\frac{\sigma h_m - B_c \gamma}{A'} - (\alpha_e \sigma + \beta_e \sigma^n) = \sum_{i=0}^{4} \alpha_i (\sigma) H^i \tag{3.45}$$

The stress was iteratively solved for a given magnetic field intensity H by

$$\frac{\sigma_{j+1} h_m - B_c \gamma}{A'} - \left(\alpha_e \sigma_{j+1} + \beta_e \sigma_{j+1}^n\right) = \sum_{i=0}^{4} \alpha_i (\sigma_j) H^i \tag{3.46}$$

The differences in Eqs. (3.46) and (3.40) are the inclusion of the coupling stiffness coefficient B_c, midplane curvature κ, and a modified extensional stiffness coefficient A'. While the curvature of the laminate induced stress on the magnetostrictive material, this effect was hypothesized as small in magnitude since the calculation of the B_c term involved squaring the thickness of the laminae.

3.3.3 Induced Sensing Voltage

The stresses calculated using Eqs. (3.40) and (3.46) was assumed to be instantaneous for a time-dependent magnetic field given by Eq. (3.19). Inertial effects from the magnetic field caused by time dependency were assumed negligible because of the laminate structure. The magnetostrictive lamina was considered "very thin," and the strength of the surrounding regions, i.e., the CFRP laminae, was much greater than that of the Terfenol-D bulk material [53]. Therefore, no movement was assumed to occur for the magnetostrictive lamina as it was constrained in place sandwiched between the other two regions (regions 1 and 2). The differential form of Eq. (2.100),

$$dB = d_c d\sigma + \mu dH \tag{3.47}$$

related the incremental change in stress and magnetic field intensity to the incremental change of the magnetic flux. Integrating this expression using the parameters obtained from the iterative stress scheme yielded the total magnetic flux density:

$$B = \int_0^\sigma d_c d\sigma + \int_0^H \mu dH \tag{3.48}$$

$$\begin{aligned} &= \left[\int_0^\sigma a_1(\sigma)\, d\sigma + H\int_0^\sigma a_2(\sigma)\, d\sigma + H^2\int_0^\sigma a_3(\sigma)\, d\sigma + H^3\int_0^\sigma a_4(\sigma)\, d\sigma\right] \\ &\qquad + b_0(\sigma)\, H + tr\left[b_0(\sigma)(H_1 - H) + b_1(\sigma)(H - H_1)\right] \end{aligned} \tag{3.49}$$

where tr is the tracer function defined by

$$tr = \begin{cases} 0, & H < H_1 \\ 1, & H > H_1 \end{cases}$$

The a_i terms in Eq. (3.49) were defined in Eq. (3.14). All of the constants included in the magnetic flux expression were dependent on the prestress experienced by the material. The integration of the

a_i terms was found using either an analytical or numerical approach, but no further discussion was made regarding this integration in [53]. The induced voltage in the sensing coil was determined by

$$v(t) = -nA\frac{dB}{dt} \tag{3.50}$$

where n is the number of turns in the sensing coil, and A is the effective magnetic flux flow cross-sectional area. The magnetic flux found in Eq. (3.49) was substituted into Eq. (3.50). The derivative of B was feasible assuming a time-dependent magnetic field.

3.4 Examination of the Nonlinear Constitutive Relationships

3.4.1 Stress-Strain Relationship

The Ramberg-Osgood model proposed in Eq. (3.6) was plotted against the experimental data presented by Butler [120]. The model was only applicable for the case of a bias magnetic field intensity of 0 kA/m. The proposed Ramberg-Osgood approximation shown in Fig. 3.4 closely followed the experimental data for $0 \leq \epsilon \leq 0.7e-3$ and then began to diverge from the experimental data above a strain of $0.7e-3$. The results of the Krishnamurthy Ramberg-Osgood model visually appeared to differ from that presented in [53]. The plot found in the literature appeared to have a smaller difference between the proposed Ramberg-Osgood model and the presented experimental data in the region $\epsilon > 700e-6$. To investigate this discrepancy, the experimental data for the stress-strain relationship presented in [53] was extracted using the MATLAB GRABIT function [121] for comparison against the data presented by Butler [120]. Visually, the difference between the extracted experimental data from [120] and [53] was negligible until a strain of $0.8e-3$ in which case the [53] set fell below that of the [120] as illustrated in Fig. 3.4. No discussion of additional stress-strain experiments was included in [53]; therefore, the difference in the [53] values was associated with the extraction of the experimental data curve from its original source in [120]. The error statistics in Tab. 3.4 illustrate the difference in the experimental data given in literature by [120] and [53] from the examination of their differences from the proposed Ramberg-Osgood model. The proposed Ramberg-Osgood model followed closer to the experimental data presented by [53] than that by [120]

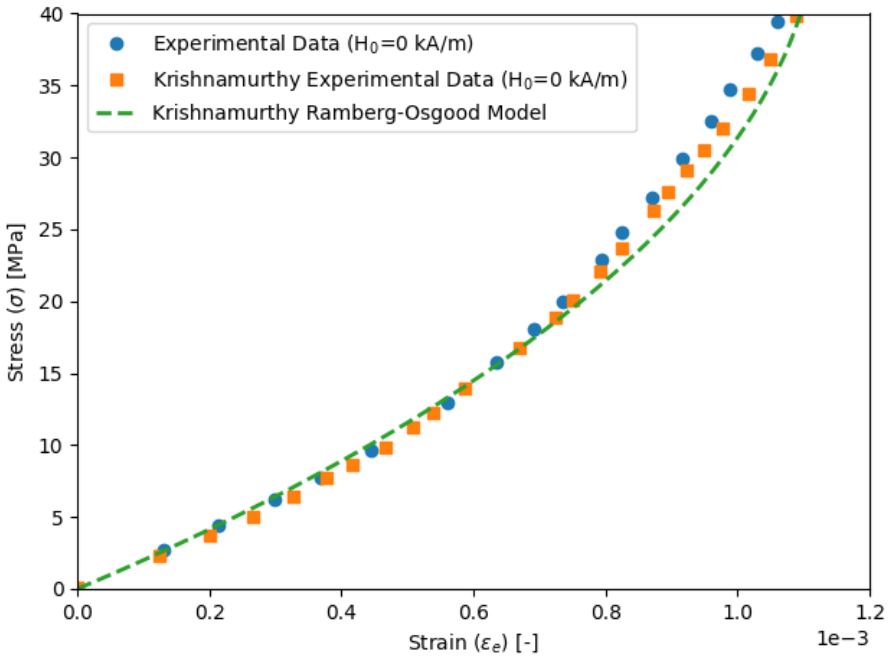


Figure 3.4: The Ramberg-Osgood model proposed in [53] plotted against the experimental data given by [120].

Table 3.4: Error statistics for the fit of the proposed Ramberg-Osgood model against experimental data found in literature.

Experimental Dataset	Max Error	Mean Squared Error	Root Mean Squared Error
Butler [120]	6.082e-05	1.282e-09	3.581e-05
Krishnamurthy [53]	1.088e-03	5.038e-07	7.098e-04

based on the mean squared error (MSE) calculated as [124]:

$$MSE\left(y,\hat{y}\right)=\frac{1}{n_{samples}}\sum_{i=0}^{n_{samples}-1}\left(y_i-\hat{y}_i\right)^2 \tag{3.51}$$

At values of strain less than approximately $0.7e-3$, the proposed Ramberg-Osgood model followed closer to the Butler data. Above this point, when the two experimental datasets began the noticeably differ, the Ramberg-Osgood model fell closer to the Krishnamurthy et al. [53] data.

When the range of stress was increased from $0 \leq \sigma \leq 40$ MPa to $0 \leq \sigma \leq 100$ MPa, Fig. 3.5 shows that the model given by [53] diverged from the experimental data. The Krishnamurthy Ramberg-Osgood model appeared parabolic in behavior under the extended stress domain. This

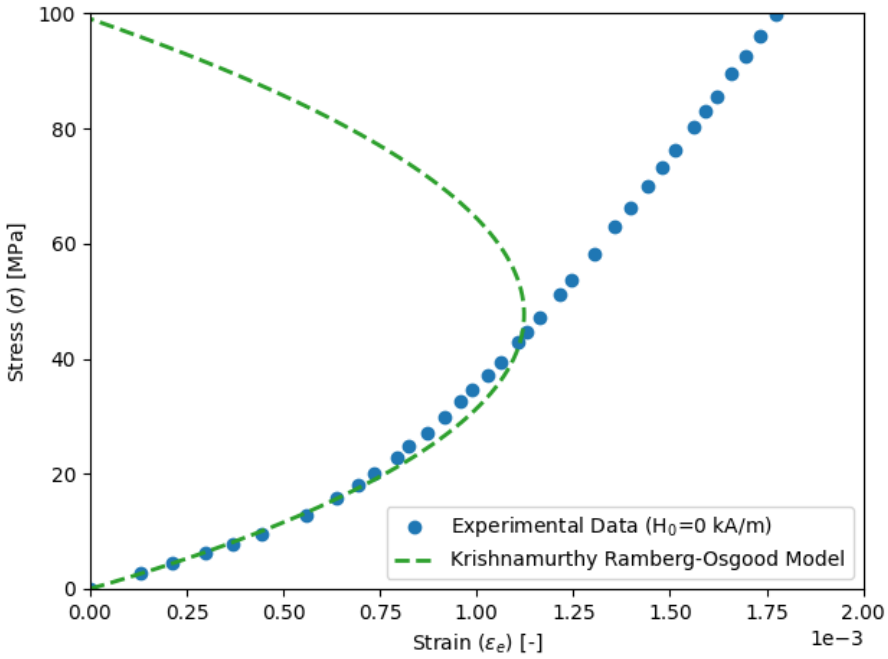


Figure 3.5: When the stress was increased above that given consideration in [53], the proposed Ramberg-Osgood model diverged from the experimental data given by [120].

behavior was attributed to the value of n, the exponent of the power term in Eq. (3.2). Krishnamurthy et al. [53] proposed a value of $n = 1.818$, indicating that the proposed Ramberg-Osgood model includes an almost quadratic term. The divergence of the proposed model raised applicability concerns as [53] only included a subset range of stresses, $0 \leq \sigma \leq 40$ MPa. Assuming these constants to be true beyond $\sigma = 40$ MPa would lead to a significant deviation in results based on the behavior presented in Fig. 3.5.

The collapse of the proposed Ramberg-Osgood model for the stress range of $0-100$ MPa led to an investigation to develop a new model to represent the stress-strain relationship. Furthermore, a generalized model was desired for application to all the stress-strain curves presented in [120]. The stress-strain curves in Fig. 3.1 are nonlinear curves with constantly changing curvatures. A Fourier series was selected to model the constant undulations in the curves because it combines trigonometric functions. The general form of a Fourier series is

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos(nx) + \sum_{n=1}^{\infty} a_n \sin(nx) \tag{3.52}$$

The Fourier series expression investigated in this work differed from Eq. (3.52) by combining the a_0 coefficient with the one-half factor:

$$f(x) = a_0 + \sum_{n=1}^{p} a_n \cos(n\omega x) + \sum_{n=1}^{p} a_n \sin(n\omega x) \tag{3.53}$$

where p is the number of terms included in the series. Additionally, the period of the trigonometric functions, ω, was determined in the investigation. A generalized three-term Fourier series was fit to the experimental data to yield a model for the stress-strain relationship:

$$\sigma = a_0 + \sum_{n=1}^{3} a_n \cos(n\omega\epsilon) + \sum_{n=1}^{3} a_n \sin(n\omega\epsilon) \tag{3.54}$$

$$= a_0 + a_1 \cos(\omega\epsilon) + b_1 \sin(\omega\epsilon) + a_2 \cos(2\omega\epsilon) + b_2 \sin(2\omega\epsilon) + a_3 \cos(3\omega\epsilon) + b_3 \sin(3\omega\epsilon) \tag{3.55}$$

Using the `curve fitting` function in the scipy package [125] for Python 3.8.5, a script (see Appendix D.1.1) was written to determine the coefficients listed in Eq. (3.55). The extracted experimental data from the stress-strain plot in [120] was inputted into the function to optimize the parameters of the Fourier series to fit the experimental data. The values obtained using the Python script for all five bias magnetic field intensities given by [120] are listed in Tab. 3.5. The values for the constants are only applicable when the stress is in units of Pascals and the strain in units of m/m; therefore, the extracted values in Fig. 3.1 must be multiplied by appropriate magnitudes to obtain a usable form. Figure 3.6 shows the three-term Fourier series functions plotted as solid lines compared against the experimental data plotted using circular markers.

In terms of error statistics, the three-term Fourier series functions were proficient in fitting the nonlinear curves of the extracted data. The coefficient of determination (R^2) for all five bias magnetic field intensity curves, Tab. 3.6, were equal to 1.000. Based on this parameter, the Fourier series model was deemed to have an exceptional goodness of fit. The Krishnamurthy Ramberg-Osgood model had a coefficient of determination of 0.987 for the stress interval of $0 - 40$ MPa. A parametric study was conducted by changing the number of terms in the Fourier series. The error statistics significantly decreased when changing the number of terms from one to three. However, the differences in statistics for the three- and four-term series were much less, suggesting the error values to be converging. The three-term series was found to best capture the experimental data

Table 3.5: Constants for a three-term Fourier series modeling the stress-strain relationship of Terfenol-D.

H_0 [kA/m]	a_0	a_1	a_2	a_3
0	6.014e+07	-7.667e+07	1.307e+07	3.428e+06
30	4.516e+06	-4.995e+07	5.479e+07	-9.352e+06
60	-7.478e+10	1.063e+11	-3.580e+10	4.248e+09
90	-1.394e+10	-3.549e+09	2.471e+10	-7.222e+09
120	-2.539e+10	3.543e+10	-1.118e+10	1.147e+09

H_0 [kA/m]	b_1	b_2	b_3	ω
0	3.938e+06	-1.537e+07	3.498e+06	-1.359e+03
30	-8.170e+07	1.264e+07	5.558e+06	-8.744e+02
60	3.790e+10	-2.880e+10	6.634e+09	2.771e+02
90	6.221e+10	3.025e+09	-2.233e+10	4.564e+01
120	1.476e+10	-1.094e+10	2.451e+09	3.149e+02

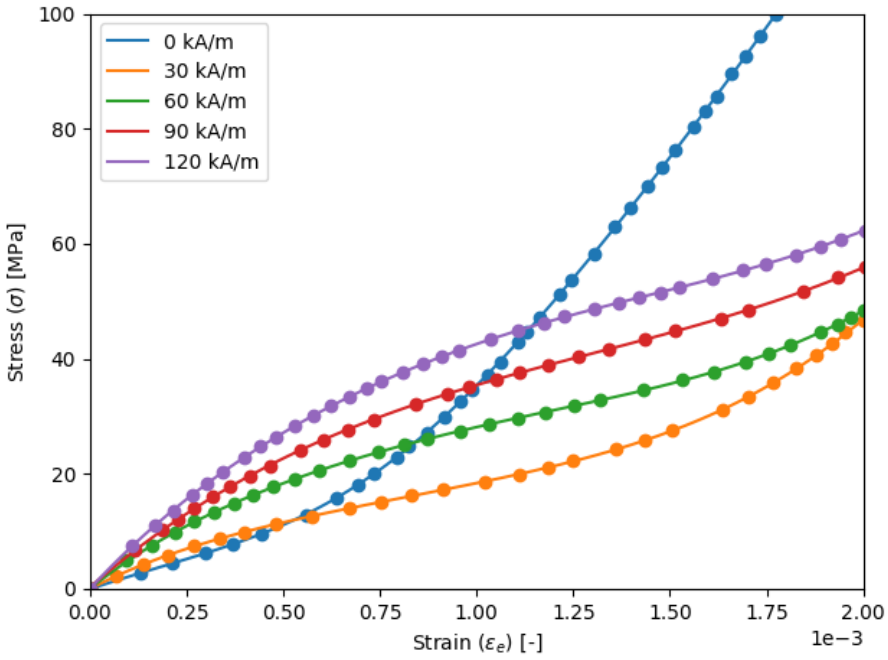


Figure 3.6: The proposed three-term Fourier series fitted the experimental data given by [120]. The experimental data is plotted using the circular markers, and the color of the marker corresponds to the respective bias magnetic field intensity.

Table 3.6: Error statistics for the fit of the proposed three-term Fourier series model against experimental data found in [120].

H_0 [kA/m]	Max Error	MSE	RMSE	R^2
0	2.813e-01	1.117e-02	1.057e-01	1.000
30	2.236e-01	4.737e-03	6.883e-02	1.000
60	1.702e-01	4.077e-03	6.386e-02	1.000
90	2.982e-01	1.295e-02	1.138e-01	1.000
120	2.406e-01	4.568e-03	6.759e-02	1.000

as the inclusion of any additional terms showed only a marginal decrease in error. Increasing the number of terms beyond three would introduce additional complexity and computational time to a model while offering marginal, if any, advantages. An increased number of terms would lead to overfitting of the experimental data.

While the Fourier series captured the shape of the experimental data, there was concern about whether this type of function was appropriate for modeling this relationship. Fourier series are commonly employed to reduce noise in signals or characterize oscillating curves, but the stress-strain curves did not demonstrate any oscillatory behavior. When expanding the domain (or range) beyond that given by the experimental data, the sinusoidal nature of the model was visible as the Fourier series began to oscillate. The proposed three-term Fourier series model was determined to be valid for modeling the nonlinear stress-strain behavior for the domain given by the experimental data in [120]. Unlike Krishnamurthy's Ramberg-Osgood model, the general Fourier series could be applied to all five bias magnetic field intensity values, thus, improving the characterization of Terfenol-D.

A model in the same formulation presented in [53] was also defined for the case of a stress-controlled experiment:

$$\epsilon_e = a_0 + \sum_{n=1}^{3} a_n \cos(n\omega\sigma) + \sum_{n=1}^{3} a_n \sin(n\omega\sigma) \tag{3.56}$$

$$= a_0 + a_1 \cos(\omega\sigma) + b_1 \sin(\omega\sigma) + a_2 \cos(2\omega\sigma) + b_2 \sin(2\omega\sigma) + a_3 \cos(3\omega\sigma) + b_3 \sin(3\omega\sigma) \tag{3.57}$$

A model capable of estimating the strain in terms of the stress was implemented in the model presented in Section 3.3. A three-term, stress-controlled Fourier series relationship was derived

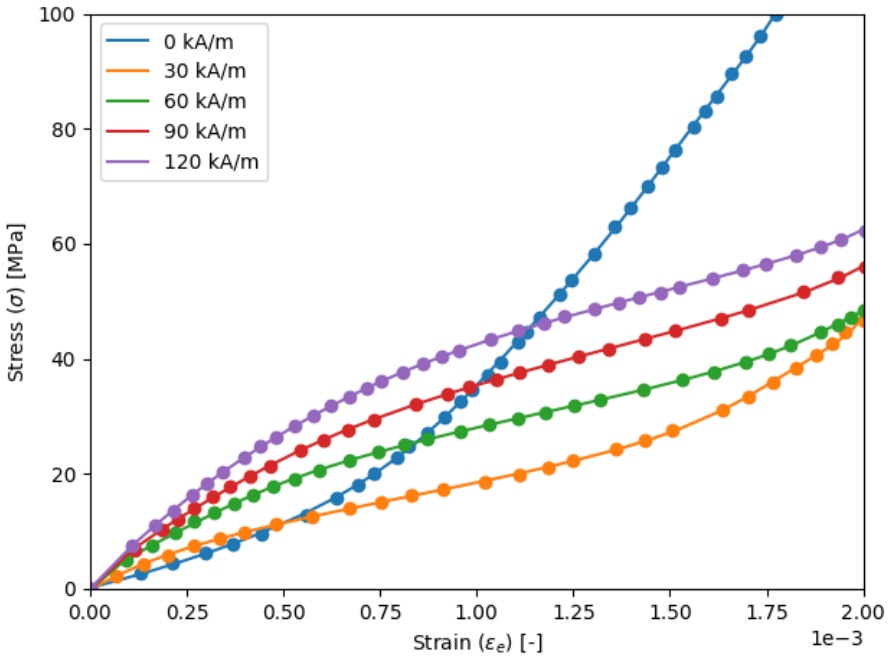


Figure 3.7: The coefficients for a stress-controlled Fourier series were determined to model the experimental data and resemble the formulation presented in [53].

to replace the previously proposed Ramberg-Osgood model. The constants for the stress-controlled model are listed in Tab. 3.7. The fitted Fourier series shown in both Figs. 3.7 and 3.8 were determined using the extracted experimental data. In Fig. 3.7, the three-term Fourier series was plotted with the experimental data as inputs. Figure 3.8 illustrates a shortcoming of the Fourier model: the oscillatory nature. In this plot, all of the Fourier series functions were plotted with an input stress range of $0 - 100$ MPa. Above the maximum stress observed for each bias magnetic field intensity curve from the experimental data, three of the curves began oscillating in this increased stress range.

3.4.2 Compliance Term

The compliance term was calculated per the relationship defined in Eq. (3.7). Using the NumPy package [126] in Python, the difference between adjacent points was calculated as

$$\Delta(i) = x(i+1) - x(i) \tag{3.58}$$

The discrete differences in the stress and strain values were calculated for the experimental data

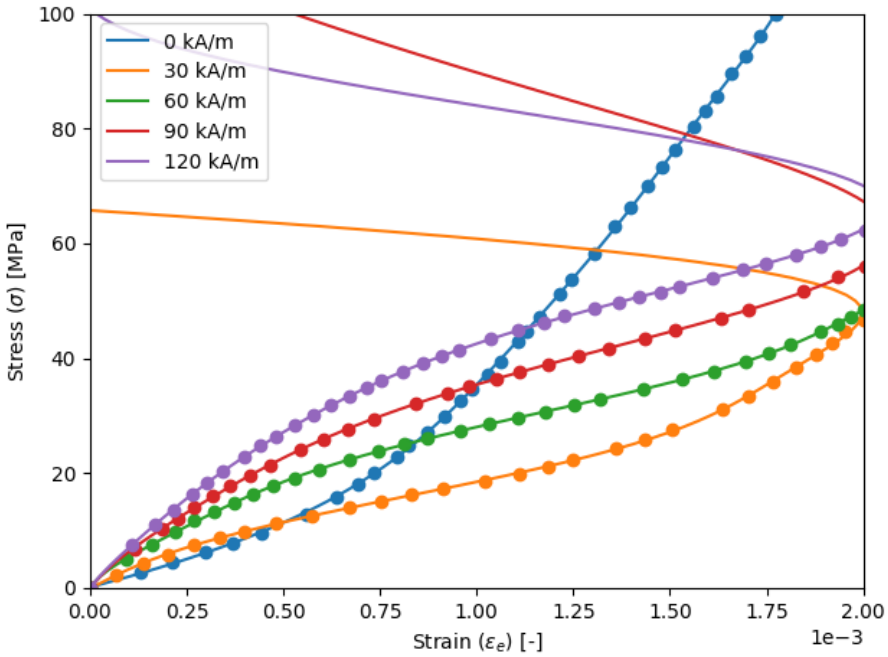


Figure 3.8: The stress-controlled three-term Fourier series was plotted with a set stress range of $0 - 100$ MPa. Three of the curves displayed the sinusoidal behavior expected from the sum of trigonometric functions within this domain.

Table 3.7: Constants for a three-term stress-controlled Fourier series modeling the stress-strain relationship of Terfenol-D. The magnitudes of the coefficients differ for the stress-controlled formulation than those of the strain-controlled.

H_0 [kA/m]	a_0	a_1	a_2	a_3
0	-3.452e-04	-7.911e-04	1.110e-03	1.827e-05
30	6.298e-05	-5.030e-04	6.784e-04	-2.355e-04
60	2.253e-03	-1.721e-03	-8.233e-04	2.936e-04
90	9.068e-04	-9.994e-04	1.244e-04	-3.043e-05
120	8.345e-04	-8.739e-04	-8.843e-06	4.839e-05
H_0 [kA/m]	b_1	b_2	b_3	ω
0	2.597e-03	1.838e-04	-2.232e-04	2.484e-08
30	1.978e-03	-4.279e-04	-1.999e-04	5.459e-08
60	-1.744e-03	6.585e-04	1.825e-04	5.186e-08
90	8.264e-05	1.234e-05	3.604e-05	4.946e-08
120	-3.521e-04	2.919e-04	-1.442e-05	5.745e-08

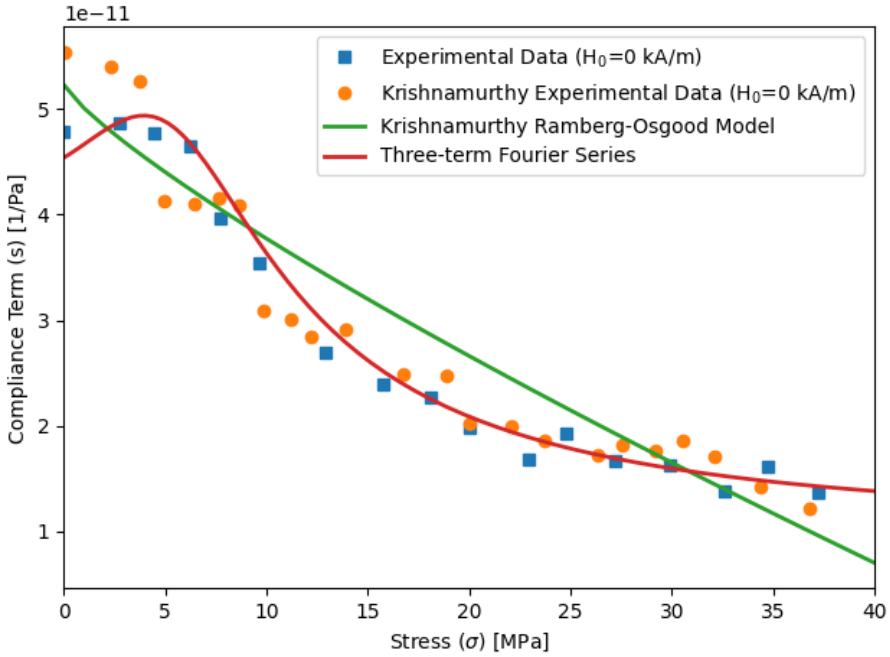


Figure 3.9: The compliance values for the $H_0 = 0$ kA/m experimental data and the proposed functions to model the experimental stress-strain behavior.

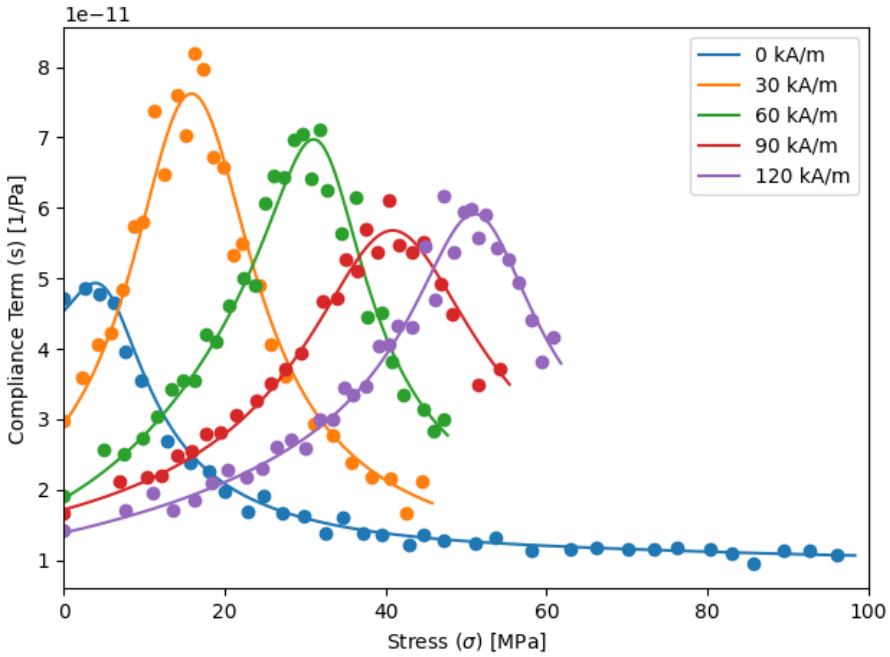


Figure 3.10: The compliance values for the proposed Fourier series modeling the stress-strain behavior of Terfenol-D. The experimental data from [120] is plotted using circular markers.

plotted in Fig. 3.4 in addition to the proposed three-term Fourier series within the Python script. The compliance term was then found by dividing the discrete difference of the strain by the discrete difference of the stress. The derivative of the Krishnamurthy Ramberg-Osgood function was taken such that the compliance term was calculated by Eq. (3.9). Figure 3.9 shows that the experimental compliance values were comparable except for stress values less than 5 MPa. The proposed Ramberg-Osgood model resembled a linear function. Based on Eq. (3.9), the y-intercept of the compliance term curve using the proposed Ramberg-Osgood function is equal to α_e, which correlates to $1/E$. The exponent of Eq. (3.9) equaled 0.818; therefore, the curve was not truly linear and displayed some curvature. The Fourier series resembled the Butler [120] experimental data the closest of the presented curves/data. Figure 3.10 illustrates the relationship between the experimental [120] and Fourier series compliance terms. The compliance terms for the Fourier series were only calculated to the maximum stress values for the respective bias magnetic field intensities shown in Fig. 3.6 to avoid improper values due to the oscillatory nature of the Fourier series. The Fourier series tended to underestimate the peak values of the compliance terms for the respective bias magnetic field intensities, especially for the cases of $H_0 = 30$ and $H_0 = 90$ kA/m.

3.4.3 Strain-Magnetic Field Intensity Relationship

The strain-magnetic field intensity model, Eq. (3.10), was plotted for completeness of the investigation of the Krishnamurthy et al. [53] analytical model. Using the magnetic field intensity domain given by [120] of $0 \leq H \leq 1500$ Oersted, the strain was calculated, and the curves were plotted in Fig. 3.11. From initial observation, the seven curves appeared to have a similar shape as the experimental data in Fig. 3.2 for values of $H < 1000$ Oersted. Above this value, the polynomial curves appeared to diverge from the shape of the experimental data. When the experimental data and the Krishnamurthy polynomials were plotted together in Fig. 3.12, an error was observed. The magnitude of the calculated strain was six magnitudes greater than the experimental data. The cause of this error was not mentioned in [53], but it was believed that this error occurred due to improper data used in the curve fitting process. Neglecting the adjustment of the strain values in Fig. 3.2 or multiplying by the incorrect magnitude would significantly alter the relationships that deviate from the experimental data without correction.

A new quartic polynomial relationship was defined to accurately reflect the strain-magnetic field intensity curves in the correct magnitude illustrated in Fig. 3.13. The same formulation pro-

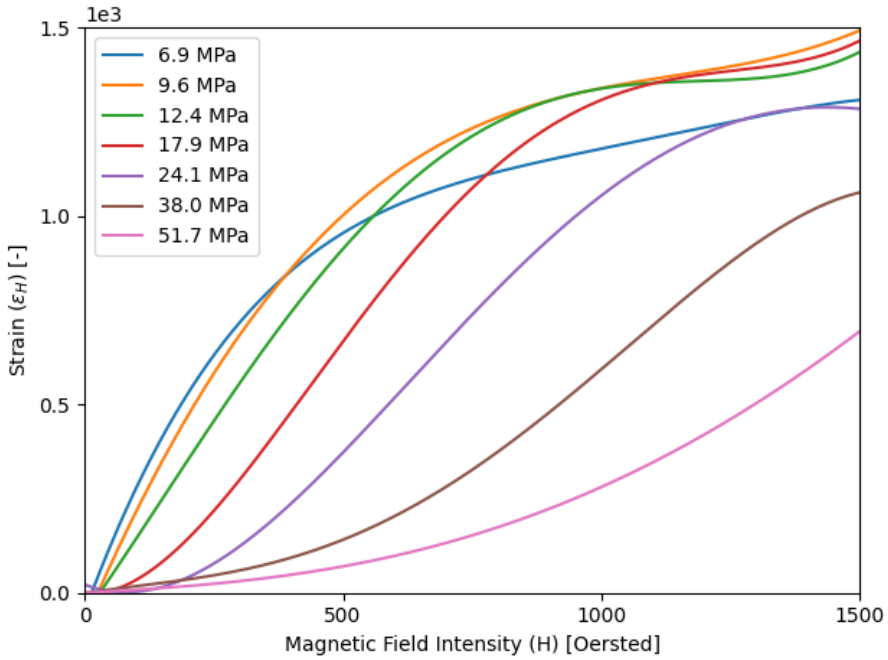


Figure 3.11: The fourth-degree polynomial representation of the strain-magnetic field curves by [53] were found to be orders of magnitude off from the experimental data.

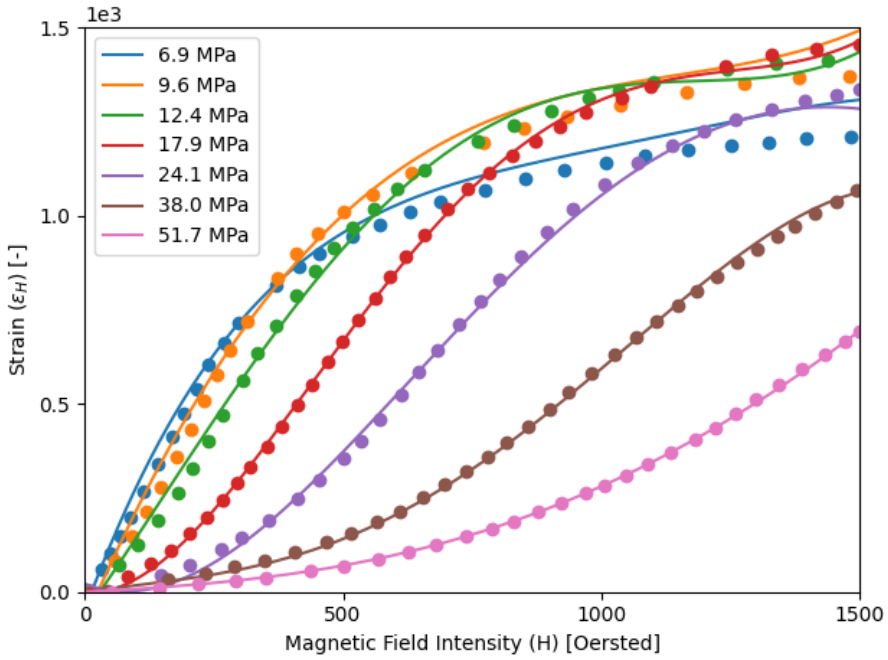


Figure 3.12: The quartic polynomial functions [53] deviated from the experimental data given by [120]. The experimental data is given by the circular markers. Note that the magnitudes of the strain values are inconsistent and should not be compared without proper adjustment of values.

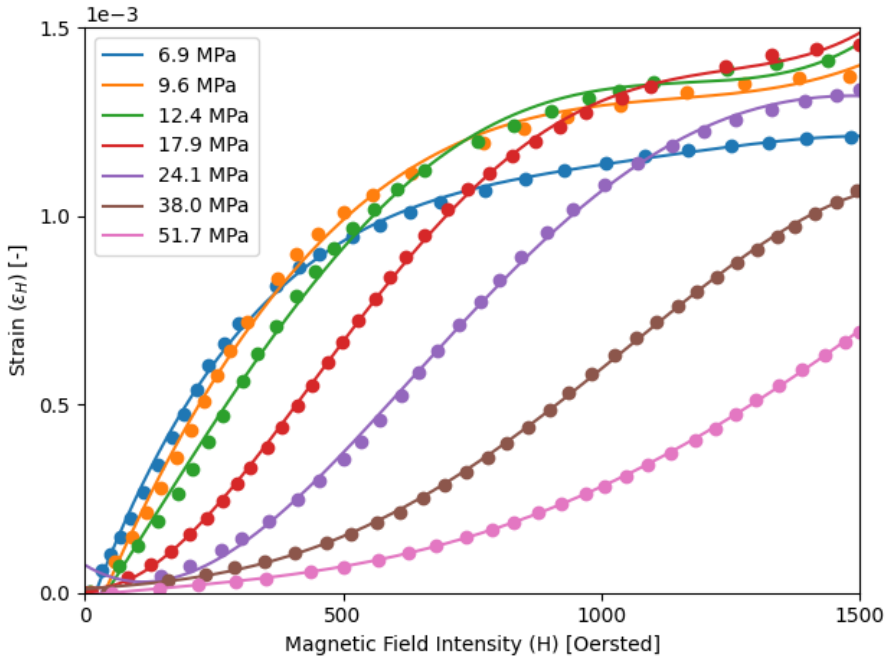


Figure 3.13: The fourth-degree polynomial strain-magnetic field intensity relationship with improved constants defined in this work.

posed in [53] was employed to improve its fit by redefining the constants. Using the `curve fitting` function [125] in Python 3.8.5, seven sets of coefficients were determined for the polynomial (one set for each prestress value). Table 3.8 lists the values of the new constants. The new constants displayed an improved fit to the experimental data as the maximum difference between the experimental data and the quartic curves displayed in Tab. 3.9 was on the magnitude of 10^{-5} for six of the seven curves. Considering all the statistical parameters examined, the most significant error occurred for the 9.6 MPa prestress curve. As the magnetic field intensity approached the maximum of the given domain, this curve began to diverge from the experimental data. Deviation in the prestress curve of 24.1 MPa was also observed about $H = 0$ Oersted as the curve for this function did not converge to the origin.

Because a Fourier series was shown to be a viable model for the nonlinear stress-strain curves, multi-term Fourier series models were investigated for modeling the strain-magnetic field intensity behavior of Terfenol-D. A three-term, Fig. 3.14, and five-term, Fig. 3.15, Fourier series were implemented using the `curve fitting` function [125] in Python. The three-term series exhibited several undulations at upper values in the domain, especially for the prestress curves of 6.9 and 9.6

Table 3.8: Constants for a fourth-degree polynomial series modeling the strain-magnetic field intensity relationship of Terfenol-D.

σ_j [MPa]	a_0	a_1	a_2	a_3	a_4
6.9	-7.627e-05	3.743e-06	-4.677e-09	2.766e-12	-6.202e-16
9.6	-1.249e-04	3.341e-06	-2.454e-09	3.884e-13	1.434e-16
12.4	-8.829e-05	2.081e-06	8.286e-10	-2.422e-12	9.359e-16
17.9	1.319e-05	-1.484e-07	5.164e-09	-5.291e-12	1.567e-15
24.1	7.460e-05	-8.414e-07	4.226e-09	-3.029e-12	6.363e-16
38.0	9.900e-06	1.099e-07	-1.179e-11	9.215e-13	-4.345e-16
51.7	-6.904e-06	1.410e-07	-1.477e-10	3.974e-13	-1.019e-16

Table 3.9: Error statistics for the fit of the improved fourth-degree polynomial model against experimental data found in literature.

σ_j [MPa]	Max Error	MSE	RMSE	R^2
6.9	4.946e-05	1.965e-10	1.402e-05	0.999
9.6	9.407e-05	6.665e-10	2.582e-05	0.997
12.4	6.631e-05	5.630e-10	2.373e-05	0.997
17.9	3.018e-05	9.024e-11	9.499e-06	1.000
24.1	4.608e-05	1.569e-10	1.252e-05	0.999
38.0	1.520e-05	3.319e-11	5.761e-06	1.000
51.7	4.343e-06	3.592e-12	1.895e-06	1.000

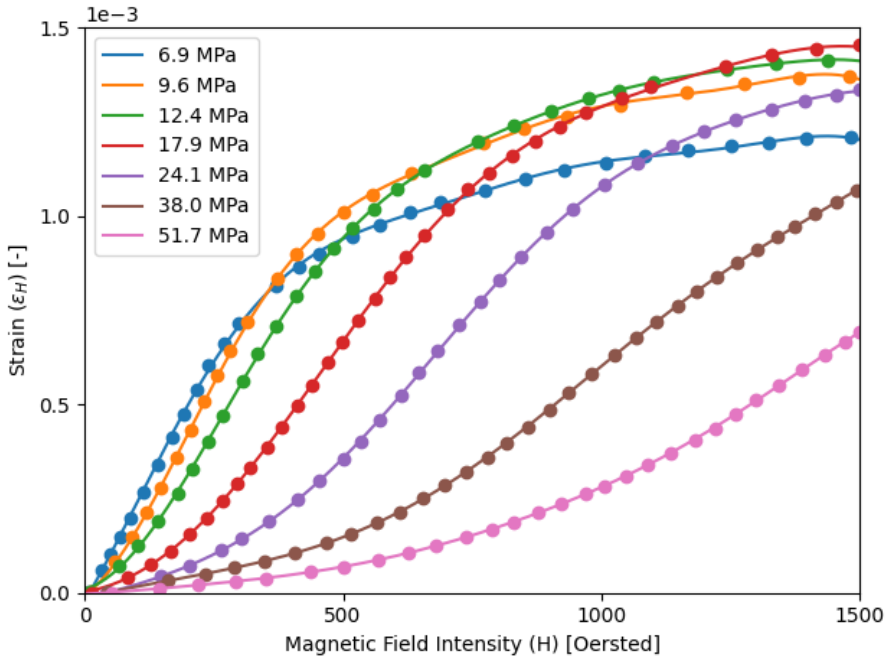


Figure 3.14: The three-term Fourier series modeling the experimental strain-magnetic field data.

MPa at approximately 1200 Oersted. The five-term Fourier series, given as

$$\epsilon_H = a_0 + \sum_{n=1}^{5} a_n \cos(n\omega H) + \sum_{n=1}^{5} a_n \sin(n\omega H) \tag{3.59}$$

$$= a_0 + a_1 \cos(\omega H) + b_1 \sin(\omega H) + a_2 \cos(2\omega H) + b_2 \sin(2\omega H) + a_3 \cos(3\omega H) + b_3 \sin(3\omega H) + a_4 \cos(4\omega H) + b_4 \sin(4\omega H) + a_5 \cos(5\omega H) + b_5 \sin(5\omega H) \tag{3.60}$$

eliminated these oddities and resulted in smooth curves to model the experimental data with the constants listed in Tab. 3.10. The error values in Tab. 3.11 improved over those for the fourth-degree polynomial by an average of an order of magnitude. As with the stress-strain curves, the Fourier series depended on the experimental data domain because of the Fourier series's oscillatory nature. Beyond the presented upper limit of the magnetic field intensity domain, the Fourier series models diverged from expected behavior by oscillating like trigonometric functions. Their use beyond the given maximum magnetic field intensity was discouraged without further data to define the region.

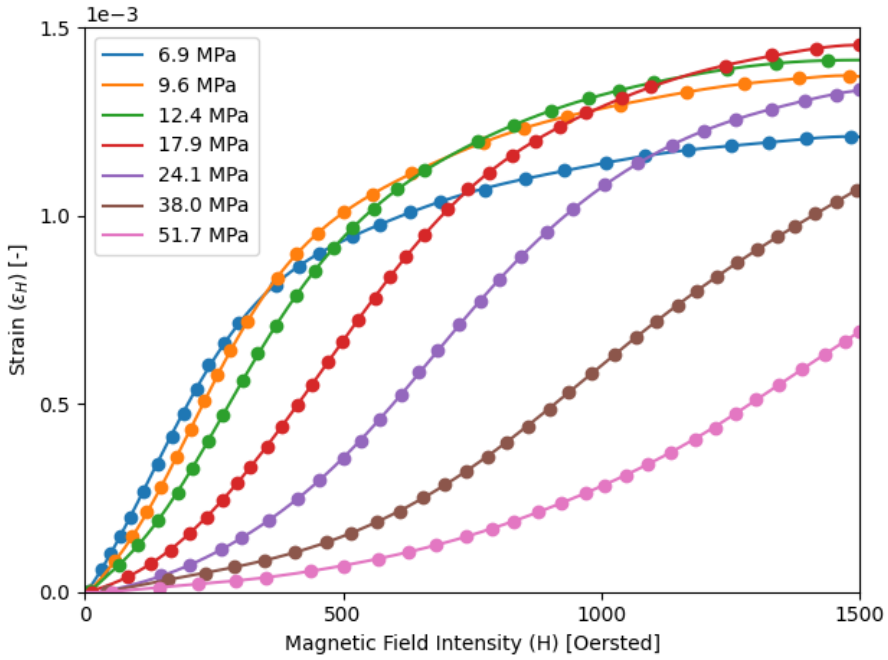


Figure 3.15: The five-term Fourier series modeling the experimental strain-magnetic field data.

Table 3.10: Constants for a five-term Fourier series modeling the strain-magnetic field intensity relationship of Terfenol-D.

H_0 [Oe]	a_0	a_1	a_2	a_3	a_4	a_5
6.9	7.661e-04	-5.638e-04	-1.605e-04	-2.392e-05	-1.874e-05	-8.350e-06
9.6	-7.461e+00	8.915e+00	4.200e-02	-2.202e+00	7.644e-01	-5.959e-02
12.4	-6.199e+00	7.312e+00	1.855e-01	-1.878e+00	6.251e-01	-4.527e-02
17.9	6.199e-04	-7.950e-04	1.393e-04	8.646e-05	-4.433e-05	-9.222e-06
24.1	2.977e-04	-8.025e-04	4.385e-04	1.566e-04	-8.730e-05	-1.707e-05
38.0	1.938e-03	-1.196e-03	-1.358e-03	4.784e-04	1.590e-04	-3.061e-05
51.7	8.732e-01	-1.027e+00	-3.327e-02	2.710e-01	-9.109e-02	6.923e-03
H_0 [Oe]	b_1	b_2	b_3	b_4	b_5	ω
6.9	1.437e-04	1.812e-04	5.785e-05	9.246e-06	6.583e-06	2.558e-03
9.6	8.952e+00	-7.572e+00	2.148e+00	1.902e-02	-6.523e-02	9.871e-04
12.4	7.522e+00	-6.270e+00	1.706e+00	4.486e-02	-5.557e-02	8.909e-04
17.9	3.943e-04	1.487e-04	-1.417e-04	-4.039e-05	1.272e-05	2.307e-03
24.1	6.470e-04	1.950e-04	-2.408e-04	-5.713e-05	1.938e-05	2.224e-03
38.0	-2.367e-03	7.513e-04	5.828e-04	-1.619e-04	-2.269e-05	1.756e-03
51.7	-1.066e+00	8.894e-01	-2.429e-01	-6.336e-03	8.141e-03	1.071e-03

Table 3.11: Error statistics for the fit of the proposed five-term Fourier series model against experimental data found in literature.

H_0 [Oe]	Max Error	MSE	RMSE	R^2
6.9	8.019e-06	6.752e-12	2.598e-06	1.000
9.6	5.324e-06	4.442e-12	2.108e-06	1.000
12.4	4.525e-06	1.754e-12	1.325e-06	1.000
17.9	3.442e-06	2.384e-12	1.544e-06	1.000
24.1	3.414e-06	2.365e-12	1.538e-06	1.000
38.0	2.839e-06	1.051e-12	1.025e-06	1.000
51.7	2.133e-06	5.150e-13	7.176e-07	1.000

3.4.4 Piezomagnetic Coefficient

A method for calculating the piezomagnetic coefficient was given by Eq. (3.11) as the slope of the strain-magnetic field intensity curve. The approach utilized in Section 3.4.2 to calculate the compliance term was also employed for the piezomagnetic coefficient using discrete differences within a Python script. The extracted experimental data was compared with the polynomial model proposed in [53] in Fig. 3.16. The polynomial piezomagnetic coefficient curve proposed in [53] was found to be a factor of 10^6 off from the experimental data, so Fig. 3.16 shows the polynomial coefficient curve multiplied by a factor of 10^{-6}. The primary reason for the inclusion of the Krishnamurthy quartic polynomial in Fig. 3.16 was to examine the shape of the curve against the experimental data. As with the compliance term, the Krishnamurthy relationship differed at the lower values of the domain, i.e., $H < 200$ Oersted. At magnetic field intensity values above that value, however, the polynomial function followed the shape of the experimental data until a deviation began at $H = 1000$ Oersted. The improved fourth-degree polynomial relationship proposed in this work showed the same behavior as the Krishnamurthy polynomial at the lower end of the domain; however, this model followed the experimental data closer than the Krishnamurthy polynomial. The polynomial formulation for the strain-magnetic field intensity curves was deemed insufficient for modeling the relationship based on the deviations of the piezomagnetic coefficient values. Figure 3.17 exhibits the failure of the polynomial model to represent the piezomagnetic coefficients at magnetic field intensities below approximately 400 Oersted for prestress values of 6.9, 9.6, and 12.4 MPa.

As with the compliance term, the Fourier series best fit the experimental data for the piezomagnetic coefficient as evident in Fig. 3.18. The oscillating nature of the Fourier series was

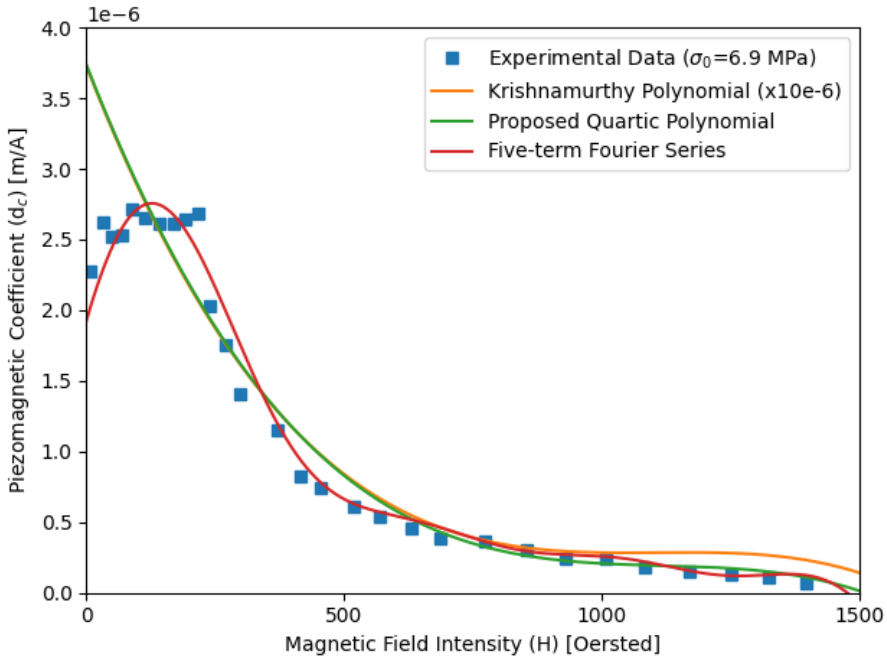


Figure 3.16: The proposed five-term Fourier series was found to resemble the experimental piezomagnetic coefficients while the polynomial models diverged at low magnetic field intensities for a prestress of 6.9 MPa.

noted at the extremes of the domain. Near $H = 0$ Oersted, the Fourier series diverged from the expected behavior. Local minima appeared such that the slope of the curve changes sign at approximately 50 Oersted; the 12.4 MPa prestress curve demonstrates this behavior. Deviations were also observed at the upper limit of the domain where the slope of the lines (except for prestress values of 38.0 and 51.7 MPa) began to increase negatively because of the Fourier series's propensity to oscillate. A case of overfitting was noticed for the two smallest prestress values around 1200 Oersted. The curves for the seven prestress values undulated throughout the magnetic field intensity domain to attempt to fit the experimental data. Further investigation of the number of terms in a Fourier series could yield coefficients that display a smoother behavior. Increasing the number of terms, though, can lead to overfitting.

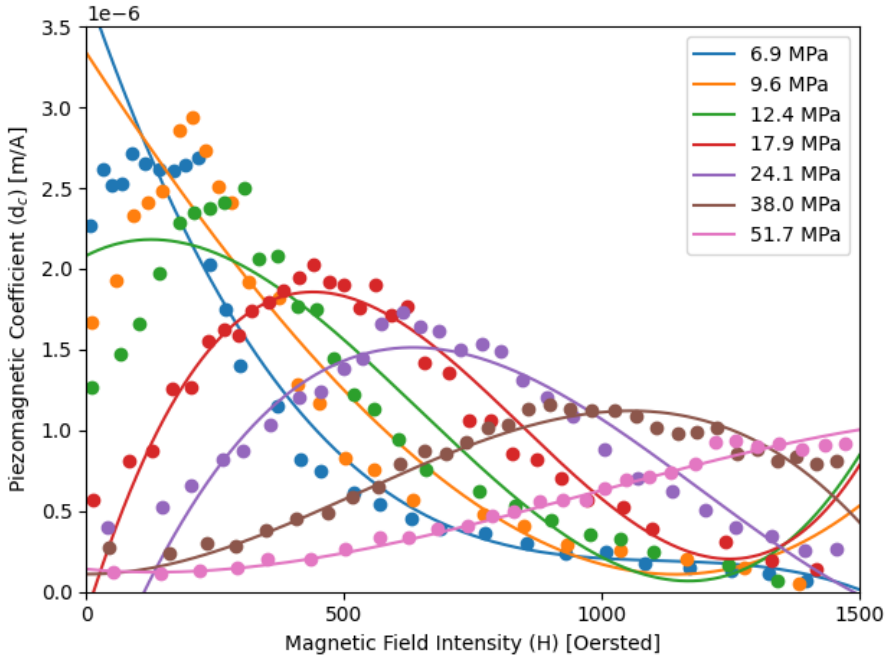


Figure 3.17: The proposed quartic polynomial fails to model the lower three prestress values at low magnetic field intensities.

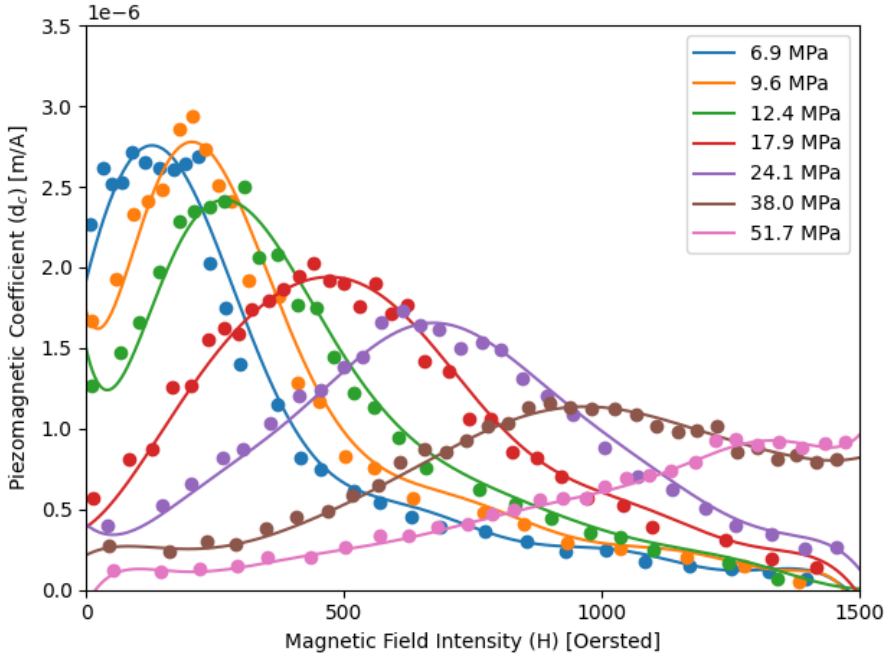


Figure 3.18: The proposed five-term Fourier series was found to resemble the calculated piezomagnetic coefficients using experimental data.

3.5 Examination of the Krishnamurthy Model

The iterative stress calculation scheme proposed in [53] (Ssection 3.3) was scripted using the Python script in Appendix D.1.2 to determine the stress induced in the magnetostrictive material. The symmetric laminate used in this effort consisted of eight unidirectional CFRP laminae made from HexPly AS4/3501-6 prepreg material with a single magnetostrictive lamina located at the midplane of the laminate; therefore, there were four plies of CFRP above and below the magnetostrictive lamina. Bending effects were concerned negligible by selecting a symmetric laminate. Terfenol-D was used as the magnetostrictive material, and its bulk properties were employed to remain consistent with [53]. The thicknesses of both the CFRP and Terfenol-D laminae were equal to 125 micrometers (0.125×10^{-3} meters). The orientation angles of the laminae were also assumed equal such that the CFRP laminae were aligned in the same direction. No bias magnetic field was assumed applied during the laminate fabrication, i.e., $H_0 = 0$ kA/m, so the direction of the Terfenol-D lamina was irrelevant in this investigation. The incorporation of the bias field direction would be embodied in the model by manipulating the material properties. No discussion was given to characterizing the bias field direction. The Terfenol-D lamina was assumed to be aligned with the CFRP fibers such that its orientation yielded the most significant magnetostriction effect. The laminae were aligned along the x-direction of the laminate to coincide with the Euler-Bernoulli model derivation, where the only non-zero strain occurred in the x-direction. An arbitrary, constant magnetic field of 10 Oersted was selected as the only loading source during the analysis. In a realistic experiment, the magnetic field would be sinusoidal and dependent on time, according to Eq. (3.19). This examination was concerned with an instantaneous value, so time effects were neglected.

3.5.1 Verbatim Krishnamurthy Model

The first attempt at implementing this model followed the proposed method verbatim to that in [53]. The A_1 and A_2 values were calculated according to Eq. (3.28) where only the elastic moduli were used in the calculation of these elastic terms. The longitudinal elastic modulus, E_1, of the CFRP was employed because the Krishnamurthy model was only concerned with the x-direction, which coincided with the longitudinal direction of the fibers. Though the Ramberg-Osgood and quartic relationships were shown to yield inconsistent results compared to experimental data, they were incorporated into the verbatim implementation of the model for thoroughness. The version of

Eq. (3.10) used in the verbatim implementation did not include the α_0 term as the summation of the fourth-degree polynomial was presented in [53] to begin with an index of one ($i = 1$).

The variation in magnitude of the polynomial expression for the strain-magnetic field intensity relationship in [53] was evident in Fig. 3.19. The left-hand side of Eq. (3.40) appeared to be a flat curve compared to the magnetic field intensity strain, ϵ_H. The magnetic field intensity strain value was found to differ depending on whether the α_0 term was included. The effect of the term was noticeable for stress values below approximately 38 MPa. Convergence of the two solutions was observed above this stress value. The predicted stress in the magnetostrictive material for the given conditions briefly converged with and without the α_0 term at approximately 18 MPa before diverging until approximately 38 MPa.

The effect of the applied magnetic field intensity was examined by varying its magnitude. As the applied magnetic field intensity increased above 10 Oersted, though, the solution convergence and the nature of the quartic polynomial curves changed. The curves of the fourth-degree polynomial with and without the α_0 were influenced by the increase of the magnetic field intensity because of the polynomial formulation; the magnetic field intensities were raised to powers of 1, 2, 3, and 4. The effect of the powered terms dominated the effect of the inclusion of the α_0 term. The difference in the curves of the two variations decreased as the magnetic field intensity was increased until they appeared to collapse on top of each other. The convergence of a solution, i.e., when the LHS and RHS curves were equal, disappeared at $H = 168$ Oersted.

The left-hand side of Eq. (3.40) contained the calculated elastic strains that combined the geometry of the laminate with the stress-strain relationship of the magnetostrictive material. Figure 3.20 illustrates that the effect of the $LHS1$ term defined as

$$LHS1 \coloneqq \frac{\sigma h_m}{A_1 + A_2} \tag{3.61}$$

was negligible compared to the discussed stress-strain relationships. The A_1 and A_2 terms, which both equaled 69.3 MPa-m, dominated in terms of magnitude the value of the σh_m term. The overall $LHS1$ term increased linearly as the stress increased. The proposed Ramberg-Osgood relationship yielded comparable values to the three-term Fourier series until approximately 44 MPa, at which the slope of the Ramberg-Osgood model changed sign and the estimated strain decreased. This behavior was previously discussed in Section 3.4.1 to arise due to a curve fitting process using only

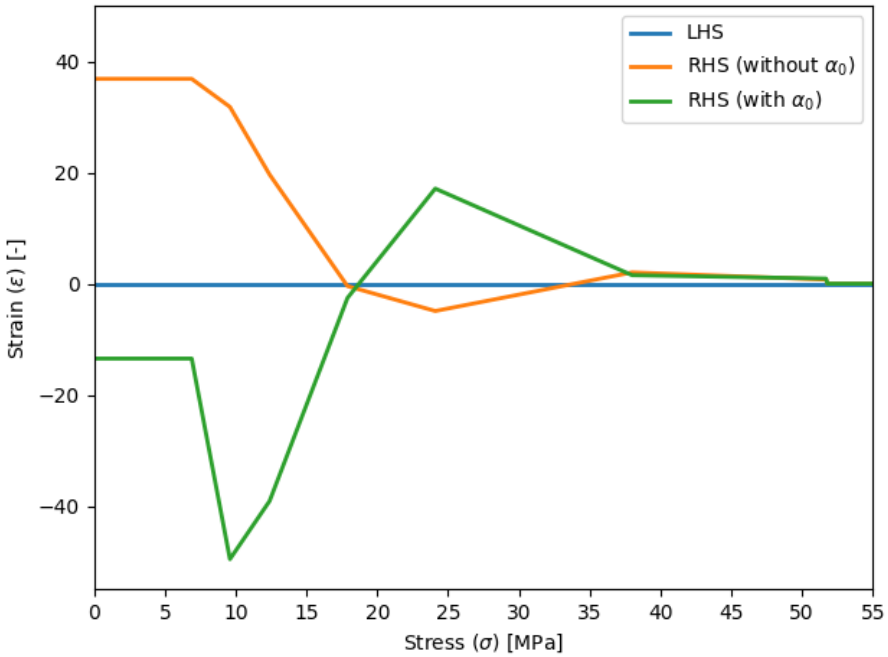


Figure 3.19: The values of the iterative Euler-Bernoulli model proposed in [53] for $H_0 = 0$ kA/m and $H = 10$ Oersted converged around at a stress value of 18 MPa.

a subset of the experimental stress. While the proposed Ramberg-Osgood model was presented for stress values above the suggested 40 MPa limit in Fig. 3.20, the results are not considered above the limit due to known divergence issues.

3.5.2 Improved Models used with the Krishnamurthy Model

The improved nonlinear constitutive relationships proposed in this work were substituted for their respective terms in Eq. (3.40). Figure 3.21 illustrates the improvement of the magnitude of the strain predictions of the new relationships over those from [53]. The improved relationships for the magnetic field intensity strain (RHS) significantly differed from the sum of elastic strain terms (LHS). The convergence of the elastic and magnetic field strains occurred only for the case of the improved fourth-degree polynomial, including the α_0 term at approximately 1 MPa. Figure 3.22 shows a more informative view of Fig. 3.21 by decreasing the range of the y-axis. The variation of the three magnetic field intensity strains was found to be on the magnitude of 10^{-4} which was an order of magnitude less than the elastic strain values; hence, little variation was illustrated in Fig. 3.21. Convergence of a solution for the stress was not found for any applied magnetic field greater

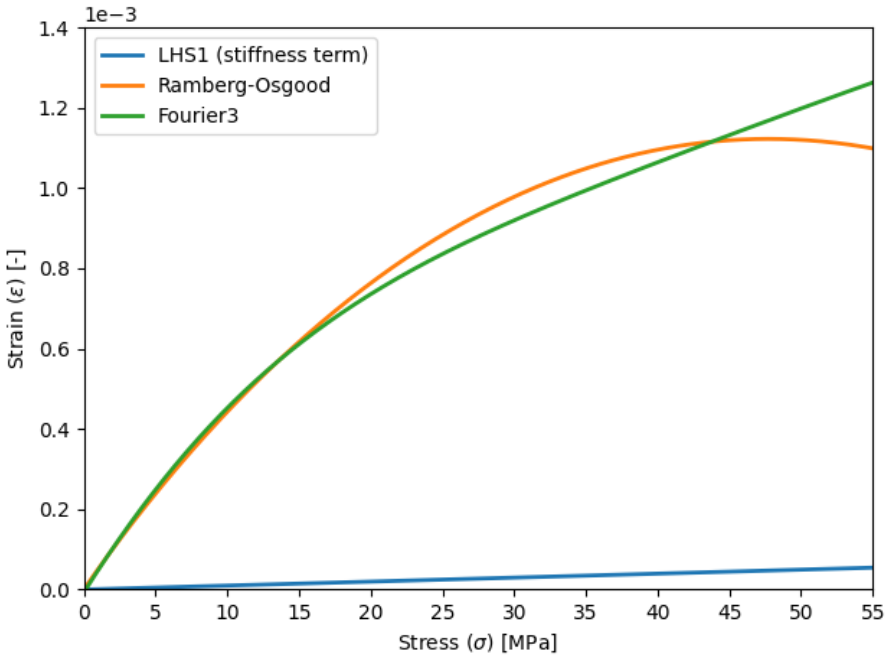


Figure 3.20: The elastic strain calculated from the characterization of the stress-strain curves dominated the left-hand side of the iterative equation.

than 8 Oersted with a bias magnetic field intensity of 0 kA/m.

Several issues were noted with the iterative model proposed in [53]. First, the model utilizes the nonlinear constitutive relationships for the stress-strain and strain-magnetic field curves. The elastic strain calculated using the proposed Ramberg-Osgood function would be incorrect above a stress value of 40 MPa because of its derivation with a subset of the experimental stress range given by [120]. The strains equated in the iterative scheme are considered to be elastic and magnetic field intensity strains. Two elastic strains were involved: a stress-strain relationship for Terfenol-D and a term associating the stress of the Terfenol-D with the elastic properties of the laminae above and below the magnetostrictive lamina. The stress involved with the stress-strain relationship was assumed to be continuous for the experimental bias magnetic fields. The magnetic field intensity strain employed seven defined experimental prestress values. The iterative scheme equated the continuous stress with the piecewise-defined prestresses. To use a continuous prestress value, [53] employed a linear interpolation method to determine the coefficients for stress values not equal to the defined prestress values. The linear interpolation method was employed for both the Krishnamurthy quartic polynomial and the improved strain-magnetic field relationships proposed in this work.

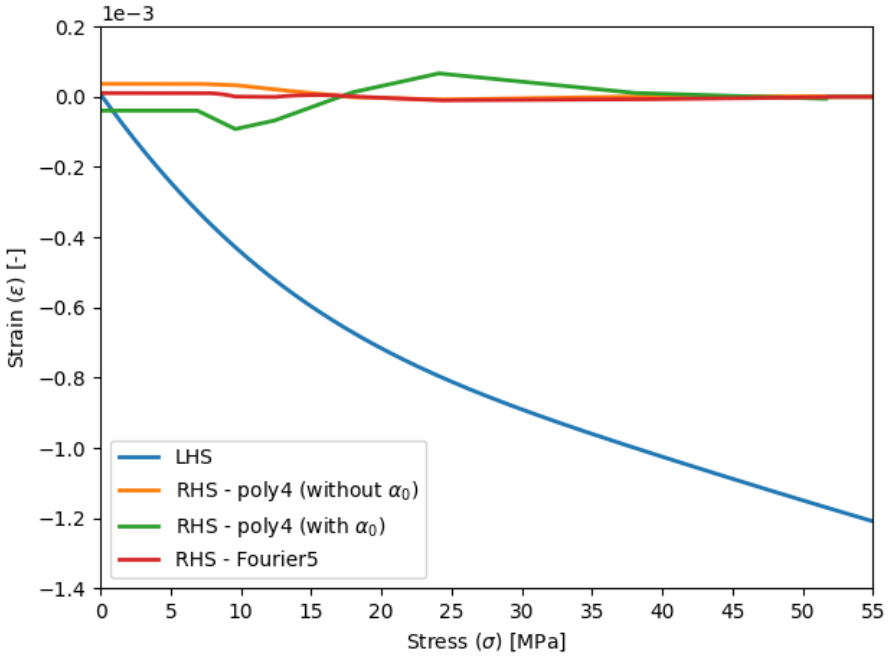


Figure 3.21: The improved nonlinear constitutive relationships were found to not converge.

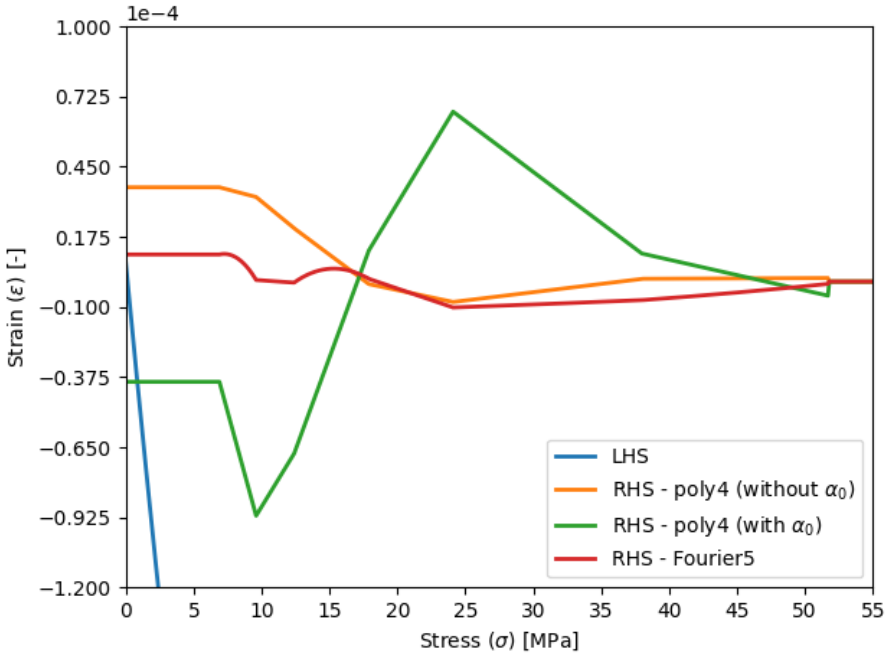


Figure 3.22: The elastic strain calculated from the characterization of the stress-strain curves dominated the left-hand side of the iterative equation.

The quantification of the laminate's structure in the iterative model, Eq. (3.28), was inconsistent with a similar quantity derived in the classical lamination theory. The Krishnamurthy model implemented an A_c term calculated using only the elastic modulus. The classical lamination theory utilizes the transformed reduced stiffness matrix, which is a combination of elastic properties. Using the transformed reduced stiffness matrix, the A_{11} term in the **ABD** matrix, which corresponds to the A_c term, was calculated for a four-ply, unidirectional CFRP stack of AS4/3501-6 as 203.0 MPa-m; an increase in the value of nearly 222%. Increasing the values of the A_i terms would only decrease the value of $LHS1$, further decreasing the likelihood of the solution converging.

3.6 Analytical Modeling Conclusions

The formulation for calculating the stress in a magnetostrictive material caused by an applied magnetic field proposed in [53] was examined in this work. Several conclusions were observed through the examination of this scheme: i) unsuitable characterization of nonlinear relationships for Terfenol-D, ii) an absence of convergence of the analytical model to determine the induced stress, and iii) limitations of the proposed model in [53]. The previously proposed functions to model the nonlinear stress-strain and strain-magnetic field intensity relationships of Terfenol-D failed to accurately quantify the relationships. Fourier series functions were introduced in this work to account for the nonlinear nature of these two relationships for Terfenol-D. The coefficient of determination was equal to 1.0 for all of the lines fit using the newly proposed Fourier series.

In terms of the stress-strain relationship, a three-term Fourier series was found to model a stress domain 60 MPa greater than the Ramberg-Osgood function previously proposed in [53] for the $H_0 = 0$ kA/m curve. The Fourier series was defined for all five bias magnetic field intensities given in [120] which were previously undefined. The Fourier series fit for the stress-strain relationship was found satisfactory with an RMSE on the order of 10^{-1} or less; it failed to diverge from the experimental data due to the curve fitting process employed to define the constants. Though a polynomial function was investigated to model the strain-magnetic field intensity relationship, a five-term Fourier series was ultimately found to exemplary model the relationship with RMSE values less than 10^{-6}. Increasing the number of terms in the Fourier series was necessary for the strain-magnetic field intensity relationship to eliminate slight deviations in the curves from the experimental data. The examination of derived terms from the stress-strain and strain-magnetic field intensity

relationships further proved the viability of a Fourier series to quantify these relationships. The previously proposed formulations to quantify the compliance term and piezomagnetic coefficient in [53] failed to capture the change in curvature of the experimental data, as seen in Figs. 3.9 and 3.16.

Using a Python script, the model proposed in [53] was examined. The original relationships proposed in [53] were first examined despite knowing the nonlinear relationships included in the model were inaccurate. The iterative stress scheme converged using the previously proposed, inaccurate relationships at approximately 18 MPa; the magnitude of the magnetic strain, ϵ_H, was nearly 10^4 times greater than the elastic strain term, ϵ_e. From this result, the previously proposed formulations from [53] were concluded to have errors. The Fourier series relationships proposed in this work failed to achieve a viable convergence for an induced stress value using the methodology suggested in [53]. It was concluded that the investigated formulation contained limitations and inadequacies to calculate the induced stress in a magnetostrictive material. The scheme was limited in application to only account for an applied magnetic field. No externally applied mechanical loads or prestress values on the magnetostrictive material were included in the formulation of the scheme. Furthermore, two elastic terms included in the scheme, A_1 and A_2, to account for the composite material structure were found to differ by 222% from a comparable value calculated using the classical lamination theory. Improvements to analytically model an embedded sensing lamina are necessary to yield the induced stress and subsequently induced sensing voltage for the investigated NDE method.

Chapter 4

Characterization of Effective Material Properties

4.1 Motivation for Characterizing Effective Properties

The previous investigations of the embedded Terfenol-D NDE method utilized the Terfenol-D sensing lamina in two forms: a particulate composite and a film. Two modeling investigations, Krishnamurthy et al. [53] and Chen and Anjanappa [127], employed the bulk material properties of Terfenol-D to reflect the embedded particulate lamina and thick film, respectively. The nonlinear characterizations proposed by [53] modeled the properties of Terfenol-D in both investigations. The effects of the composition of the sensing laminate were neglected; i.e., the inclusion of a host material was not considered.

Effective properties combine the properties of all the constituents in their determination. While the response of Terfenol-D is nonlinear and dependent on the magnetic and stress state of the material (Sections 3.2 and 3.4), the response of other constituents may be linear and elastic. Defining effective properties allows for the inclusion of the composition of the sensing lamina in determining the elastic response of the overall laminate. Analytical models utilize material properties to quantify materials and their response to loads. Eliminating a constituent material, e.g., an epoxy matrix, reduces the resemblance of the model to experimentation. The examination of the effective properties of a particulate sensing lamina sought to improve the modeling of its elastic behavior.

These properties may also apply to determining the elastic fields of the laminate.

4.2 Purpose of Effective Properties

Analyzing the behavior of composite materials becomes increasingly complicated as the number of planes of symmetry increases in addition to their inhomogeneous nature created through the combination of multiple constituents. A micromechanical approach models a laminate by modeling the constituent materials at their respective locations. The interfacial effects between constituents can be incorporated into the model to resemble the behavior of a composite better. A macromechanical approach, such as the classical lamination theory in Section 2.3, considers the composite material a solid, homogeneous body. Elastic properties used in this approach include the elastic modulus and Poisson's ratio. For a CFRP, the carbon fibers and the polymer matrix have unique material properties. The fibers are assumed transversely isotropic, whereas the matrix material is assumed isotropic. Combining these two materials yields a transversely isotropic lamina with properties calculated as functions of the constituent properties and volume fractions. For transversely isotropic fiber-reinforced composites, methods have been previously presented to compute the elastic properties of the laminate [1]:

$$E_1 = E_f V_f + E_m V_m \tag{4.1}$$

$$E_2 = \frac{E_f E_m}{E_f V_m + E_m V_f} \tag{4.2}$$

$$\nu_{12} = \nu_f V_f + \nu_m V_m \tag{4.3}$$

$$G_{12} = \frac{G_f G_m}{G_f V_m + F_m V_f} \tag{4.4}$$

where the subscripts f and m denote the fiber and matrix, respectively, and V_i is the volume fraction of the respective material in the lamina.

Effective material properties are defined as an average of quantities across a body. Unlike the approximations presented by Eqs. (4.1) – (4.4), effective properties are equivalent in all directions; ergo, a material defined by effective properties is considered isotropic. Characterizing a particulate lamina by its effective properties eliminates the need to account for any distribution discrepancies, e.g., high or low concentrations of particles, in subsequent modeling once the effective properties are found. Particle distribution is often assumed uniform throughout a lamina. A finite element

method could be employed to determine the effective properties if conglomerations of particles or matrix material were necessary to include. The geometry of the particles also contributes to the determination of the lamina's properties. Whiskers, discs, and flakes, for example, can be considered orthotropic or transversely isotropic based on assumptions set for the analysis. Spheres can be considered isotropic due to their symmetric geometry. Assumptions of a problem may increase the planes of symmetry for an individual particulate inclusion due to its scale such that the particulate may be considered isotropic independent of its geometry.

4.3 Bulk Material Properties

The material properties for the respective materials investigated in this work were collected and compared from literature [21,123,128–131]. Krishnamurthy et al. [53] along with Chen and Anjanappa [127] modeled the magnetostrictive lamina sandwiched in a composite for sensing purposes with the bulk properties of Terfenol-D given by the manufacturer ETREMA Products, Inc. (now TdVib LLC) listed in Tab. 4.1. However, the properties of particles have been found to differ from those of the material in bulk form. The crystal structure of a material determines the response of the material to loads; therefore, the quantifiable properties are directly related to the crystal structure. For a magnetic material, the properties associated with magnetism are related to the existence of dipoles within the crystal structure. Terfenol-D is commonly distributed in cylindrical rods for use as transducers. The bulk material has been engineered to have a sizeable magnetostrictive response along its axial direction by producing the cylinders with "a directional solidification process which produces a high degree of crystal orientation" [132]. Transforming the Terfenol-D from a bulk material to a particle form requires consideration of the crystallographic direction to limit any effects on the magnetic properties [21, 75, 132]. A design utilizing Terfenol-D likely seeks to utilize the material's high value of magnetostriction. Any reduction in this response would negate the material selection process considered to elect the usage of Terfenol-D.

While research has been conducted on the properties of Terfenol-D in bulk form, the ability to mechanically analyze micrometer-sized particles independently has yet to be demonstrated. Kaleta et al. [23] presented obtained properties using rods of Terfenol-D powder composites and varying bias magnetic fields to control the alignment of dipoles in Terfenol-D particles in the composites. Dobrzański et al. [22] investigated the properties of varying volume fractions of Terfenol-D

Table 4.1: Material properties of Terfenol-D given by the manufacturer [25].

Physical Property	Metric Value
Density (ρ)	9.25 g/cc
Ultimate Tensile Strength (σ_{UTS})	28.0 MPa
Elastic Modulus (E)	$25.0 - 35.0$ GPa
Compressive Strength (Y)	700 MPa
Bulk Modulus (K)	90.0 GPa
Electrical Resistivity (ρ)	6.00 $\times 10^{-5}$ ohm-cm
Magnetic Permeability (μ)	$4.5 - 10$
Curie Temperature (TC)	357 °C
Coefficient of Thermal Expansion (α)	12.0 μm/(m·°C)
Specific Heat Capacity (c_p)	$0.320 - 0.370$ J/g·°C
Thermal Conductivity (k)	$10.5 - 10.8$ W/(m·K)
Melting Point (T_{melt})	1240 °C
Magnetomechanical Conversion Efficiency	$49 - 56\%$
Magnetomechanical Coupling Factor	$0.7 - 0.75$
Magnetostrictive Energy Density	$0.014 - 0.025$ J/cm^3
Magnetostrictive Strain Production	$1500 - 2000$ μm/m
Sound Speed	$1640 - 1940$ m/s

embedded in a host matrix. Further complicating the classification of Terfenol-D is its nonlinear behavior, which has yet to be characterized for a particle. In bulk form, properties of the material are dependent on the magnetic and mechanical state the material is exposed to during manufacture/fabrication [120]. Both the applied bias magnetic field orientating its crystals, specifically the dipoles, and the subjected prestress on the particles affect the properties of Terfenol-D illustrated by the relationships presented in Sections 3.2 and 3.4. To demonstrate the calculation of the effective properties of a particulate sensing lamina, the calculations utilized the manufacturer's elastic properties in Tab. 4.1.

A unidirectional CFRP prepreg material was selected in this work consisting of HexTow AS4 carbon fibers and Hexcel 3501-6 structural epoxy. A CFRP prepreg consists of a layer of carbon fibers impregnated in a matrix, and individual lamina of the prepreg are stacked to form a laminate. An individual lamina is assumed to be transversely isotropic by assuming that the through-thickness and transverse direction have similar behavior. The thickness of a prepreg hinders the ability to obtain the through-thickness material properties. The elastic properties of HexPly AS4/8552 are presented in Tab. 4.2, or the elastic properties ($E_1, E_2, G_{12}, \nu_{12}$) can be approximated using Eqs. (4.1) – (4.4) with the individual constituents properties listed in Tabs. 4.3 and 4.4. The material properties

Table 4.2: Material properties of HexPly AS4/3501-6 prepreg [128].

Property	Value
Fiber volume fraction (V_f)	60%
Longitudinal modulus (E_1)	126 GPa
Transverse modulus (E_2)	11 GPa
In-plane shear modulus (G_{12})	6.6 GPa
Major Poisson's ratio (ν_{12})	0.28
Through-thickness Poisson's ratio (ν_{23})	0.4
Longitudinal tensile strength (X_T)	1950 MPa (nonlinear, [128])
Longitudinal compressive strength (X_C)	1480 MPa
Transverse tensile strength (Y_T)	48 MPa
Transverse compressive strength (Y_C)	200 MPa (nonlinear, [128])
In-plane shear strength (S_{12})	79 MPa (nonlinear, [128])
Longitudinal thermal coefficient (α_1)	-1×10^{-6} °C^{-1}
Transverse thermal coefficient (α_2)	26×10^{-6} °C^{-1}
Thickness (t)	0.22 mm

given for the constituents by [128] differ from those provided by the manufacturer [131, 133] likely due to variations in testing. CFRP prepregs are highly susceptible to moisture ingression, which affects their properties; over time, the properties and the adhesiveness of the prepreg decrease if not cured within the recommended shelf life [131]. Other variation causes include the experimental data collection methods, the design of the testing specimen, and the actual testing instruments. Fabrication errors are prone to occur during laminate production, especially with prepreg materials, as delaminations and voids can form from incomplete adhesion between laminae or the striation of fibers.

Table 4.3: HexTow AS4 carbon fiber material properties [128].

Property	Value
Longitudinal modulus (E_1)	225 GPa
Transverse modulus (E_2)	15 GPa
In-plane shear modulus (G_{12})	15 GPa
Transverse shear modulus (G_{23})	7 GPa
Major Poisson's ratio (ν_{12})	0.2
Longitudinal tensile strength (X_T)	3350 MPa
Longitudinal compressive strength (X_C)	2500 MPa
Longitudinal thermal coefficient (α_1)	-0.5×10^{-6} °C^{-1}
Transverse thermal coefficient (α_2)	15×10^{-6} °C^{-1}

Table 4.4: Hexcel 3501-6 matrix material properties [128].

Property	Value
Longitudinal modulus (E)	4.2 GPa
Shear modulus (G)	1.567 GPa
Poisson's ratio (ν)	0.34
Tensile strength (X_T)	69 MPa
Compressive strength (X_C)	250 MPa
Shear strength (S)	50 MPa
Thermal coefficient (α_1)	45×10^{-6} °C^{-1}

4.4 Effective Property Approximations

The determination of effective material properties allows a complex material such as a particulate composite to be modeled as an isotropic material. Calculating an effective property employs an averaging method that combines the properties of the entire body at points throughout the body,

$$\overline{P} = \langle P \rangle = P_{eff} = \frac{1}{V} \int P dV \tag{4.5}$$

where P is the property of interest, V is the volume of the body, and $\overline{P}$ and $\langle \cdot \rangle$ denote an averaged quantity [134]. Mathematical expressions have been defined in literature for calculating the effective property values instead of using an integral based approach. These expressions account for the volume of the body using the volume fractions of the constituents since

$$\sum_{i=1}^{n} f_i = 1 \tag{4.6}$$

where n is the total number of constituents in a composite and f_i are the respective volume fractions. Volume fractions for the constituents have been determined using a representative volume element (RVE) [135] or ratios of the cross-sectional areas for each constituent [1].

4.4.1 Effective Elastic Properties

Five well-known effective property models are presented in the subsequent section. For the particulate laminate investigated, 1 denotes the Tefenol-D inclusion and 0 the epoxy matrix. Elasticity principles are utilized to calculate a set of properties to model a laminate. These properties include the elastic modulus E, Poisson's ratio ν, shear modulus G, and bulk modulus K.

One of the simplest models for estimating properties is the linear rule-of-mixtures method called the Voigt approximation. This approximation was derived under the assumption that the strain throughout a composite is uniform,

$$\epsilon = \epsilon_1 = \epsilon_2 = \cdots = \epsilon_i \tag{4.7}$$

and the total stress caused by force P equals the sum of the stresses experienced by the constituents

such that

$$P = \sigma A = \sum_{i=1}^{n} \sigma_i A_i \tag{4.8}$$

Considering a two-dimensional cross-section of the laminate rather than a three-dimension element, the volume fraction for a constituent is defined as

$$f_i = \frac{A_i}{A} \tag{4.9}$$

where A represents the cross-sectional area. Dividing Eq. (4.8) by A, the stresses are associated with the volume fractions found using Eq. (4.9). Since the strain experienced in each material is equal, employing Hooke's law allows for the elastic moduli to be related to the volume fraction:

$$E = E_0 f_0 + E_1 f_1 \tag{4.10}$$

A more generalized expression of the Voigt approximation for a two-phase material is

$$P = P_0 f_0 + P_1 f_1 \tag{4.11}$$

where the subscripts 0 and 1 denote the respective material. The Voigt approximation was derived to calculate the elastic modulus [136], but other properties have been estimated, such as the Poisson's ratio and specific mass [137]. Jones illustrates the use of this model to estimate the longitudinal elastic modulus, Eq. (4.1), and major Poisson's ratio, Eq. (4.3), of a fiber-reinforced composite [1]. Kursa et al. [138] suggested that the Voigt approximation could be used to estimate the effective bulk, $\overline{K}$, and shear $\overline{G}$ moduli of a two-phase composite.

Alternatively, the Reuss approximation assumes that each constituent material experiences a stress equal to that experienced by the overall composite resulting in the approximation [139]

$$E = \left(\frac{f_0}{E_0} + \frac{f_1}{E_1} \right)^{-1} = \frac{E_0 E_1}{E_0 f_1 + E_1 f_0} \tag{4.12}$$

This expression resembles that of Eqs. (4.2) and (4.4). Hill found that the Voigt and Reuss models represented the upper and lower bounds, respectively, of the elastic modulus approximations [140]. An averaged approximation of the bounds leads to a simplified Hill model which is part of the

Voigt-Reuss-Hill (VRH) approximation:

$$E = \langle E_{Voigt}, E_{Reuss} \rangle = \frac{E_{Voigt} + E_{Reuss}}{2} \tag{4.13}$$

The VRH approximation demonstrated an inefficiency and inaccuracy involved in estimating effective properties [141]. A broad range of potential predictions exists between the defined upper and lower bounds, so researchers derived additional models to fill the region between the bounds in hopes of more accurate approximations.

Hashin-Shtrikman [142] derived a model based on the interaction of the elastic properties of the constituents paired with the principle of minimum potential energy. The overall effective properties of the laminate are found using elasticity relationships:

$$E = \frac{9KG}{3K + G} \tag{4.14}$$

$$\nu = \frac{3K - 2G}{2\left(3K + G\right)} \tag{4.15}$$

The Hashin-Shtrikman bounds are given by

$$K = K_i + \frac{f_j}{\frac{1}{K_j - K_i} + \frac{1 - f_j}{K_i + G_i}} \tag{4.16}$$

$$G = G_i + \frac{f_j}{\frac{1}{G_j - G_i} + \frac{(1 - f_j)(K_i + 2G_i)}{2G_i(K_i + G_i)}} \tag{4.17}$$

with the bulk and shear moduli terms calculated as

$$K_k = \frac{E_l}{3\left(1 - 2\nu_l\right)} \qquad G_k = \frac{E_l}{2\left(1 + \nu_l\right)} \tag{4.18}$$

where $i, j, k, l = 1, 0$ for a two-phase composite. The lower bound of the model is found by setting $i = 0$, $j = 1$, and $k = l$. The upper bound is found when $i = 1$, $j = 0$, and $k \neq l$.

Mori and Tanaka [143] developed an approximation using "the solution of a single inclusion in an infinite matrix phase" [141]. The Eshelby Inclusion Method [144–146] calculated the strain in the infinite matrix based on the average stress of the matrix material of a two-phase composite. The bulk and shear moduli used to estimate the elastic modulus and Poisson's ratio in Eqs. (4.14)

and (4.15), respectively, are

$$\frac{K-K_0}{K_1-K_0} = \frac{f_1}{1+(1-f_1)\,\frac{K_1-K_0}{K_0+\frac{4}{3}G_0}} \tag{4.19}$$

$$\frac{G-G_0}{G_1-G_0} = \frac{f_1}{1+(1-f_1)\left(\frac{G_1-G_0}{G_0+\frac{G_0(9K_0+8G_0)}{6(K_0+2G_0)}}\right)} \tag{4.20}$$

Kursa et al. [138] proposed an alternative notation for the above models. A general formula, a set of four equations, were defined based on the shear modulus, bulk modulus, and volume fraction of the inclusion like that of the Hashin-Sktrikman and Mori-Tanaka models. The formulation proposed in [138] for a two-phase composite,

$$\overline{K} = f_1\alpha_i^P K_i + (1-f_1)\,\alpha_m^P K_m \tag{4.21}$$

$$\overline{G} = f_1\alpha_i^D G_i + (1-f_1)\,\alpha_m^D G_m \tag{4.22}$$

$$f_1\alpha_i^P + (1-f_1)\,\alpha_m^P = 1 \tag{4.23}$$

$$f_1\alpha_i^D + (1-f_1)\,\alpha_m^D = 1 \tag{4.24}$$

can be used for various approximations by changing the concentration factors α_r^P and α_r^D. For example, the Kursa formulation equals the Voigt approximation when

$$\alpha_i^P = \alpha_i^D = 1 \tag{4.25}$$

and the Reuss approximation when

$$\alpha_i^P = \frac{K_0}{f_0K_1+(1-f_1)\,K_1} \qquad \alpha_i^D = \frac{G_0}{f_0G_1+(1-f_1)\,G_1} \tag{4.26}$$

With the calculated effective bulk and shear moduli using Eqs. (4.21) and (4.22), elasticity relationships, Eqs. (4.14) and (4.15), were used to define the two remaining elastic terms.

4.4.2 Coefficient of thermal expansion

The effects of temperature can lead to unexpected stress and strain within a material if not considered. The classical lamination theory, for example, can be expanded to account for variations

in the coefficients of thermal expansion as thermal strains occur from a change in temperature,

$$\epsilon^T = \alpha \Delta T \tag{4.27}$$

The coefficients of thermal expansion can significantly differ between the constituent materials and impact the behavior of a material. For instance, a fiber-reinforced material can display bistability if the laminate is laid up with an asymmetrical stacking sequence [147]. The Hexcel 3501-6 epoxy has a coefficient of thermal expansion nearly four times greater than that of Terfenol-D, suggesting that it will expand and contract more than the Terfenol-D particles under a given temperature difference. The matrix, then, will induce stress on the Terfenol-D particles. As with the elastic properties, effective coefficients of thermal expansion are necessary to define a homogeneous particulate lamina.

Makarian et al. [75] showed that the most general approximations, the Voigt and Reuss models, are applicable to the coefficient of thermal expansion. In the Voigt approximation, a uniform strain exists in a material following a change in temperature. The formulation of the model is the same as that in Eq. (4.11):

$$\overline{\alpha} = f_0 \alpha_0 + f_1 \alpha_1 \tag{4.28}$$

Likewise, the Reuss approximation assumes that a uniform stress field exists throughout a composite following an imposed temperature variation and can be expressed as

$$\overline{\alpha} = \frac{f_0 E_0 \alpha_0 + f_1 E_1 \alpha_1}{f_0 E_0 + f_1 E_1} \tag{4.29}$$

This expression differs from that in Eq. (4.12) because the elastic moduli are necessary to include when examining the stress per Hooke's law. Additional models are included in Appendix A.

4.5 Evaluation of Approximations

Application of laminate models, such as the classical lamination theory, utilizes homogeneous laminae to quantify the overall laminate behavior. The individual laminae contribute to the laminate's overall stress and strain response under applied loads (e.g., mechanical, thermal, hygroscopic). The approximations presented in Section 4.4 and Appendix A transform the behavior and properties of an inhomogeneous particulate composite into a homogeneous material through aver-

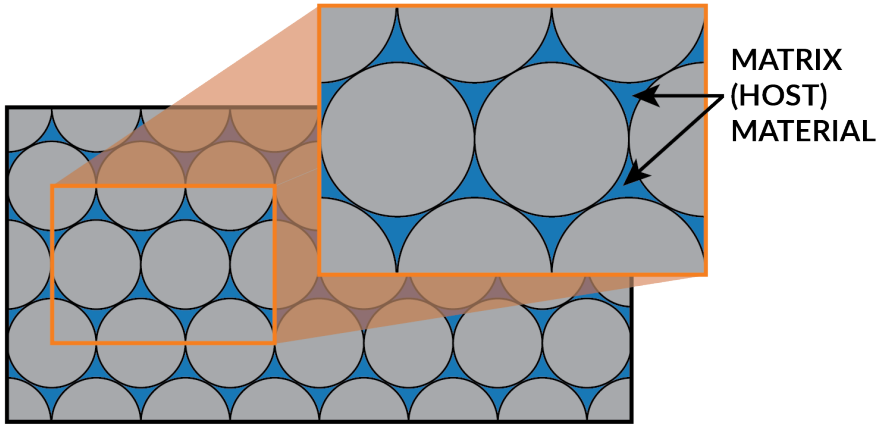


Figure 4.1: Two-dimensional visualization showing spherical inclusion packed tightly. Matrix/host material still exists within the composite in pockets between the inclusions.

aged values. Previous studies examining embedded Terfenol-D particles for damage sensing modeled the embedded magnetostrictive lamina using the bulk properties of Terfenol-D [53,55,56]. While this simplified the analysis, the results failed to appropriately account for the influence of the host material on the properties of the embedded sensing lamina. The assumption of bulk properties resembles that of a particulate lamina having a volume fraction of inclusions equal to one, or $f_1 = 100\%$. For a volume fraction of one, a solid, homogeneous piece of material would require fabrication or extraction; in the case of a lamina, a film could be produced using sputtering techniques [148, 149]. Considering the host matrix in determining the properties of the sensing lamina would improve an embedded particle sensing model to resemble experimental results better.

A particulate lamina consisting of Terfenol-D particles and Hexcel 3501-6 was selected in this work in an effort of continuity from previous studies [6, 21, 27, 57, 150]. The volume fraction of the inclusion was set to range from $f_1 = 0\%$ to 100% to illustrate the significance of the ratio of constituents on the overall property values. Limitations on the actual volume fraction of inclusions would exist for a physical lamina as the volume fraction increases. A spherical geometry does not allow for a continuous domain of material, meaning that regions of epoxy material would theoretically exist in pockets between spheres as illustrated in Fig. 4.1. Sideridis et al. [151] derived a model demonstrating the maximum volume fraction of inclusions based on their radius and the number of particles in an RVE. Investigations have implemented numerous mathematical algorithms to ascertain the precise value of the random close packing (RCP) factor for spheres [152]. The average

Table 4.5: Possible packing factor values for spherical particles based on the fabrication method employed [153].

Model	Description	Void Fraction	Packing Density
Thinnest regular packing	Cubic lattice	0.4764	0.5236
Very loose random packing	Spheres slowly settle	0.44	0.56
Loose random packing	Dropped into bed; packed by hand	$0.40 - 0.41$	$0.59 - 0.60$
Poured random packing	Spheres poured into bed	$0.375 - 0.391$	$0.609 - 0.625$
Random close packing (RCP)	Bed vibrated	$0.359 - 0.375$	$0.625 - 0.641$
Densest regular packing	FCC or HCP lattice	0.2595	0.7405

and accepted value for the RCP of spheres is a volume fraction of 64%, as given in Tab. 4.5.

4.5.1 Evaluation of Effective Elastic Property Approximations

The models presented in Section 4.4 were scripted with Python 3.8.5 (given in Appendix D.2.2) to characterize a particulate lamina consisting of Terfenol-D and Hexcel 3501-6. The volume fraction of the Terfenol-D particles was designated as the independent variable to examine the effect of the volume fraction on the overall properties. The approximations for the effective elastic, bulk, and shear moduli are presented in Figs. 4.2 – 4.4, respectively, and the findings for the effective Poisson's ratio are presented in Fig. 4.5. The estimation curves for the effective elastic properties were predicted to display positive slopes since the elastic properties of Terfenol-D, listed in Tab. 4.1, were greater than those of the Hexcel 3501-6 epoxy in Tab. 4.4.

All six models converged at the extreme volume fractions to the appropriate property values; i.e., at $f_1 = 0\%$, the properties equaled that of the Hexcel 3501-6 epoxy since no Terfenol-D inclusions were present. The Reuss, Hashin-Shtrikman, and Mori-Tanaka approximations had continuously increasing slopes as the volume fraction of particles increased. The slope of the curves changed for volume fractions above the maximum volume fraction, the RCP factor, of $f_1 = 64\%$. Estimations above this volume fraction were considered hypothetical since they cannot be realistically recreated in experimentation using uniform spherical inclusions. Below the RCP factor, the curves for the approximations appeared to have a negligible change in slope as the volume fraction of particles increased, especially in the region of $0\% \leq f_1 \leq 40\%$. A linear relationship could replace the more complicated approximations in the attainable volume fraction domain. The effective Poisson's ratio curves in Fig. 4.5, however, showed more nonlinearity than the curves of the moduli; therefore, linear

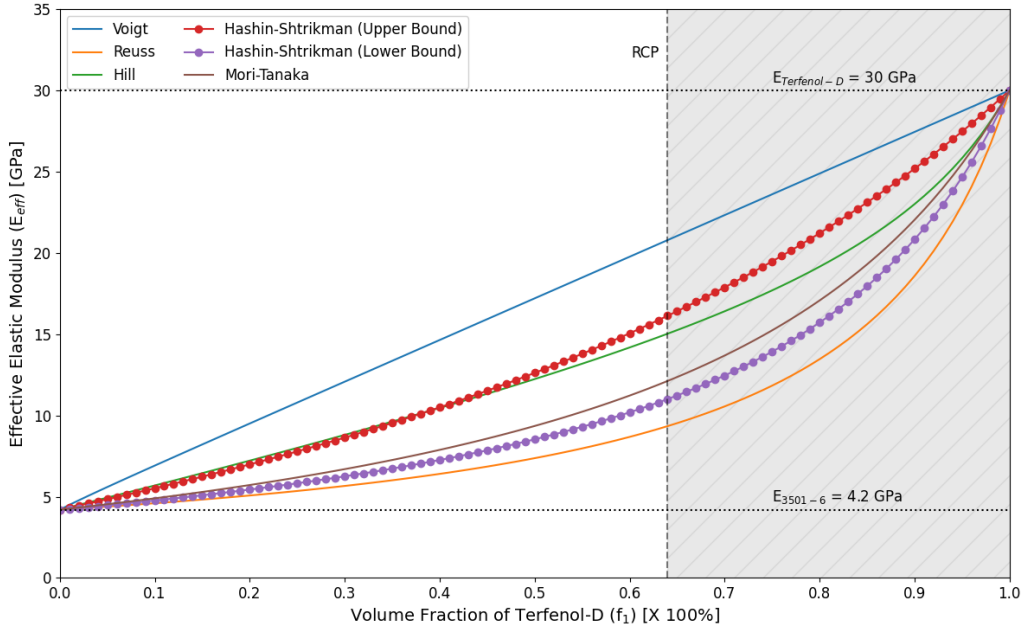


Figure 4.2: Evaluation of the effective elastic modulus for various volume fractions of Terfenol-D particles.

relationships would yield greater errors for simplifying the approximation curves.

4.5.1.1 Moduli

The findings of Hill [140] citing that the Voigt and Reuss approximations were the upper and lower bounds, respectively, for effective property estimations were confirmed by examining Figs. 4.2 – 4.4. The approximation curves for the elastic (Fig. 4.2) and shear moduli (Fig. 4.4) displayed a similar shape for the respective approximations. The Hashin-Shtrikman formulation resulted in a modified upper and lower bound tighter than that of the Voigt-Reuss bounds. Both the Hashin-Shtrikman and Mori-Tanaka estimations fell between the Voigt-Reuss bounds, and the Mori-Tanaka approximation fell between both sets of bounds. The Hill approximation, an average of the Voigt and Reuss models, corresponded to the middle of the Voigt-Reuss bounded region of possible values.

The estimations using the Hill approximation for the moduli fell between the Hashin-Shtrikman bounds except for the case of the effective bulk modulus (Fig. 4.3). The shape of the Voigt-Reuss bound for the bulk modulus was determined as the cause for this exception as a larger deviation between the bounds was observed for this property. The behavior of the Voigt approximation curve significantly differed from that of the Reuss, Mori-Tanaka, and Hashin-Shtrikman curves.

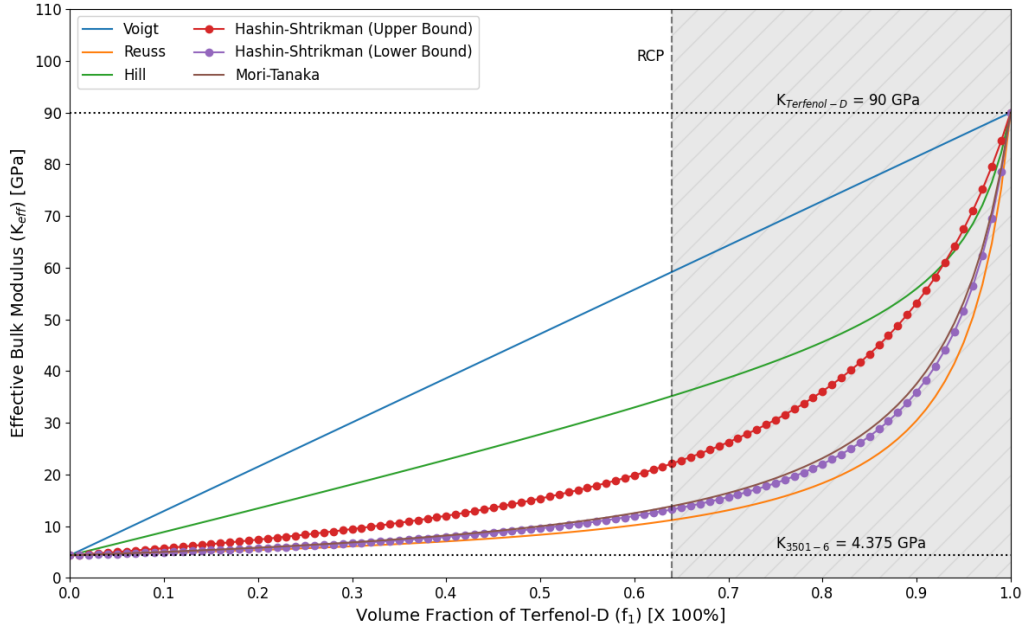


Figure 4.3: Evaluation of the effective bulk modulus for various volume fractions of Terfenol-D particles.

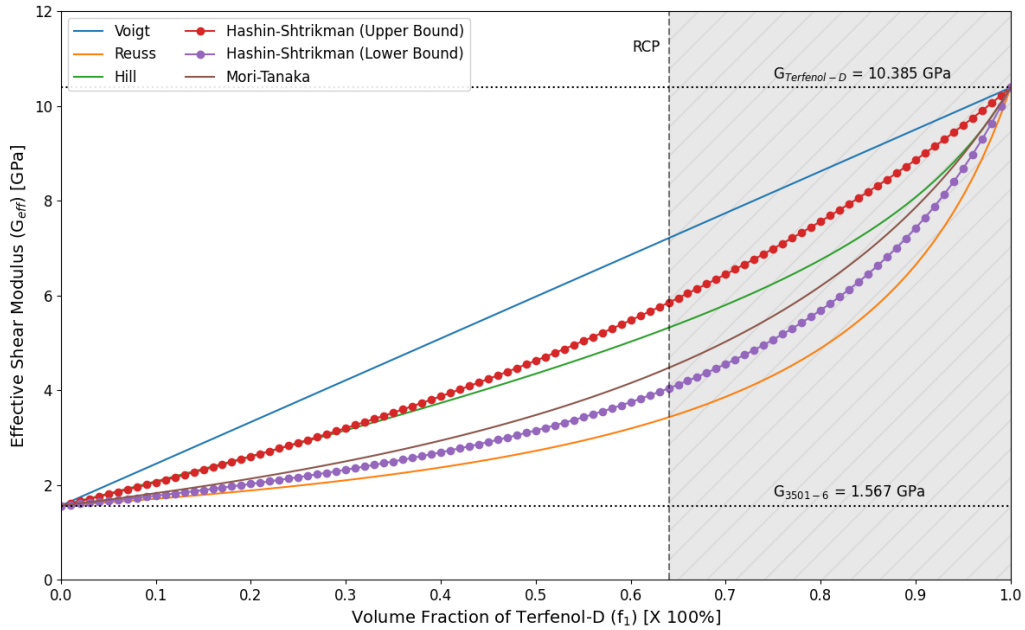


Figure 4.4: Evaluation of the effective shear modulus for various volume fractions of Terfenol-D particles.

These three formulations showed a similar shape, especially in the domain above the RCP factor. These approximations asymptotically diverged from the bulk modulus of the Hexcel 3501-6 epoxy. The slopes of their curves increased as the inclusion volume fraction increased; then it drastically increased above $f_1 = 80\%$. This Reuss approximation behavior increased the bounded region of possibility as the distance between the Voigt-Reuss bounds decreased at a slower rate than with the other two moduli.

The Voigt model was a linear function for the moduli as shown in Figs. 4.2 – 4.4 with a constant slope (gradient). The linear expression followed the form

$$\overline{P} = (P_0 - P_1)\, f_0 + P_1 \tag{4.30}$$

where P represents one of the three moduli (E, G, K). From the plots, the y-intercepts of the linear curves were visually determined as the moduli of the Hexcel 3501-6 epoxy matrix material, P_1, since they were the lesser values for the two considered constituent materials.

The elasticity relationships in Eqs. (4.14) and (4.15) were used to determine the effective elastic modulus and Poisson's ratio for the Voigt-Reuss-Hill (VRH) approximations to remain consistent with the methodology employed in the Hashin-Shtrikman and Mori-Tanaka approximations. Using the Voigt and Reuss approximations for the elastic modulus resulted in the same estimates as those found using the elasticity expression given in Eq. (4.14). The same result was not true for the effective Poisson's ratio evident in Fig. 4.6.

4.5.1.2 Poisson's Ratio

Though Loja et al. [137] stated that the Voigt approximation could be employed to estimate the effective Poisson's ratio, the elasticity expression given in Eq. (4.15) yielded a nonlinear curve that increased the bounded region of possible effective Poisson's ratios. Per its formulation, when directly applied, the Voigt approximation produced a linear curve with the form of Eq. (4.30). The approximation curve found through implementing the elastic relationship for the Poisson's ratio diverged from the Hexcel 3501-6 epoxy value and asymptotically approached the Poisson's ratio of Terfenol-D. Figure 4.5 shows that using the elasticity relationship results in a more incompressible particulate lamina than any other models. The elasticity method determined the Terfenol-D to have a more significant impact on the effective Poisson's ratio.

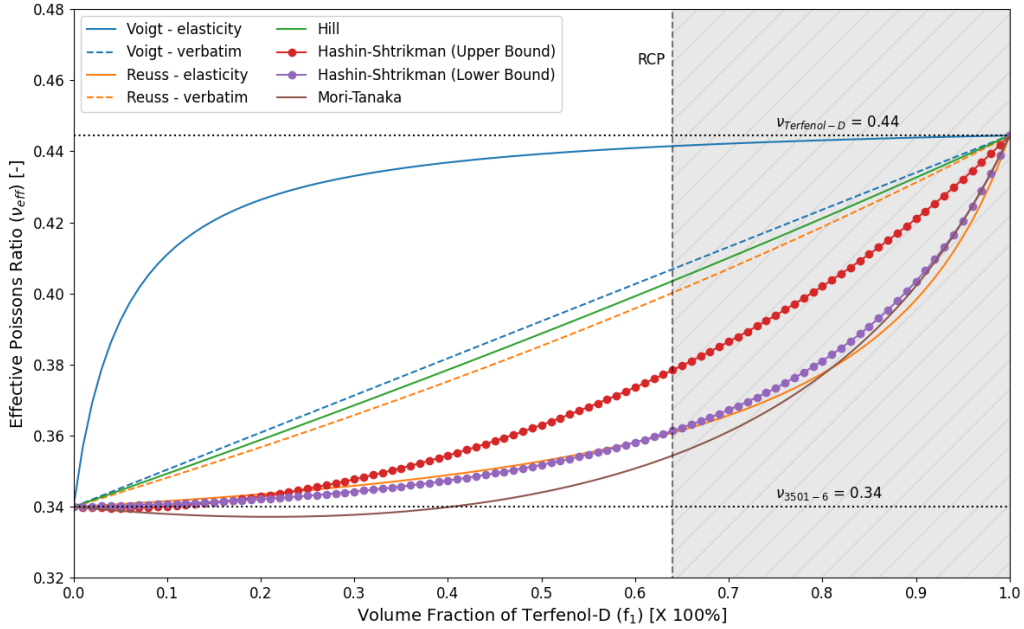


Figure 4.5: Evaluation of the effective Poisson's ratio for various volume fractions of Terfenol-D particles using elasticity relationships.

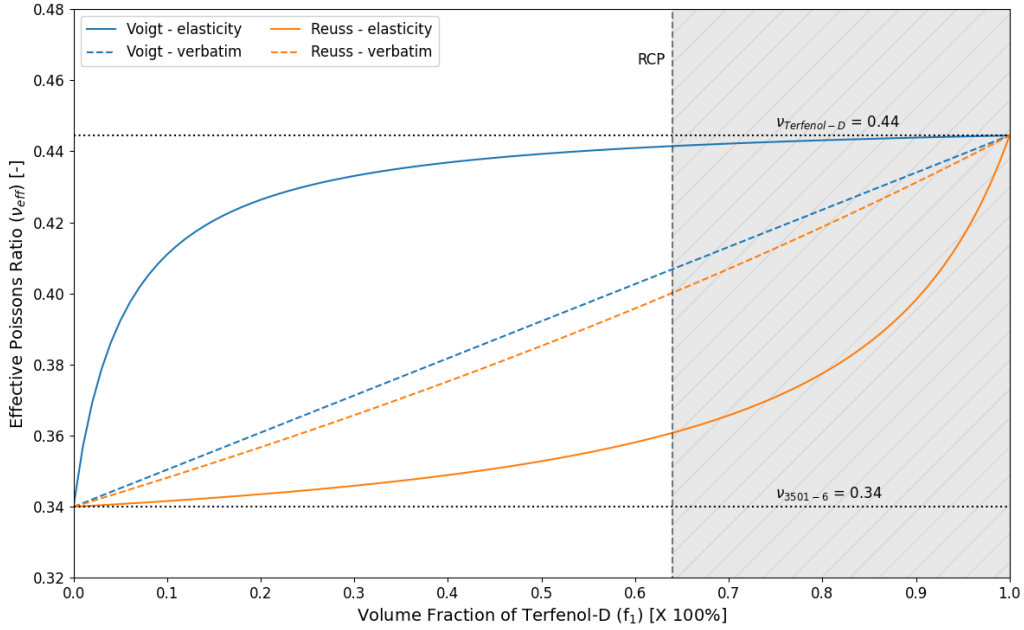


Figure 4.6: Calculating the effective Poisson's ration with the elasticity relationship results in a greater bounded region of possible values.

The shape of the Voigt and Reuss approximation curves in Fig. 4.6 appeared to be near reflections of one another across the line connecting the two extreme values of ($f_1 = 0\%$, $\nu_{eff} = \nu_{3501-6}$) and ($f_1 = 100\%$, $\nu_{eff} = \nu_{Terfenol-D}$). This line resembled that of the approximation using the Voigt approximation directly. A slight variation in the curves employing the elasticity relationships was observed such that one formulation converged more asymptotically than the other at the respective domain extremes; i.e., the Voigt curve employing the elasticity relationships converged asymptotically to the Poisson's ratio of Terfenol-D at higher inclusion volume fractions whereas the Reuss formulation asymptotically converged to the Poisson's ratio of Hexcel 3501-6 epoxy at lower volume fractions. At the RCP factor, the elasticity approach yielded effective values closest to that of the constituent materials. The Voigt approximation resulted in a 0.68% percent difference from the Poisson's ratio of Terfenol-D at the RCP factor. In contrast, the Reuss approximation estimated a difference of 5.71% from the Poisson's ratio of Hexcel 3501-6 epoxy. This difference illustrated the convergence of the Voigt-elasticity approximation to the upper bound.

An additional concern of the Poisson's ratio approximation was a discrepancy that opposed the statement of Hill [140] that the Reuss approximation coincides with the lower bound of possible property values. Applying the Reuss approximation directly for the Poisson's ratio as suggested by Loja et al. [137], resulted in an approximation predicting the second-highest effective Poisson's ratio only behind the Voigt formulations as shown in Fig. 4.5. Both the verbatim and elastic formulations of the Reuss approximation failed to estimate the lowest bound for an experimentally feasible laminate, i.e., a laminate with $f_1 < 64\%$. Below a Tefenol-D volume fraction of 61%, the lower bound of the Hashin-Shtirkman approximation predicted a lower effective Poisson's ratio than the Reuss-elasticity approximation. Only above the RCP factor was the Reuss-elasticity approximation observed to fall below the Hashin-Shtrikman lower bound and yield the lowest approximation bound for all the investigated approximations. The Mori-Tanaka approximation was found to estimate the Poisson's ratio as the smallest value until $f_1 \approx 82\%$. Further research into the effective Poisson's ratio was deemed necessary to address the discrepancies between the bounds and the approximation of the Mori-Tanaka approximation.

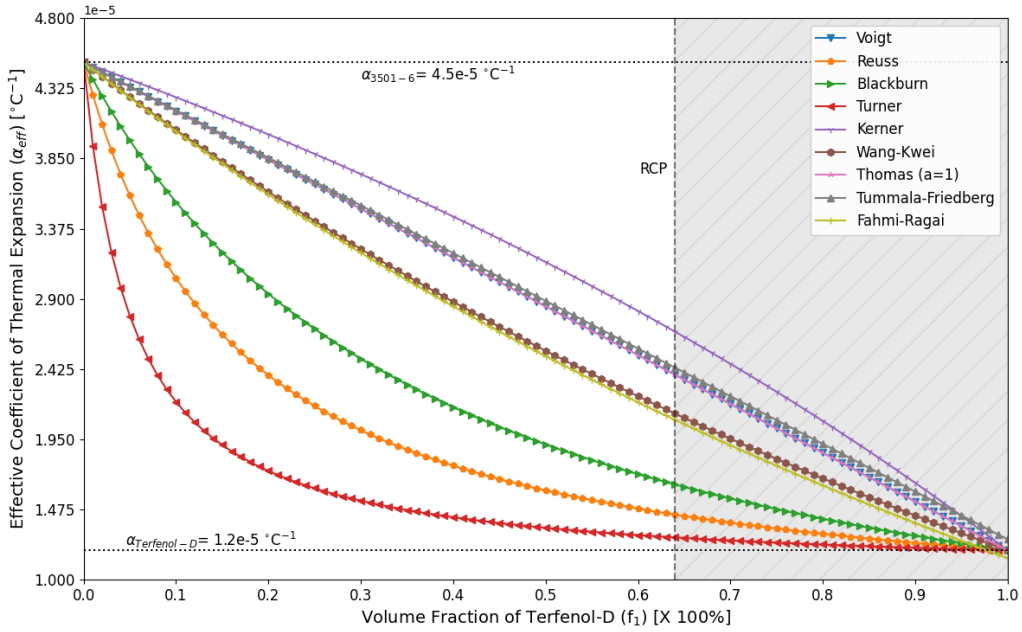


Figure 4.7: The effective coefficient of thermal expansion ranges between the values of the constituent materials. Unlike the elastic properties, the Reuss model is not the lower bound for the coefficient of thermal expansion.

4.5.2 Evaluation of Effective Coefficient of Thermal Expansion Approximations

The estimations for the effective coefficients of thermal expansion presented in Section 4.4.2 and Appendix A displayed a different behavior than the estimates for the elastic properties illustrated in Figs. 4.2 – 4.5. The coefficient of thermal expansion of the epoxy matrix ($\alpha_{3501-6} = \alpha_0 = 4.5 \times 10^{-5}$ °C^{-1}) was reported in literature to be greater than that of Terfenol-D ($\alpha_{Terfenol-D} = \alpha_1 = 1.2 \times 10^{-5}$ °C^{-1}). The Voigt and Reuss approximations were no longer found to be the upper and lower bounds of the approximations demonstrating that the findings of Hill [140] do not extend to the coefficient of thermal expansion. Figure 4.7 shows that the Turner formulation estimates the effective coefficient of thermal expansion to be less than the Reuss approximation. The Turner approximation estimated the lamina's coefficient of thermal expansion closer to the lower value than other approximations. The Thomas formulation with $a = 1$ yielded the same estimation values as the Voigt approximation because of its formulation, Eq. (A.12). The Tummala-Friedberg formulation followed the Voigt estimate closely until $f_1 \approx 35\%$ and then estimated values slightly greater than the Voigt approximation. Above the RCP factor, the Tummala-Friedberg approximations diverged

further from those of the Voigt approximation. The effective coefficient of thermal expansion was predicted for this approximation to be 7.49×10^{-7} °C^{-1} greater than that of Terfenol-D at $f_1 = 100\%$ suggesting inaccuracies with the model at high volume fractions of an inclusion. The Kerner approximation yielded the greatest effective coefficient of thermal expansion values for the domain except at the upper extreme ($f_1 \geq 95\%$); ergo, the Kerner formulation represented the upper bound for the effective coefficient of thermal expansion. The Fahmi-Ragai formulation closely resembled the Wang-Kwei approximation for inclusion volume fractions below 50%. As the volume fraction increased, these two approximations began to diverge, and the Fahmi-Ragai formulation predicted an effective coefficient of thermal expansion at $f_1 = 100\%$ to be 5.27×10^{-7} °C^{-1} less than that of bulk Terfenol-D. The rest of the models were found to converge to the coefficient of thermal expansion of Terfenol-D at $f_1 = 100\%$.

The Cribb formulation, Eqs. (A.3) – (A.5), utilized the effective bulk and shear moduli of the lamina in addition to the coefficients of thermal expansion, bulk moduli, and shear moduli of the constituent materials. Figure 4.8 shows the effective coefficient of thermal expansion curves calculated using the elastic property approximations presented in Section 4.4.1. The idea of the VRH approximation was found to be false for the coefficient of thermal expansion. The inverse of the discussed behavior in [140] was observed. The Cribb estimates using the Voigt approximations resulted in the lower bound, whereas the Reuss approximations yielded the upper bound. Figure 4.8 is almost a reflection of Fig. 4.3 regarding the behavior of the curves. The Cribb-Voigt approximation resulted in a nonlinear lower bound, while the Cribb-Reuss formulation produced a linear upper bound. The Mori-Tanaka curve resided close to the lower bound of the Hashin-Shtrikman estimate as observed with the elastic property estimates. The Hill curve deviated from this trend, but the other approximations followed the same behavior regarding the distancing from one another. Figure 4.9 shows that the Turner formulation coincides with the Cribb-Voigt approximation. Nearly half of the coefficient of thermal expansion formulations yielded estimates similar to that of the Voigt approximation.

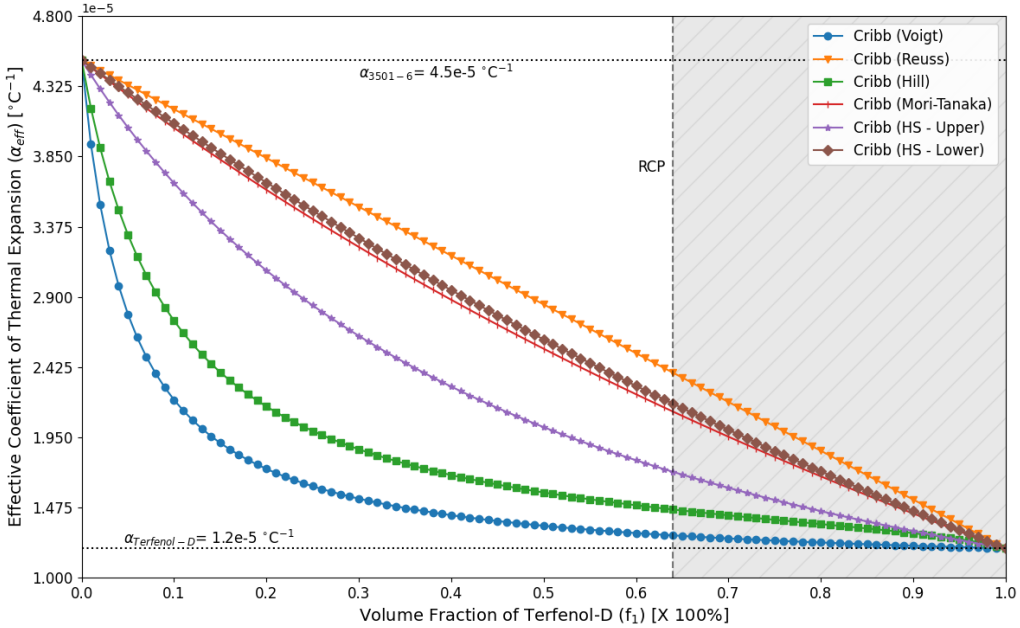


Figure 4.8: The Cribb model uses the effective bulk and shear moduli to determine the effective coefficient of thermal expansion.

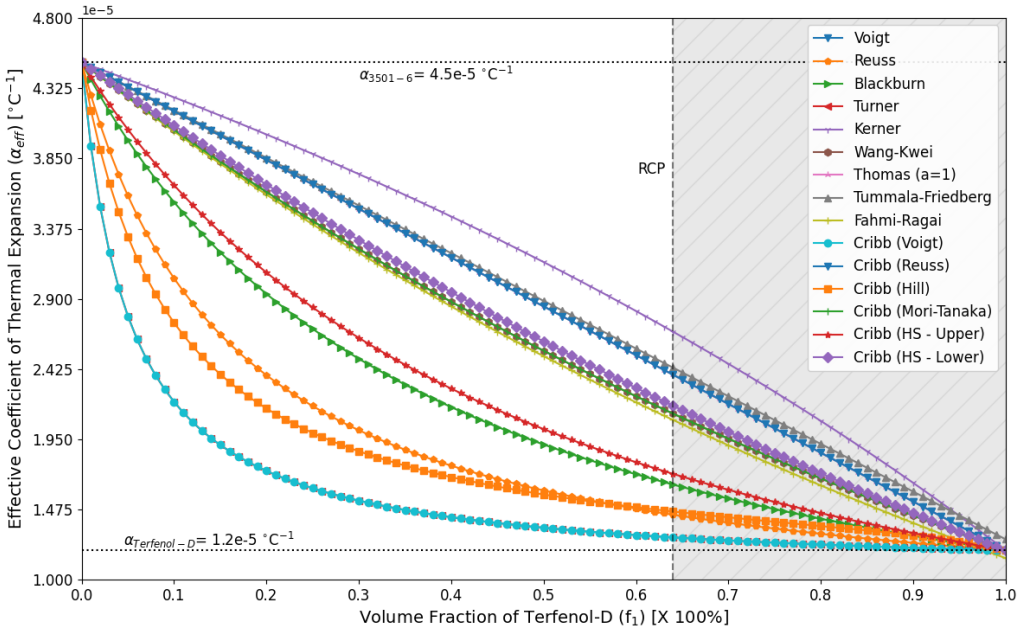


Figure 4.9: Evaluation of all the presented models for the effective coefficient of thermal expansion for various volume fractions of Terfenol-D particles.

4.6 Laminate Analysis with Effective Properties

4.6.1 Single Lamina

The determination of effective properties for a composite material enables the implementation of laminate analysis tools. Homogeneous materials are defined using averages of the properties of the constituent materials, so the complexity associated with a composite is reduced. A particulate lamina was analyzed to examine the effect of the effective properties on the induced stress magnitude. The sensing lamina consisted of Terfenol-D particles embedded in a host matrix of Hexcel 3501-6. The Python script in Appendix D.3 calculated the stress and strain of a single lamina using the classical lamination theory as presented in Section 2.3. The laminae investigated included single plies of HexPly AS4/3501-6, Hexcel 3501-6, bulk Terfenol-D, and particulate composites of Terfenol-D/3501-6. The effective elastic properties discussed in Section 4.4 approximated the values of the elastic modulus, Poisson's ratio, and shear modulus. The volume fraction of particles within the particulate laminae was independently manipulated to examine its effect. The analysis assumed that the orientation of all the laminae was equal with an orientation angle of zero degrees, i.e., the unidirectional fibers of the CFRP aligned with the longitudinal direction of the lamina referred to as the x-direction in Fig. 2.2. There were no mechanical loads applied to the laminae, so the force and moment resultants equaled zero. Instead, the analysis examined the laminae following the curing process of fabrication. The script applied a temperature difference of -120 °C to account for the cool-down period; this process reflected the fabrication method employed for creating laminates of HexPly AS4/3501-6. The stress and strain induced in the laminae, therefore, were caused by thermal effects.

Figure 4.10 illustrates the variations in the stress magnitudes respective to the effective elastic property approximations. The stress for the particulate lamina was equivalent in the longitudinal and transverse directions, Figs. 4.10a and 4.10b, respectively, while the HexPly AS4/3501-6 lamina had differing stress magnitudes due to the difference in longitudinal and transverse coefficients of thermal expansion. The HexPly AS4/3501-6 CFRP prepreg was the only lamina to exhibit positive stress in the longitudinal direction of the ply because of the negative coefficient of thermal expansion in the fiber direction. The bulk Terfenol-D lamina experienced a greater induced compressive stress than a lamina of only Hexcel 3501-6. The difference in stress magnitude agreed with the comparison of the coefficient of thermal expansions for these respective materials in Section 4.5.2. The effective

property approximations demonstrated behavior when calculating stress comparable to that of the elastic properties. The Voigt approximation yielded the greatest stress values in magnitude; contrarily, the Reuss approximation produced the lowest stress magnitudes. The Voigt approximation for the coefficient of thermal expansion predicted the particulate lamina to experience the most compressive stress of any of the approximations. Furthermore, the Voigt approximation predicted the particulate lamina to experience more compressive stress for volume fractions of Terfenol-D greater than approximately 21% than a lamina of only bulk Terfenol-D.

The approximation of the coefficient of thermal expansion was changed from the Voigt approximation in Fig. 4.10 to examine the effect of this parameter. The thermal effects were isolated for study in this analysis by the decision to involve no mechanical loads. When using the Reuss approximation for the coefficient of thermal expansion shown in Fig. 4.11, the magnitudes of the thermal stress for the particulate lamina decreased; therefore, they became less compressive. Only the Voigt elastic property approximation produced stress values between those for the constituent materials. The other four elastic property approximations yielded stress values for feasible volume fractions, i.e., less than the random close packing factor of 64%, greater than that of a pure Hexcel 3501-6 lamina. This result contradicted the expected behavior of the stress magnitudes being more comparable with that of Terfenol-D because the Reuss coefficient of thermal expansion approximation predicted the effective value to be closer to Terfenol-D than Hexcel 3501-6.

The Blackburn approximation for the coefficient of thermal expansion [154] produced estimates close to the median of the examined approximations in Fig. 4.9. Examining this approximation offered insight into how the coefficient of thermal expansion impacted the induced stress state in a particulate lamina as it was not a bound approximation. Figure 4.12 shows that the stress curves for the effective elastic property approximations were similar to those in Fig. 4.11. The magnitudes of the stress values were less for all volume fractions using the Blackburn approximation. The Voigt and Hashin-Shtrikman upper bound approximations appeared near reflections across a line from their starting and ending points. These were the only two estimates to predict the stress of the particulate lamina to lie between its constituents. The other three approximations predicted the stress magnitude to be closer to that of a pure Hexcel 3501-6 lamina.

The extreme coefficient of thermal expansion approximations, i.e., the upper and lower bounds, found in Fig. 4.9 are plotted in Fig. 4.13. The magnitude of the stress was directly related to the value of the effective coefficient of thermal expansion. In the case of the Kerner approxi-

mation, the upper bound in Fig. 4.13a, most of the elastic property approximations yielded stress magnitudes between those of the constituent materials. The stress curve of the Voigt elastic property approximation significantly deviated from the other approximations at nearly 40 MPa greater in magnitude at $f_1 = 64\%$ than a lamina of only bulk Terfenol-D. The Cribb-Voigt coefficient of thermal expansion approximation using the Reuss, Mori-Tanaka, and Hashin-Shtirkman lower bound approximations calculated the stress in the lamina to have almost the same magnitude for Terfenol-D volume fractions between 12% and 45%. The stress for these three formulations was nearly constant in this volume fraction range, which was unobserved with other coefficient of thermal expansion approximations.

4.6.2 Laminate

The classical lamination theory analysis was extended from a single lamina to a laminate. Three laminates were investigated to examine the inclusion of an embedded sensing lamina and a delamination. Figure 4.14 shows the stress distribution for a $[0_4]$ HexPly AS4/3506-1 CFRP laminate. The magnitude of the stress in this laminate was comparable to that of a single lamina of HexPly AS4/3501-6 in the longitudinal stress plots in Section 4.6.1. Because the CFRP prepreg material properties are independent of the volume fraction of Terfenol-D particles, the stress is unchanged as the volume fraction changes. The symmetric, unidirectional laminate also implied that the stress field is uniform across the cross-section of the laminate.

A particulate sensing lamina was embedded into the $[0_4]$ HexPly AS4/3506-1 CFRP laminate in Figs. 4.15 and 4.16. The magnitude of the stress in the CFRP laminae decreased with the introduction of a sensing lamina; the sign also changed such that a compressive stress magnitude was observed. This behavior of compressive stress in the longitudinal direction of a CFRP prepreg was observed for an asymmetric laminate with an unlike lamina, i.e., not a HexPly AS4/3501-6 lamina, embedded at its midplane. The sensing lamina had a higher magnitude of stress than the prepreg lamina when embedded along the midplane. The magnitude of stress was nearly two-thirds less than that observed for a single lamina in Fig. 4.12a. The presence of a lamina made from other materials in Figs. 4.15 and 4.16 showed that the magnitude of the stress in the sensing lamina is affected by the structure of the laminate.

The introduction of a delamination in Fig. 4.16 decreased the magnitude of the stress in the sensing lamina from the case of no delamination present in Fig. 4.15. The properties of the

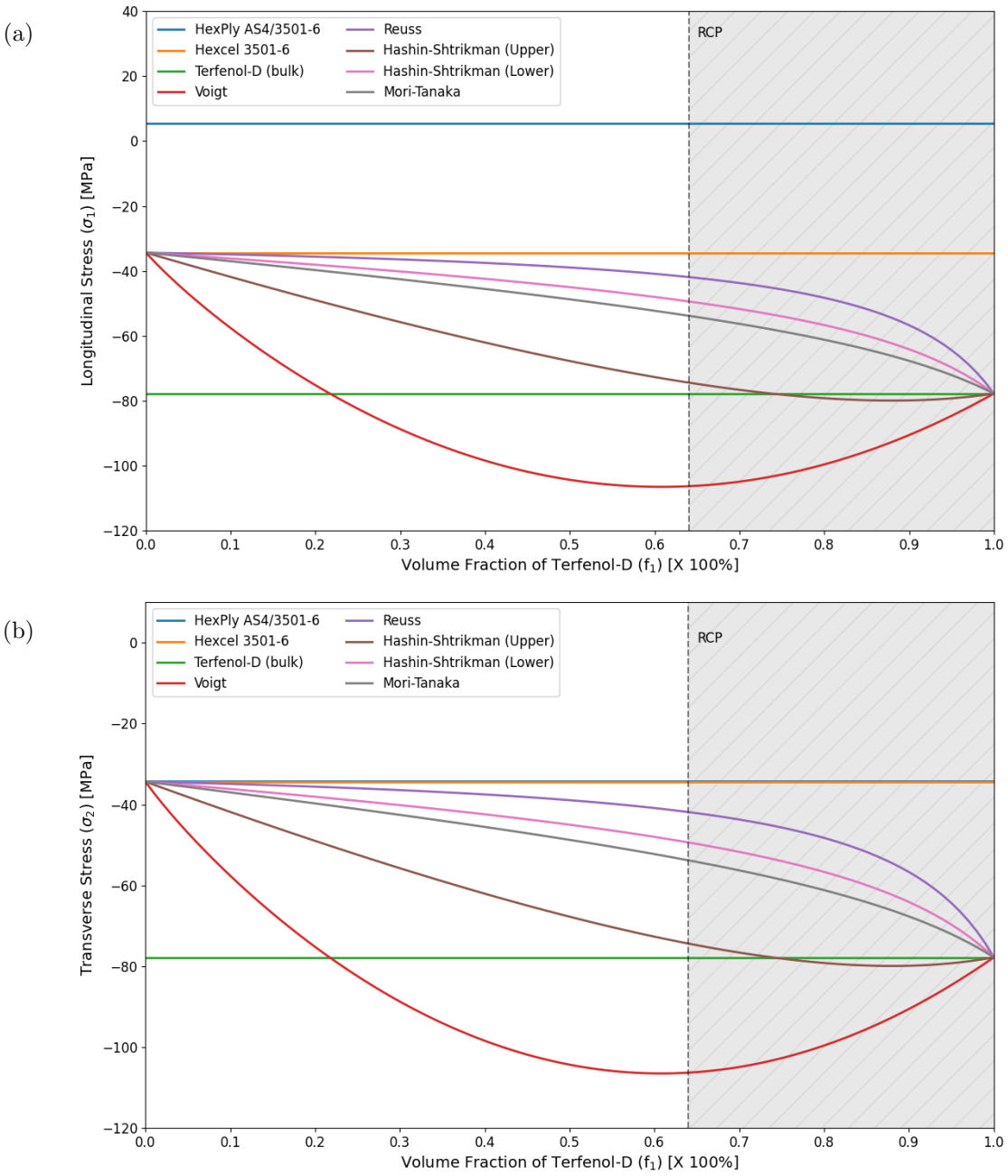


Figure 4.10: Stress for particulate sensing laminae using effective property approximations. The coefficient of thermal expansion was calculated using the Voigt approximation; (a) longitudinal and (b) transverse direction.

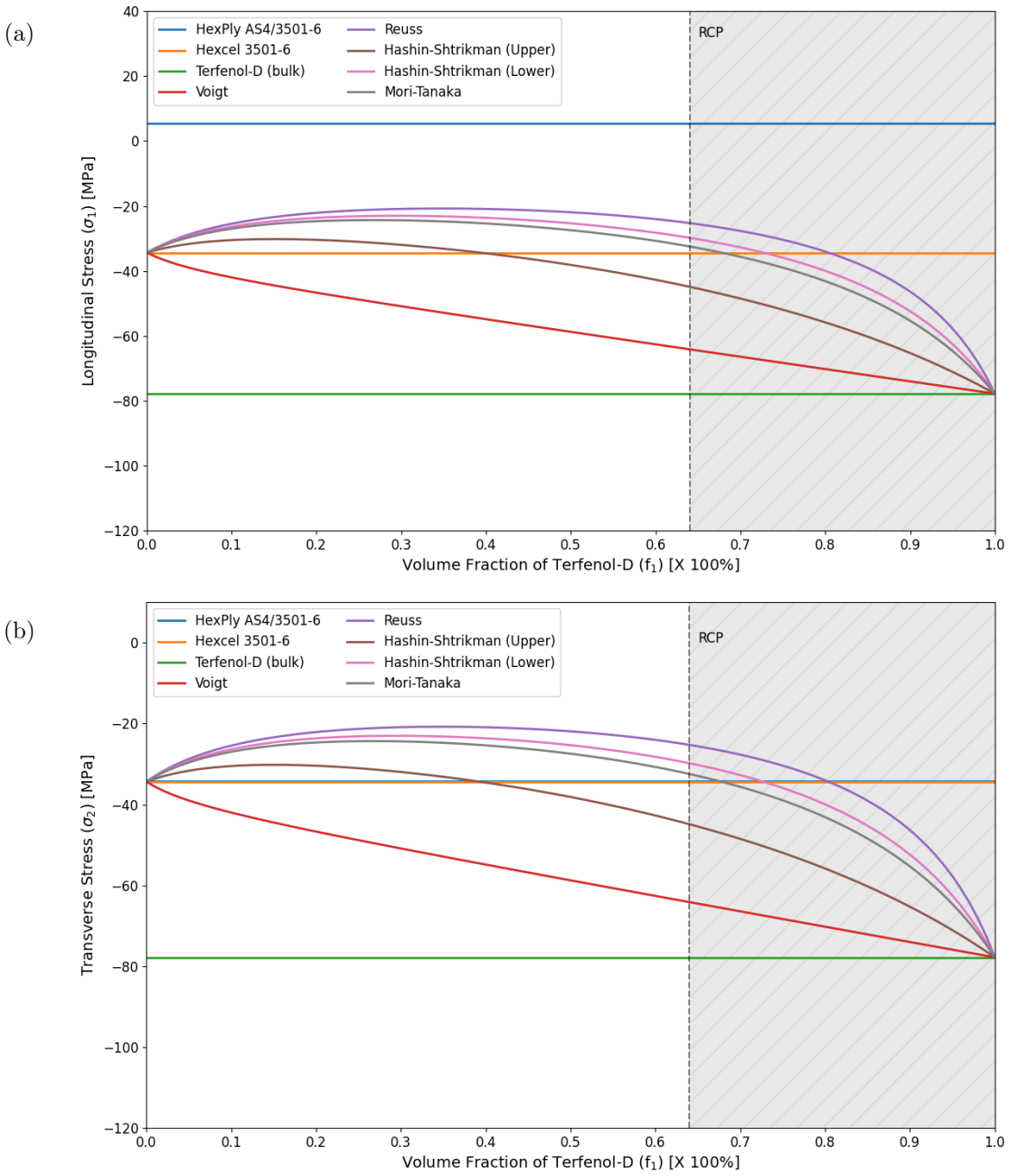


Figure 4.11: Stress for particulate sensing laminae using effective property approximations. The coefficient of thermal expansion was calculated using the Reuss approximation; (a) longitudinal and (b) transverse direction.

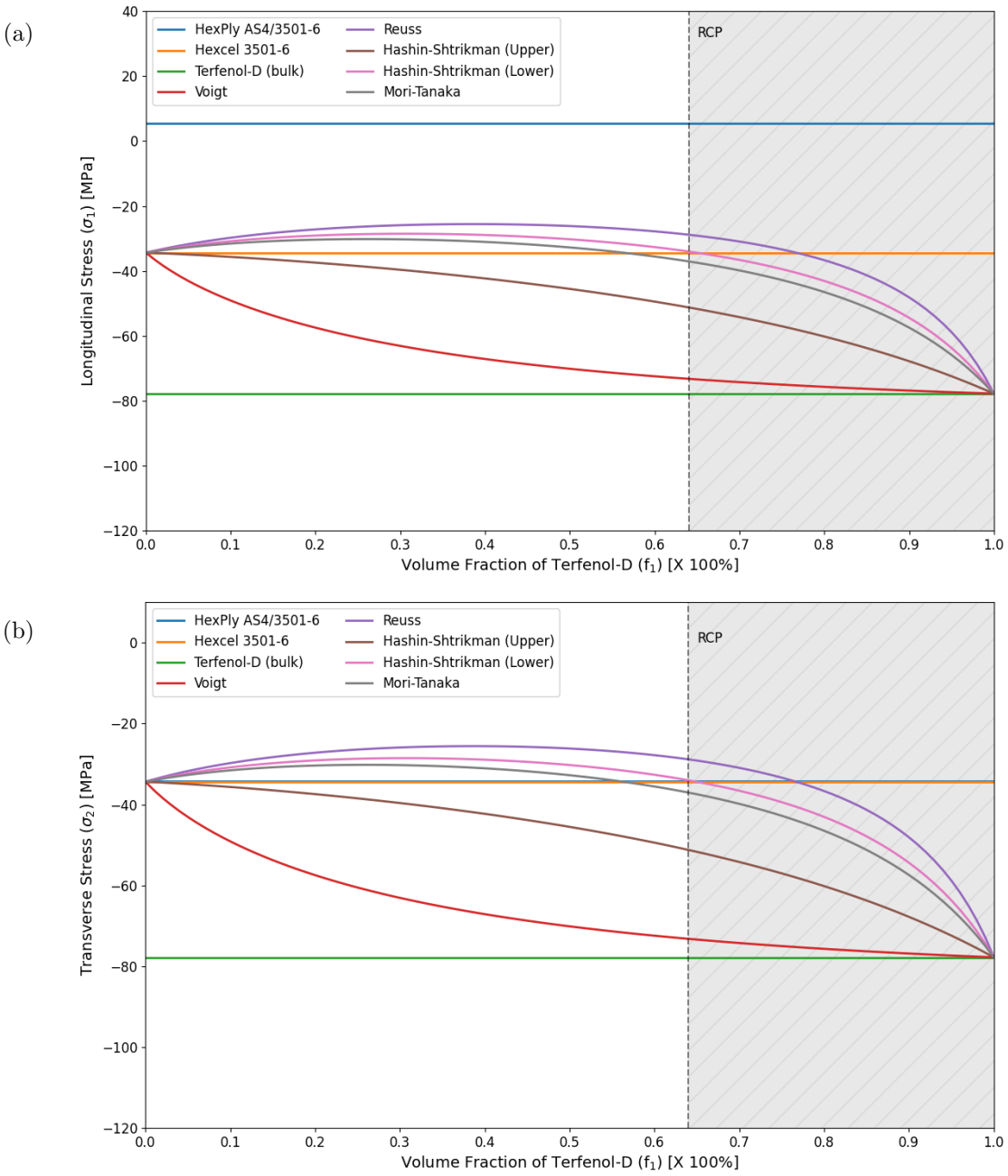


Figure 4.12: Stress for particulate sensing laminae using effective property approximations. The coefficient of thermal expansion was calculated using the Blackburn approximation; (a) longitudinal and (b) transverse direction.

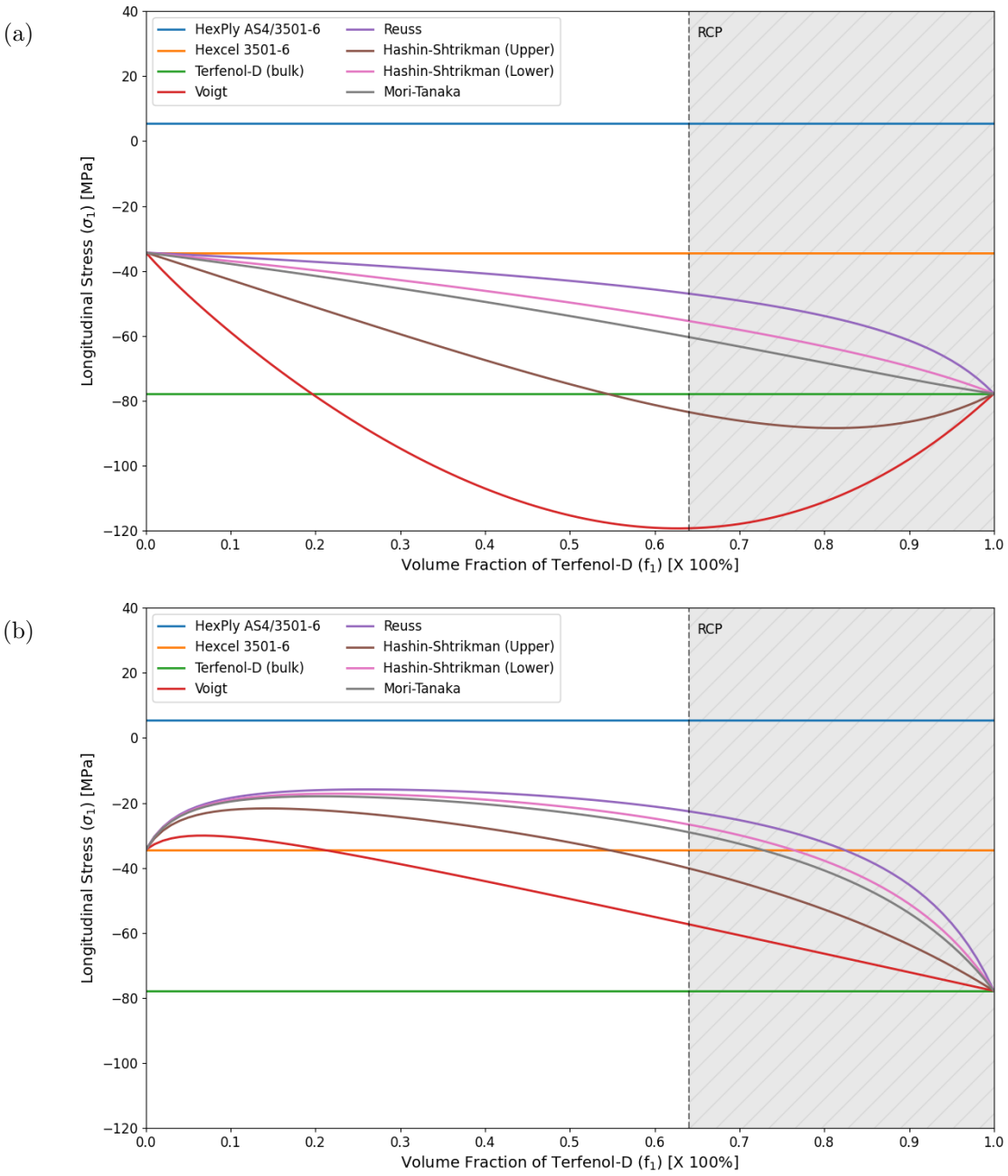


Figure 4.13: Stress for particulate sensing laminae using effective property approximations. The coefficient of thermal expansion was calculated using the extreme bound approximations per Fig. 4.9; (a) Kerner (upper bound) and (b) Cribb-Voigt (lower bound) approximations.

Table 4.6: Delamination material properties and information used in the classical lamination theory.

Property	Value
Longitudinal modulus (E)	100 Pa
Shear modulus (G)	0 GPa
Poisson's ratio (ν)	0
Thermal coefficient (α)	0 °C^{-1}
Thickness (t)	0.01 mm

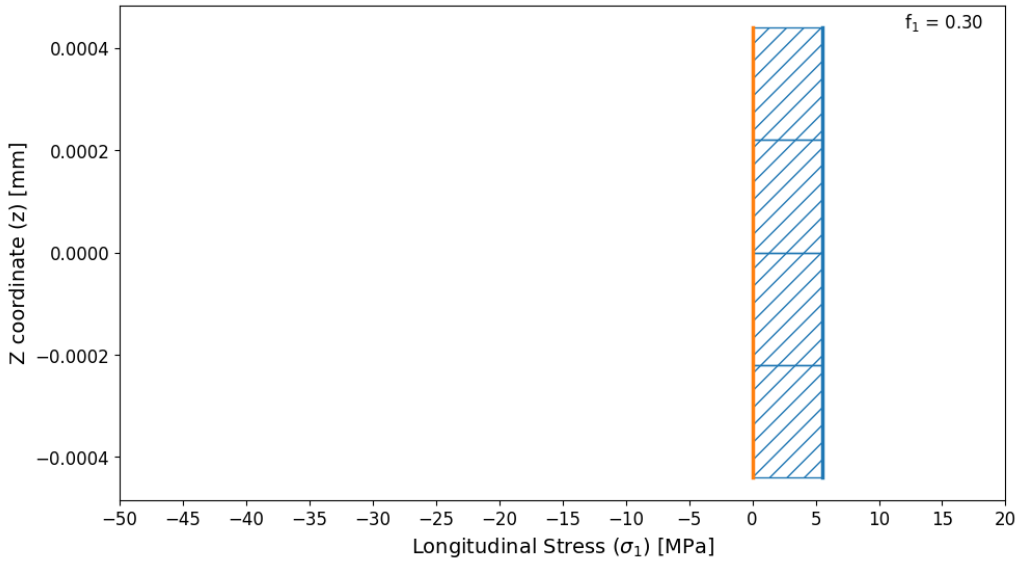


Figure 4.14: Stress distribution in the longitudinal direction for a [0_4] HexPly AS4/3506-1 CFRP laminate.

delamination used for modeling are given in Tab. 4.6. The stress distribution was no longer constant throughout the cross-section of each lamina. The top prepreg plies (above the delamination) of the laminate had greater stress magnitudes than those below the delamination. The stress at the delamination was negligible due to the difference in material properties of the adjacent laminae. A significant change in stress occurred at the delamination interfaces, and a difference in nearly 22 MPa existed between the interfaces at the top and bottom of the delamination. While the magnitude of the stress in the sensing lamina decreased with the introduction of a delamination, the stress difference across the delamination suggested the presence of damage and encouraged the evolution of damage at this location.

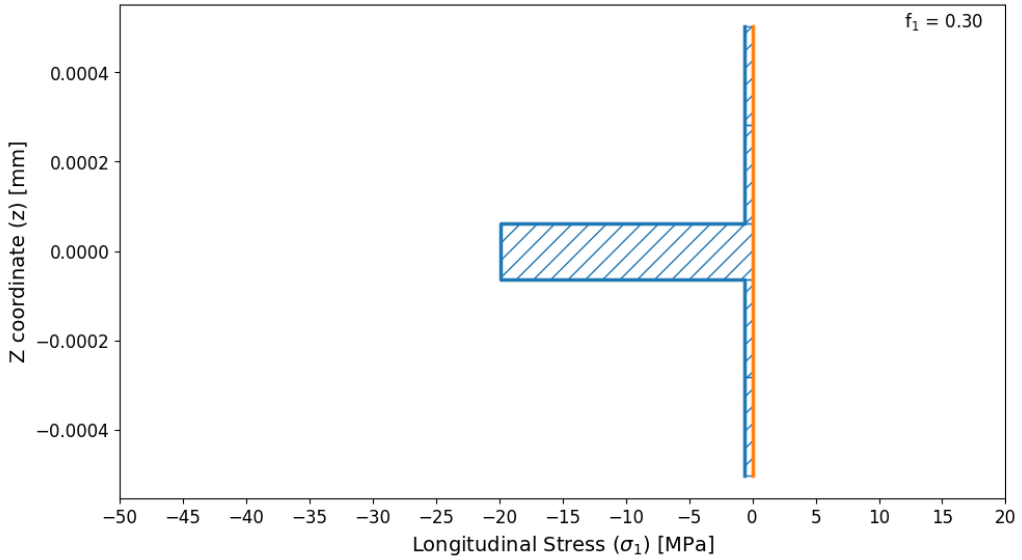


Figure 4.15: Stress distribution in the longitudinal direction for a $[0_2/m/0_2]$ HexPly AS4/3506-1 CFRP laminate embedded with a Terfenol-D/3501-6 sensing lamina. The effective properties of the sensing lamina were approximated using the Voigt estimate, and the coefficient of thermal expansion with the Blackburn estimate. The volume fraction of particles equals 30%.

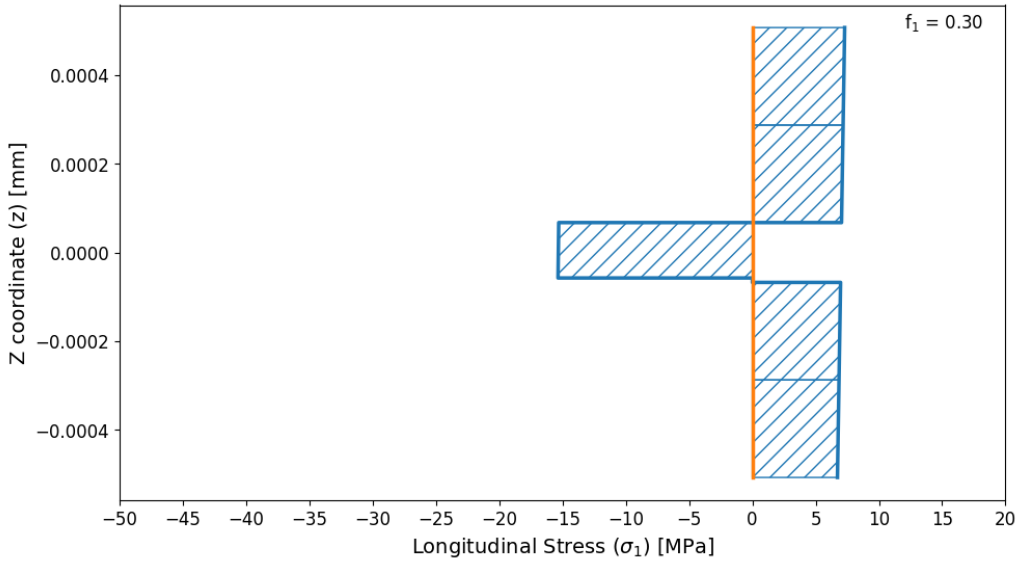


Figure 4.16: Stress distribution in the longitudinal direction for a $[0_2/m/d/0_2]$ HexPly AS4/3506-1 CFRP laminate embedded with a Terfenol-D/3501-6 sensing lamina and delamination. The effective properties of the sensing lamina were approximated using the Voigt estimate, and the coefficient of thermal expansion with the Blackburn estimate. The volume fraction of particles equals 30%.

4.7 Conclusion

The effective properties of a Terfenol-D particulate lamina were investigated in this work to examine how the inclusion of the host (matrix) material properties affect the mechanical properties of the sensing lamina. The volume fraction of spherical Terfenol-D particles embedded in a host matrix of Hexcel 3501-6 was varied from 0 to 100%; though, a maximum random close packing factor of 64% was taken as the realistic maximum volume fraction that could be achieved based on the discussed fabrication methods. Effective properties were necessary to combine the property values of the constituent materials of the particulate composite. The bulk properties of Terfenol-D, specifically the nonlinear relationships derived from the stress-strain behavior, could be coupled with the calculated effective properties.

The effective elastic, bulk, and shear moduli displayed an expected behavior of a bounding problem previously discussed in [140] where the Voigt and Reuss approximations corresponded to the upper and lower bounds, respectively, of possible values for the modulus. The effective Poisson's ratio failed to adhere to the Voigt-Reuss bounds despite previous literature stating that these approximations were appropriate for the Poisson's ratio. The particulate lamina was less incompressible than bulk Terfenol-D, and the Poisson's ratio was typically closer to that of the matrix material. The region of possible values of the effective coefficient of thermal expansion was the most expansive of all the effective properties as 15 approximations were explored.

Using the effective coefficient of thermal expansion, the induced thermal stress in a particulate lamina was examined. The value of the coefficient of thermal expansion was concluded to impact the induced stress significantly. Nearly a 60 MPa difference was obtained using the upper (Kerner) and lower (Cribb-Voigt) approximations of the coefficient of thermal expansion with the effective elastic properties calculated using the Voigt approximation for a volume fraction of $f_1 = 30\%$.. The induced thermal stress in the sensing lamina was negative in magnitude, meaning that the sensing lamina was in compression. The analysis of two CFRP laminates with an embedded sensing lamina verified this finding. The compressive stress was due to the contraction of the matrix material under a $-120°$C temperature change; this result agreed with previous findings.

Chapter 5

Application of Machine Learning

5.1 Motivation for Implementation

The application of machine learning has increased in recent times because of its ability to solve a wide variety of problems. A machine learning algorithm operates by learning from previous experiences to make predictions about new, never-before-seen information. Statistically based, machine learning functions act like a black box where the algorithm manipulates the parameters involved in training internally; the process of analyzing data is solely contained within the algorithm. Previous work has investigated the application of machine learning for structural health monitoring and nondestructive evaluation, and promising results were obtained (Section 2.8). The discussed NDE method investigated in this work, i.e., embedded magnetostrictive particles for damage sensing, has yet to be coupled with machine learning.

Several complexities were identified through experimental work for detecting damage within a composite material using embedded magnetostrictive particles. The spread of the magnetostrictive particles affects the induced sensing voltage [6, 7, 57, 150]. An accumulation of Terfenol-D particles within the sensing lamina increases the localized sensing voltage resulting in false readings. Alternative distribution techniques were investigated to create a more even spread and limit any distribution effects on the voltage [57, 127]. Delaminations were intentionally introduced into experimental laminates by embedding film or tape at specified locations. Ideally, the sensing voltage changes when the coil apparatus passes over a region containing a delamination when implementing this NDE method. However, the sensing voltages obtained for various experimental laminates failed to distinguish any

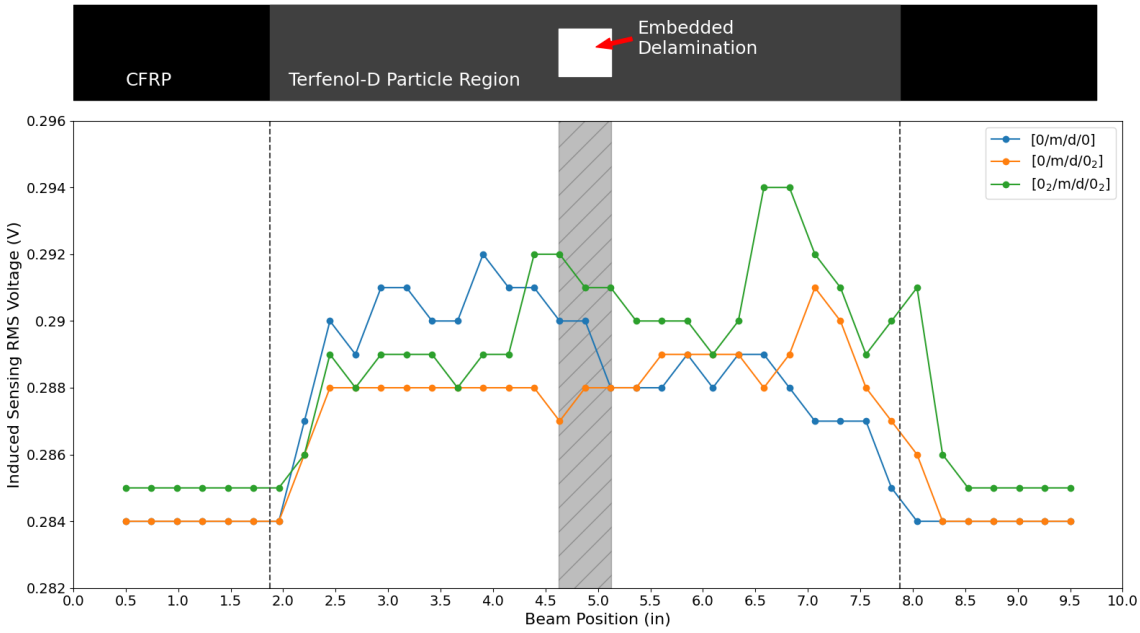


Figure 5.1: Sensing voltage for three composite beams with embedded Terfenol-D particles. The embedded delamination region is shown by the gray region. (recreated from data taken from [57])

effects in the voltages caused by the embedded delaminations along with the number of plies in the laminate. Figure 5.1 illustrates the variations in the sensing voltage for three experimental CFRP laminate beams with different stacking sequences [57]. A half-inch square delamination was embedded in the laminate during fabrication at 4.625 inches from the side of the beam denoted by the zero inch mark. Changes in the sensing voltage can be observed between the regions with and without Terfenol-D particles; the particles were spread between approximately 1.875 and 7.875 inches from the zero mark of the beam. The response of the regions with embedded Terfenol-D particles was not consistent across the three presented beams, and the behavior at and around the delamination was unique for each beam. By visual inspection of Fig. 5.1, the location, much less the existence, of a delamination in the beams could not be determined.

The use of machine learning with the discussed NDE method could predict the presence of delaminations better than the visible review of a sensing voltage plot. A machine learning algorithm combines various input parameters to make predictions based on trends identified during the learning process. In this application, information about the data/sample point would coincide with the prediction making of an algorithm to determine the presence of a delamination. The variation in the sensing voltage for undamaged regions and those with a delamination would be accounted for within

the black-box approach of the algorithms, and relationships could be formulated. The classification ability of machine learning algorithms would allow a user to input raw, experimental data and yield a prediction of whether a delamination or other damage precursor existed at a specific location.

5.2 Experimental Data

Data was necessary to train and test the prediction performance of a machine learning process with the embedded magnetostrictive particle NDE method. Previous literature presented experimental data for a static beam containing embedded Terfenol-D particles for damage sensing. Intentional delaminations were incorporated in the experiments to examine the impact of damage on the induced sensing voltage. A limited set of experimental data limits the exploration of machine learning for this NDE method, so analytical data was sought not only to compare against experimental data but as a means for generating data points. The analytical model based on the Euler-Bernoulli model presented by Krishnamurthy et al. [53] was investigated in Chapter 3, but the model failed to converge to predict the stress in the magnetostrictive material; ergo, the induced voltage could not be calculated. Without an analytical model, the investigation of a machine learning approach for predicting the presence of a delamination was continued utilizing only experimental data.

After perusing the literature, experimental data for this NDE method was presented in four works [6, 57, 127, 155]. A total of 1,380 experimental data points were either obtained from tables or extracted from plots using the GRABIT function [121] in MATLAB. Each data point contained the following information, if attainable:

- Source data taken from
- Fiber material
- Matrix material
- Stacking sequence
- Number of plies **above** Terfenol-D lamina
- Number of plies **below** Terfenol-D lamina
- The presence of Terfenol-D (1 – yes, 0 – no)
- Peak sensing voltage V_{peak}

- RMS sensing voltage V_{RMS}
- The presence of an embedded delamination (1 – yes, 0 – no)

In some cases, the RMS voltage was not presented or assumed to not be given; the peak voltage was assumed to be presented. The RMS voltage was calculated as

$$V_{RMS} = \frac{V_{peak}}{\sqrt{2}} \tag{5.1}$$

The breakdown of the data is presented in Tab. 5.1 and contains statistical information about the RMS voltage for each literature source. The number of points with and without a delamination is presented in Fig. 5.2 and shows that a lack of experimental delamination data exists. In total, only 70 (5.07%) data points contained delaminations. For a machine learning problem, especially one attempting to classify the presence of a delamination, a target label occurring in only 5% of the total dataset raised concern that the algorithm would be unable to properly "learn" sufficient information about a delamination state. Examining the mean RMS sensing voltages, the experimental setups were found to be inconsistent based on the discrepancies in the magnitudes of these mean values. The data found in [57] and [6] was most comparable based on the statistics of the RMS voltage.

A new parameter was defined to create a dataset with more comparable voltage values. An adjusted RMS voltage value was defined for each data point as

$$V_{RMS,adjusted} = \frac{V_{RMS}}{\overline{V}_{RMS,\text{ no Terfenol-D}}} \tag{5.2}$$

where the average RMS voltage values with no Terfenol-D term ($\overline{V}_{RMS,\text{ no Terfenol-D}}$) in the denominator was calculated for each CFRP beam. The distribution of the adjusted RMS voltages displayed

Table 5.1: Statistics of the extracted experimental data from literature. The experimental setups varied evident by the value of the mean RMS sensing voltage.

Source	Data Points	Delaminations	RMS Sensing Voltage (V_{RMS}) [V]		
			Mean	Standard Deviation	Variance
Rudd [57]	1216	42	0.288	3.437e-03	1.181e-05
Myers et al. [6]	95	2	0.216	2.822e-03	7.966e-06
Chen & Anjanappa [127]	37	16	0.530	1.159e-01	1.343e-02
Currie et al. [155]	32	10	2.097	4.527e-02	2.049e-03

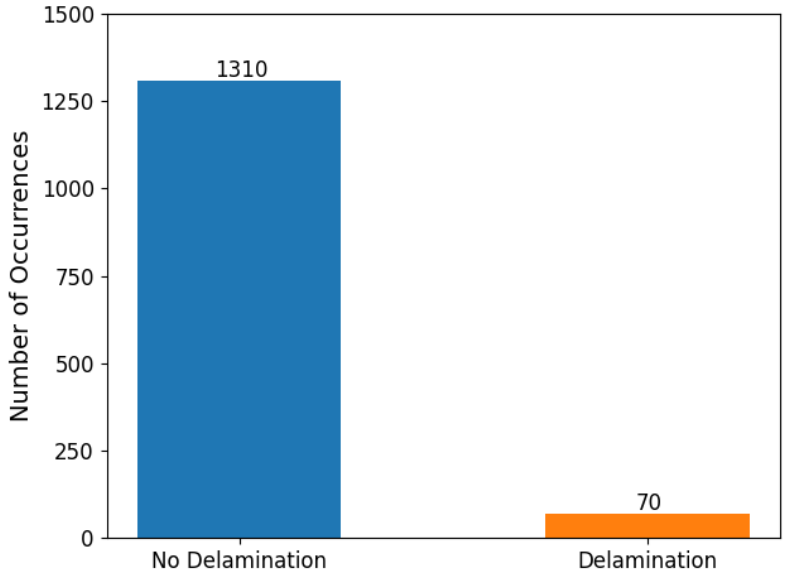


Figure 5.2: Illustration in the lack of delamination data points extracted from literature.

in Fig. 5.3 shows that the adjusted RMS voltage parameter failed to yield comparable values for the [127] data when compared to the other three datasets. Upon review of the data extracted from [127], a limited amount of information was given about the laminates. Terfenol-D was either embedded across the entire length of the laminate or not present at all. A thick film of magnetostrictive material was embedded in the laminates rather than particles. The adjusted parameter results led to the removal of the [127] data from the overall dataset to avoid introducing undesired outliers to the algorithm that could alter the prediction ability.

The data from [155] was also excluded from consideration because the stacking sequence was unknown for the experimental laminates. The effect of the stacking sequence was not distinguishable in the plots of the induced sensing voltage, such as Fig. 5.1, but [7] suggested that the ability to discern the ply variation could be beneficial. Both the datasets extracted from [57] and [6] contained information about the stacking sequence. Figure 5.4 displays the distribution of the adjusted RMS voltages for both of these datasets. The greater mean adjusted voltage the [57] dataset was attributed to the greater number of data points than [6].

Only 3.34% of the data points in the combined dataset using data from [57] and [6] contained a delamination. The two delamination points from [6] were recorded using a CFRP beam with a stacking sequence of $[(0/90)_3/m/(0/90)_3]$ whereas [57] employed unidirectional CFRP laminates in

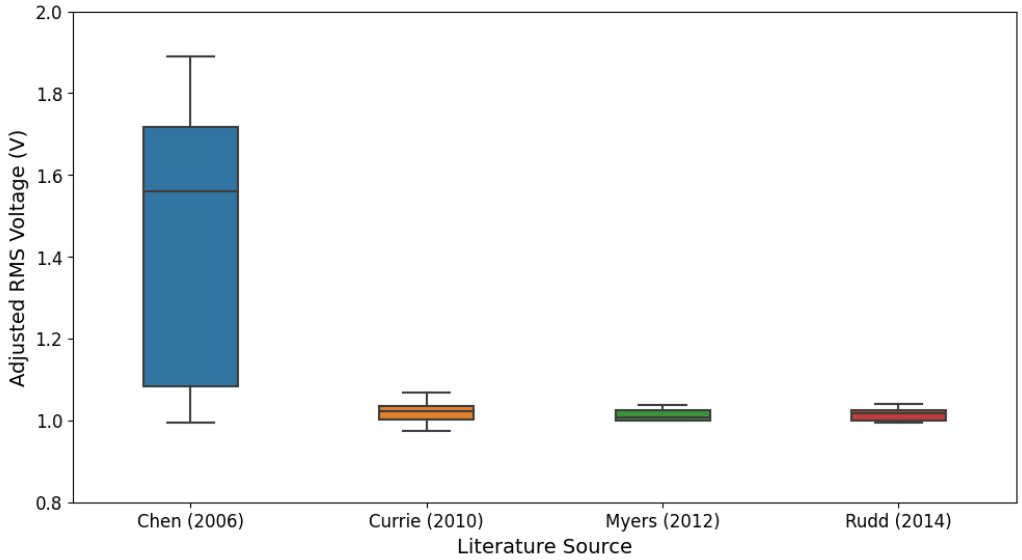


Figure 5.3: The adjusted RMS voltage parameter resulted in more comparable data values for three of the four extraction sources.

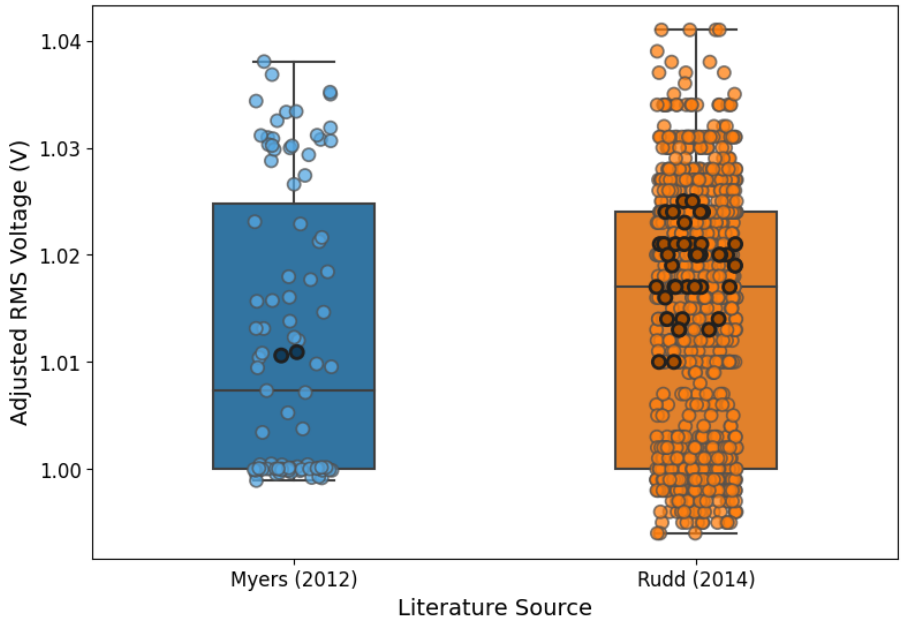


Figure 5.4: The adjusted RMS voltage data extracted from [6] and [57]. The markers imposed over the box plot are the experimental data points, and the darkened markers are data points obtained for a region containing a delamination.

all the conducted experiments. Only unidirectional CFRP laminates were considered for the finalized machine learning dataset to simplify the features in the dataset. Because the orientation angle of a unidirectional laminate is equal for all laminae, only an integer value of the number of plies above and below the Terfenol-D layer was necessary to characterize the composite. A stacking sequence with mixed orientation angles could be represented this way, but the effects of the orientation angle on the composite properties would not be appropriately considered. The materials used in [57] were also the same for all the experimental beams; therefore, there was no need to incorporate the elastic properties of the materials for this investigation.

5.2.1 Machine Learning Dataset

The data extracted from [57] contained the most number of total data points $(1,216)$ as well as the most points with an embedded delamination (42). For a classification problem, though, an introduced dataset with only 3.45% of a target label would likely yield poor results. An initial machine learning algorithm was trained and tested with the whole $1,216$ data point set to examine this hypothesis. The accuracy of the model was found to range from $98-100\%$, and immediate concerns were raised with accuracies of this magnitude. The initial algorithm was trained with a training set containing a majority of samples not containing an embedded delamination. The algorithm predicted no delaminations for almost every data point in the testing set during testing, even if a delamination existed for the respective testing sample point. The lack of delamination data meant that randomly dividing the total dataset into training and testing sets could yield sets with no delamination samples. While machine learning has been associated with "big data," the data must represent the investigated problem. It was determined that a subset of the data extracted from [57] would be best suited for the machine learning investigation in this work.

A total of 98 data points were chosen to be used in the machine learning dataset, and the extracted information for each data point is presented in Appendix B. The subset contained all 42 delamination points to maximize the exposure of an algorithm to one class of the target labels. Table 5.2 shows that just over 40% of the subset points contained a delamination, which was a significantly greater ratio than that of the original, complete dataset extracted from [57]. Slightly more points not containing an embedded delamination were selected for the subset because these points offered more variance in information about the CFRP beam. Some of the undamaged points contained no Terfenol-D. The algorithm was exposed to various voltage magnitudes because the regions not

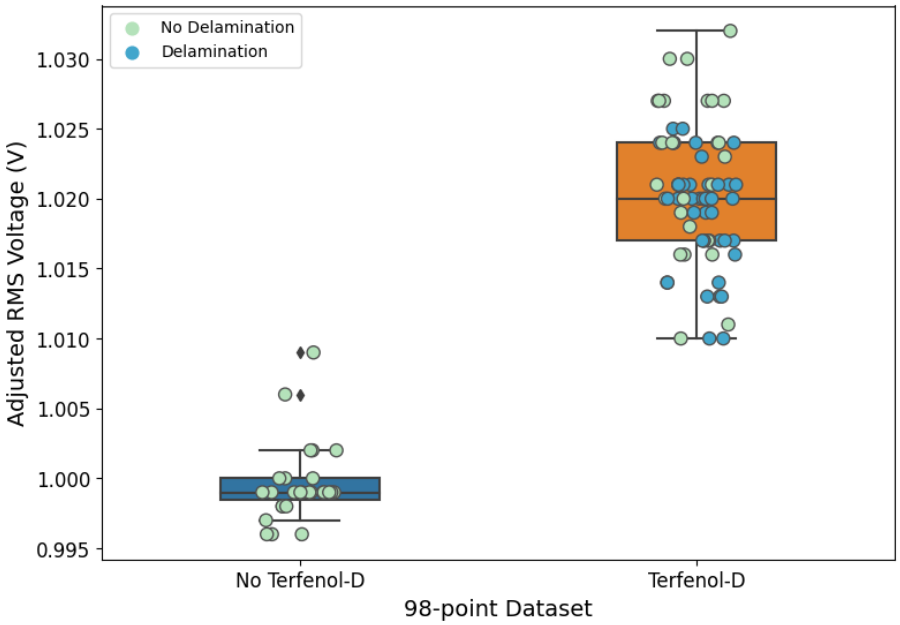


Figure 5.5: The effect of the presence of Terfenol-D on the adjusted RMS voltage was 0.020 V less on average that that of the data points with Terfenol-D particles embedded.

Table 5.2: Breakdown of the 98-point data subset taken from [57] for the machine learning algorithms.

	With Terfenol-D	Without Terfenol-D	Total
With Delamination	42	0	42 (42.86%)
Without Delamination	29	27	56 (57.14%)
Total	71 (72.45%)	27 (27.55%)	98 (100%)

containing Terfenol-D had lower induced sensing voltages shown in Fig. 5.5. Nearly 25% of the data points did not contain Terfenol-D offering an algorithm opportunity to correlate the effect of Terfenol-D on the adjusted voltage parameter. The 56 data points not containing a delamination were selected at random using a Python script. This 98-point dataset was unchanged for all the tested machine learning algorithms to avoid introducing variability caused by the data.

5.2.1.1 Sources of Error

The experimental data employed in this work was previously found in work completed by Rudd et al. [150] and Rudd [57]. The particle distribution led to variations in the induced sensing voltages. Accumulations of particles in one region resulted in increased voltages in that region due

to the localized increase in magnetostriction; likewise, regions void of Terfenol-D particles had lower induced sensing voltages. A trenching method was suggested in [57] to reduce the error caused by insufficient spreading, and the presence of localized accumulations of particles was reduced. The testing apparatus increased variability as the diameter of the actuating and sensing coils had to be larger than the width and thickness of the experimental beams. While most of the generated magnetic field was directed towards the beams, the strength of the actuating magnetic field decayed with distance. The larger the radius of the coils, the smaller the magnitude of the magnetic field intensity that reached the embedded Terfenol-D particles per the inverse square law:

$$H_{intensity} \propto \frac{1}{distance^2} \tag{5.3}$$

Alternative sensing methods investigated in [127] aimed at reducing this effect through the use of a single-sided apparatus. A film was also used by Chen et al. [127] to reduce the distribution effects.

5.3 Tested Machine Learning Algorithms

The machine learning application investigated in this work aimed to detect the presence of an embedded delamination in a CFRP using experimental data for the embedded magnetostrictive particle NDE method. This classification problem consisted of two classes: a delamination present (1) or a delamination not present (0). A select number of machine learning algorithms are applicable for classification work. The algorithms selected for this work were implemented for SHM/NDE applications in previous works. The algorithms were also found within the scikit-learn package [37] for Python. The same processing and implementation methods were followed for each algorithm examined to reduce variability in the obtained results.

5.3.1 Preparation of Data

The same dataset was employed to eliminate any inconsistencies in the information introduced to the algorithms. All of the known information for the data in Appendix B was not introduced to the algorithms. The available information, called features, that could be introduced into the algorithms included knowledge about the constituent materials (the fiber and matrix materials), the stacking sequence, the presence of Terfenol-D, sensing voltage values, and information about the

presence of a delamination. Prediction errors can arise when a large number of features are included in an algorithm.

The curse of dimensionality refers to the situation when a dataset contains too many features in a classification problem [156]. Overfitting of the model and classification error are possible results of this phenomenon. For example, 26 unique stacking sequences were present in the 98-point dataset, and each of these stacking sequences needed to be translated to a numerical value for input into the algorithms. The orientation angle of each ply could be introduced as an independent feature, but that would require the definition of 16 features for each laminate because the thickest laminate had 16 plies: ($[0_7/m/d/0_7]$), where m referred to the location of the magnetostrictive material and d the location of the embedded delamination. Quantifying each ply of a laminate in addition to including the presence of a delamination and Terfnol-D using Boolean logic would result in an undesired multiplicity of information being supplied to an algorithm. Two latent features were introduced to reduce the dimensionality of the features by reducing the stacking sequences to the number of plies above and below the magnetostrictive lamina. This method allowed the stacking sequence to be reduced from a possible 16 terms to only two (four if including the Boolean logic to represent the presence of Terfenol-D and a delamination). Integer quantities represented the stacking sequence rather than the composite nomenclature, thus creating comparable values between data points. The laminates used in [57] were made of the same fiber and matrix material allowing for the exclusion of these two features for this initial machine learning application. A total of five features were used in this two-class problem:

- CFRP plies above the Terfenol-D sensing lamina (integer)
- CFRP plies below the Terfenol-D sensing lamina (integer)
- Presence of Terfenol-D (binary: 1 or 0)
- Adjusted RMS Voltage, $V_{RMS,adjusted}$ (float)
- Presence of delamination (binary: 1 or 0)

Reducing the number of features involved in this machine learning investigation resulted in a limited scope of application. The elimination of the material properties did not allow for the explicit inclusion of their values. The obtained sensing voltage was related to the composite's material properties, so the trained algorithms were best suited for composites with HexPly AS4/3501-6 carbon fiber laminae. The use of the adjusted voltage parameter aimed at reducing the effect of

the dismissal of the consideration of the specific materials. The data points were also limited to basic information about a laminate's structure. Variations in the orientation angles of laminae were not included. Only unidirectional laminates were included in the implemented dataset. Applying new data to the trained algorithms for prediction should follow the nature of the laminates involved in training. Previous literature showed that significant error could occur when unfamiliar composites are introduced to a machine learning algorithm during testing [107, 108].

5.3.2 Introduction of Employed Machine Learning Algorithms

A total of eight different machine learning algorithm formulations were employed in this work for classification. A brief synopsis of each of the methods is included in this section to discuss the mathematical approach utilized in each formulation. Variations can be made to the methods to manipulate the relationships and functions implemented during the classification process.

5.3.2.1 Support Vector Machines

Support vector machines (SVM) create hyperplanes to divide data into clusters. They are commonly implemented for classification problems of various dimensions and can support a high dimensional space, i.e., many features, by manipulating the formulation. The optimal hyperplane maximizes the distance to the nearest data points regardless of class during the training phase. This distance, called the margin, relates to the error of the algorithm; the larger the value of the margin, the lower the value of the classification error. An SVM seeks to optimize the location of the hyperplane during training.

The formulation of an SVM assumes that a set of training vectors, $x_i \in \mathbb{R}^p,\ i = 1, \ldots, n$, falls in two classes. A target vector is defined as $y \in \{1, -1\}^n$. The overall goal of the method is to find the direction $w \in \mathbb{R}^p$ and intercept $b \in \mathbb{R}$ such that the prediction of the constraint equation, $\text{sign}\left(w^T \phi\left(x\right) + b\right)$, yields the most number of correct values when compared to the target vector. The general formulation of a SVM seeks to solve the following minimization problem:

$$\min_{w,b} \frac{1}{2}||w||^2 \tag{5.4}$$

$$\textit{subject to } y_i\left(w \cdot x + b\right) - 1 \geq 0, i = 1, \ldots, m$$

where w is the direction of the hyperplane, b is an intercept value of the hyperplane, and m is the

number of data points [157]. Alternatively, Eq. (5.4) can be expressed using vector notation [37]:

$$\min_{w,b} \frac{1}{2} w^T w \tag{5.5}$$
$$subject\ to\ y_i \left(w^T \phi(x) + b \right) \geq 1, i = 1, \ldots, m$$

where $\phi(x)$ is a function of x. Variations to the general SVM method incorporate a cost (or penalty) function to the algorithm. A support vector classifier (SVC) allows data samples to lie a distance away from their correct margin value. The C-SVC formulation includes the distance ζ_i paired with a penalty parameter C which issues a value when the sample is misclassified during training [37, 157]:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \tag{5.6}$$
$$subject\ to\ y_i \left(w^T \phi(x) + b \right) \geq 1 - \zeta_i, \quad \zeta_i \geq 0,\ i = 1, \ldots, m$$

The value of ζ_i can be manipulated to either loosen or tighten the value of the constraint equation. The value of C determines the importance of the allowable distance for a sample point from the hyperplane. A large value of the penalty term will not allow outliers to exist in the solution and will only be applicable for linearly separable data. A small value of C, on the other hand, loosens the constraint equation. In the case of $C = 0$, the hyperplane often fails to classify the data. A kernel function $K(x_i, x_j)$ is introduced using the dual problem for the primal:

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \tag{5.7}$$
$$subject\ to\ y^T \alpha = 0, \quad 0 \leq \alpha_i \leq C,\ i = 1, \ldots, m$$

where e is a vector containing only ones, Q is an $n \times n$ positive, semidefinite matrix, and α_i are called the dual coefficients. The matrix Q contains the kernel such that $Q_{ij} = y_i y_j K(x_i, x_j)$. The kernel function allows for the method to classify nonlinear separable data and can take the following

forms:

$$\text{Linear kernel:} \quad K(x_i, x_j) = x_i \cdot x_j = \langle x, x' \rangle \tag{5.8}$$

$$\text{Polynomial kernel:} \quad K(x_i, x_j) = (x_i \cdot x_j + c)^d = (\gamma \langle x, x' \rangle + c)^d, \tag{5.9}$$

$$\text{Radial Basis Function (RBF) kernel:} \quad K(x_i, x_j) = \exp\left(-\gamma ||x_i - x_j||^2\right) \tag{5.10}$$

$$= \exp\left(-\gamma ||x - x'||^2\right) \tag{5.11}$$

$$\text{Sigmoid kernel:} \quad K(x_i, x_j) = \tanh\left(\gamma \langle x, x' \rangle + c\right) \tag{5.12}$$

where c, d, and γ are constants supplied to the algorithm during the kernel function declaration. The predicted value is found as the sign of

$$\sum_{i \in SV} y_i \alpha_i K(x_i, x_j) + b \tag{5.13}$$

where SV signifies the data points within the margin of the hyperplane. The decision surface is related to the order of the kernel function. A similar method to the C-SVC is the ν-SVC formulation, which places bounds on the margin error. A linear SVC implements the formulation of the primal, Eq. (5.6), with modification to the loss term:

$$\min_{w,b} = \frac{1}{2} w^T w + C \sum_{i=1} \max\left(0, y_i \left(w^T \phi(x) + b\right)\right) \tag{5.14}$$

This type of SVM can only be applied to linearly separable data, so only the linear kernel can be used in this algorithm formulation.

5.3.2.2 Discriminant Analysis

The discriminant analysis method is defined by the order of the decision surface for the problem. The surfaces can be linear (LDA) or quadratic (QDA) and can model multiclass problems. The method involves a probabilistic model that uses the conditional distribution of the data associated with each class, $P(X \mid y = k)$. Using Bayes' rule, the predictions for each training sample point $x \in \mathbb{R}^d$ are determined as

$$P(y = k \mid x) = \frac{P(x \mid y = k) P(y = k)}{P(x)} = \frac{P(x \mid y = k) P(y = k)}{\sum_l P(x \mid y = l) \cdot P(y = l)} \tag{5.15}$$

Bayes' rule, as given in Eq. (5.15), mathematically states that the probability of $y = k$ occurring given the observation of sample x equals the conditional probability of x occurring when $y = k$ times the probability of observing $y = k$ divided by the probability of observing x. The class k is selected which maximizes the posterior probability. According to [37], a multivariate Gaussian distribution models $P(x \mid y)$ in an LDA and QDA; the density of a multivariate Gaussian distribution takes the form

$$P(x \mid y = k) = \frac{1}{(2\pi)^{d/2} |\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_k)^t \Sigma_k^{-1} (x - \mu_k)\right) \tag{5.16}$$

where d equals the number of features involved in the model. A discriminant analysis model aims to maximize the posterior probability. For a QDA, the log of the posterior is given as

$$\log P(y = k \mid x) = \log P(x \mid y = k) + \log P(y = k) + Cst \tag{5.17}$$

$$= -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x - \mu_k)^t \Sigma_k^{-1} (x - \mu_k) + \log P(y = k) + Cst \tag{5.18}$$

where the Cst term represents the $P(x)$ and other constants from the Gaussian distribution. The LDA assumes that the Gaussian distributions for each class have the same covariance matrix: $\sum_k = \sum$. The log of the posterior may then be expressed as

$$\log P(y = k|x) = -\frac{1}{2}(x - \mu_k)^t \Sigma^{-1} (x - \mu_k) + \log P(y = k) + Cst \tag{5.19}$$

The LDA formulation of the log-posterior uses the Mahalanobis distance, which describes the distance a sample point x is from the mean μ_k. Therefore, an LDA model determines the class of a sample to be that which has the closet mean according to the Mahalanobis distance.

5.3.2.3 Gradient Descent

The gradient descent algorithm is an optimization method that utilizes a hyperplane to define regions much like SVMs. For a classification problem, the training samples are defined as $(x_i, y_i), \ldots, (x_n, y_n)$ where $x_i \in \mathbb{R}^m$ and $y_i \in \{-1, 1\}$. The scoring function (hyperplane) involved in gradient descent is modeled by the linear relationship

$$f(x) = w^T x + b \tag{5.20}$$

where, as with the SVM method, $w \in \mathbb{R}^m$ and $b \in \mathbb{R}$ [37]. Predictions for a binary classification problem are determined based on the sign of Eq. (5.20). The gradient descent optimization problem seeks to minimize the training error to determine the values of the constants w and b. The training error E employs a loss function which differentiates the gradient descent method from other methods:

$$E\left(w,b\right) = \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, f\left(x_i\right)\right) + \alpha R\left(w\right) \tag{5.21}$$

where L is the loss function related to the fit of the model to the samples, R is a penalty term (also known as a regularization term), and α is a non-negative hyperparameter that controls the influence of the penalty on the error. Common loss functions include:

$$\text{Hinge:} \quad L\left(y_i, f\left(x_i\right)\right) = \max\left(0, 1 - y_i f\left(x_i\right)\right) \tag{5.22}$$

$$\text{Perceptron:} \quad L\left(y_i, f\left(x_i\right)\right) = \max\left(0, -y_i f\left(x_i\right)\right) \tag{5.23}$$

$$\text{Modified Huber:} \quad L\left(y_i, f\left(x_i\right)\right) = \begin{cases} \max\left(0, 1 - y_i f\left(x_i\right)\right)^2, & y_i f\left(x_i\right) > 1 \\ -4 y_i f(x_i), & \text{otherwise} \end{cases} \tag{5.24}$$

$$\text{Log:} \quad L\left(y_i, f\left(x_i\right)\right) = \log\left(1 + \exp\left(-y_i f\left(x_i\right)\right)\right) \tag{5.25}$$

The regularization term is also user-defined and can take the form of

$$\text{L2 norm:} \quad R\left(w\right) = \frac{1}{2}\sum_{j=1}^{m} w_j^2 = ||w||_2^2 \tag{5.26}$$

$$\text{L1 norm:} \quad R\left(w\right) = \sum_{j=1}^{m} |w_j| \tag{5.27}$$

$$\text{Elastic Net:} \quad R\left(w\right) = \frac{\rho}{2}\sum_{j=1}^{n} w_j^2 + \left(1 - \rho\right)\sum_{j=1}^{m} |w_j| \tag{5.28}$$

In machine learning, stochastic gradient descent (SGD) implements the gradient descent method for each sample. The algorithm calculates the error for each sample point and updates the values of the model parameters w and b using a learning rate η.

5.3.2.4 Nearest Neighbor

When applying the nearest neighbor (NN) algorithm to classification problems, the algorithm seeks to find the minimal distance for each sample to a number of clusters representing the targets. This type of algorithm differs from other machine learning methods as they do not result in a parameter-based model; they are considered a type of instance-based or non-generalized learning [37]. The algorithm stores the training samples for future predictions. A common nearest neighbor method is the k-nearest neighbors (k-NN) formulation where the user defines an integer value for the number of nearest neighbors k for a specific sample point. Internally, a k-NN algorithm examines each sample point and determines its classification based on the majority of the nearest neighbors around the point.

Distance calculations are necessary for determining the nearest points to a specific sample. In a brute force approach, the distance between a specific sample and the rest of the complete dataset is calculated. As the number of samples in the dataset increases, this method can prove computationally intensive. Another approach for implementing a k-NN algorithm is the use of a tree-based structure for determining the distances. This approach examines the distance relationships between values to reduce the number of calculations. Two variations of the tree method include the KD and ball tree approaches [158,159]. The KD tree approach is quick because it reduces the number of distance calculations by partitioning data, so multidimensional calculations are not required. Low dimensional problems can be quickly solved using a KD tree approach, whereas the ball tree approach is better suited for higher-dimensional problems. Data is divided into hyper-spheres using the ball tree approach, and the results of this approach can be dependent on the structure of the data. The use of hyper-spheres combined with a triangular distance calculation allows for higher-dimensional data to be modeled.

5.3.2.5 Naïve Bayes

Naïve Bayes models are beneficial for datasets with a small number of samples because they independently estimate each feature distribution. This formulation reduces dimensionality and the possibility of problems arising from the curse of dimensionality [37]. The basis of the naïve Bayes

method lies in the probability of an outcome given an instance which Bayes' rule defines by

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)\, P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)} \tag{5.29}$$

The naïve conditional independence assumption,

$$P(x_i \mid y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i \mid y) \tag{5.30}$$

is assumed for all i. Substituting this assumption into Eq. (5.29) yields a simplified expression derived from Bayes' rule

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)} \tag{5.31}$$

Under the assumption that the probability for each instance is equal, $P(x_1, \ldots, x_n)$ would be a constant value. Therefore, Eq. (5.31) can be expressed as proportional terms to result in the naïve Bayes probability model

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y) \tag{5.32}$$

The classification rule for the naïve Bayes method employs a *maximum a posteriori* (MAP) estimation to determine the class label for a sample point; the function is as follows:

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i \mid y) \tag{5.33}$$

Variations in naïve Bayes classifiers are made through altering the probability distribution of $P(x_i \mid y)$. For example, a Gaussian naïve Bayes algorithm assumes the likelihood of the features to follow a Gaussian distribution:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{5.34}$$

where the standard deviation σ_y and mean μ_y are estimated for the population, i.e., the dataset.

5.3.2.6 Decision Tree

The decision tree method partitions data using conditional statements. Decision rules are applied at each "leaf" of the tree to divide data, and data division occurs until all leaves contain a

specified number of samples. Mathematically, a series of nodes m are employed; data is analyzed at each node and then sent away from the node after being divided [37]. A set of training vectors $x_i \in \mathbb{R}^n,\ i = 1, \ldots, l$ with a corresponding label vector $y \in \mathbb{R}^l$ are passed into the initial node of a decision tree. Let the combination of these vectors be denoted by the set Q. The data is divided by the candidate split $\theta = (j, t_m)$ which contains the feature j and a threshold t_m. At the node, the dataset is divided into two subsets, $Q_{left}(\theta)$ and $Q_{right}(\theta)$:

$$Q_{left}(\theta) = (x, y) \mid x_j \leq t_m \tag{5.35}$$

$$Q_{right}(\theta) = Q \setminus Q_{left}(\theta) \tag{5.36}$$

where Eq. (5.36) states that $Q_{right}(\theta)$ contains the remaining data in Q after removing the items in $Q_{left}(\theta)$. The newly defined sets are then passed to subsequent nodes where the process is repeated. The impurity that occurs at a node concerns the error in making the data split. The parameters determined for the candidate split are found from

$$G(Q, \theta) = \frac{n_{left}}{N_m} H(Q_{left}(\theta)) + \frac{n_{right}}{N_m} H(Q_{right}(\theta)) \tag{5.37}$$

$$\theta^* = \arg\min_{\theta} G(Q, \theta) \tag{5.38}$$

where $H()$ is an impurity function. For a classification at a region R_m with a number of observations N_m, the impurity function may take the form of

$$\text{Gini:} \quad H(X_m) = \sum_k p_{mk}(1 - p_{mk}) \tag{5.39}$$

$$\text{Entropy:} \quad H(X_m) = -\sum_k p_{mk} \log(p_{mk}) \tag{5.40}$$

$$\text{Misclassification:} \quad H(X_m) = 1 - \max(p_{mk}) \tag{5.41}$$

where

$$p_{mk} = \frac{1}{N_m \sum_{x_i \in R_m} I(y_i = k)} \tag{5.42}$$

Decision trees can act as a white box where the user can define conditional relationships to be used in the model. The method results in a graphical illustration of the logic involved in classifying the dataset; however, the data partition process structure can differ with each application of the

model. The logic employed during classification can over-complicate a model and is not suited for a generalized problem.

5.3.2.7 Ensemble Methods

Ensemble methods implement several models and then combine their results to improve generalizability over the implementation of a single model [37]. The predictions of individual models are averaged using an averaging scheme. One type of ensemble method is a forest of random trees. In this approach, a set of individual decision trees are trained using the supplied data. The predictions of the individual trees are averaged together to yield the prediction of the "forest". In a random forest classifier model, only a subset of the given dataset is introduced to each decision tree. The data drawn from the dataset is replaced, so the dataset does not decrease in size for selection. Randomness increases further in an extremely randomized tree model by randomizing the threshold t_m values for each feature in the set.

5.3.2.8 Neural Network

A nonlinear relationship can be determined using the black-box formulation of a neural network model, specifically a multilayer perception (MLP) algorithm. This approach combines a series of neurons (or perceptrons) in layers as illustrated in Fig. 2.7 to determine relationships between inputs and outputs. A perceptron linearly combines its inputs $\{x_i|x_1, x_2, ..., x_m\}$ with weights w_i and a bias value b such that

$$y = f\left(w_1x_1 + w_2x_2 + ... + w_mx_m + b\right) \tag{5.43}$$

The function f is called the activation function, and it determines the value of y outputted to the subsequent layer of perceptrons. For binary problems, the activation function determines the class

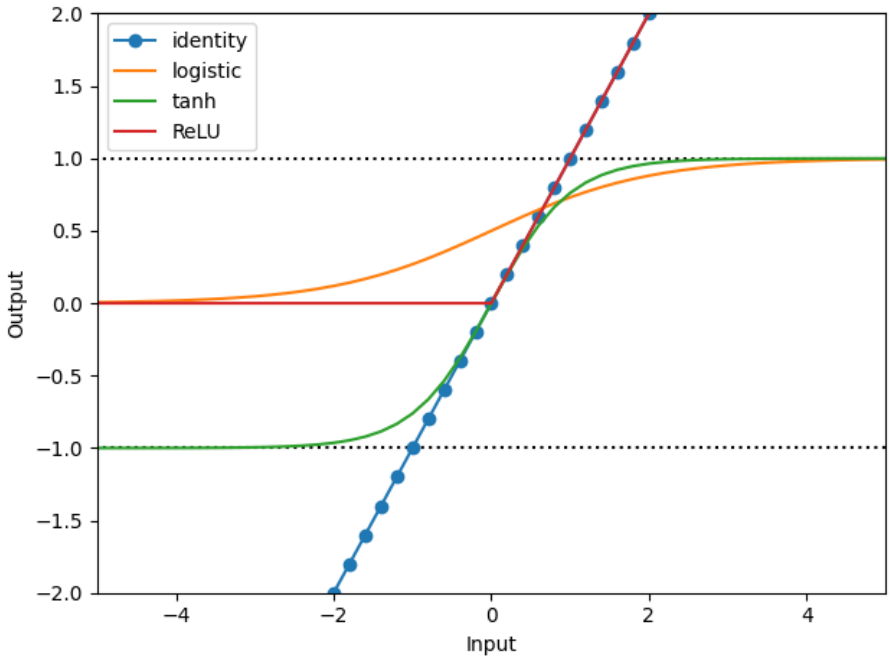


Figure 5.6: The range of the activation functions vary based on the magnitude and sign of the input.

by calculating y to equal either one or zero. Common activation functions include:

$$\text{Identity:} \quad f(x) = x \tag{5.44}$$

$$\text{Logistic/sigmoid:} \quad f(x) = \frac{1}{1 + \exp(-x)} \tag{5.45}$$

$$\text{Hyperbolic tangent (tanh):} \quad f(x) = \tanh(x) \tag{5.46}$$

$$\text{Rectified linear unit (ReLU):} \quad f(x) = \max(0, x) \tag{5.47}$$

Figure 5.6 illustrates the range an activation function maps the given domain. The obtained range can span from negative to positive infinity depending on the selected function.

A loss function applies a penalty during training to improve the performance of the model. A cross-entropy function is employed for a classification problem where [37]

$$Loss(\hat{y}, y, W) = -y \ln \hat{y} - (1 - y) \ln(1 - \hat{y}) + \alpha ||W||_2^2, \quad \alpha > 0 \tag{5.48}$$

During training, the MLP improves its performance via loss minimization by updating the weight and bias values. After calculating the new parameters, they are propagated to the previous layer of

perceptions via a method known as back-propagation. The model seeks to minimize the loss function below a specified threshold or until a set number of iterations are completed; if the threshold has not been met, convergence will not occur.

5.4 Testing Procedure

The algorithms discussed in Section 5.3.2 were investigated in this work using the defined dataset in Appendix B. A general Python script (see Appendix D.4) was written for use with the various algorithms. The scikit-learn package [37] was selected for use in the script because of its machine learning capabilities. The package contains a variety of supervised and unsupervised learning algorithms along with preprocessing and analysis tools.

Before implementing an algorithm within the script, a *pipeline* was defined to create a uniform process for all the prediction experiments. The *pipeline* allowed for multiple estimators/processes to be linked together such that data flowed from one to the next. The employed *pipeline* contained two estimators: a preprocessing step and the machine learning model. A standardization process was first employed in the *pipeline* to create a normalized distribution of data for each feature. A z-score technique was applied to the data of each feature such that the scaled sample had a value of

$$x_{scaled} = \frac{x - \mu}{\sigma} \tag{5.49}$$

where μ is the mean of the values for the specific feature, and σ is the standard deviation. Scaling the data used in an algorithm reduced the likelihood of an outlier sample or a feature set with a large variance to impact the performance of the algorithm significantly. The scaled data was then passed through the *pipeline* and inputted into the specified machine learning algorithm with user-defined parameters.

The investigated algorithms required a set of features and target labels/classes for classification. As previously mentioned, a two-class problem was of interest to classify whether or not a sample contained an embedded delamination. The presence of a delamination was extracted from the dataset for use as the target label. Four features were inputted to each model to predict the class of the sample. The number of plies above and below the Terfenol-D sensing lamina, the presence of Terfenol-D, and the adjusted RMS voltage were standardized independently and used as the feature set.

Two methods were employed to analyze the performance of the investigated machine learning algorithms. Using a looping scheme of 30 iterations, each of the $1,249$ machine learning models was trained and tested with a random split of test-train data. The data was split such that 90% of the data was used during training, and the remaining data was used for testing/prediction making. Four metrics were calculated by comparing the predicted results from the trained models to the actual classification. The accuracy, precision, recall, and F1-score were measured for each of the 30 trained models. These metrics were defined using four indicators: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). TP and TN corresponded to the actual number of each class, i.e., the number of points predicted correctly with and without a delamination. A FP represented the classification of a true negative as a positive. Similarly, the number of FNs represented the number of true positives predicted as a negative case. If the presence of a delamination was considered the *negative* case, a high number of FPs would raise concern as the model was predicting damaged states as undamaged. This type of prediction could lead to catastrophic failure and prove the model to be inefficient at predicting the presence of damage. Using these indicators, the four performance metrics were defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.50}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.51}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.52}$$

$$F1 - Score = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{5.53}$$

The second portion of the experimentation methodology incorporated a cross-validation scheme in the training of the algorithms. Repeatedly training a model with the same data can cause overfitting to occur where the model makes predictions closely resembling the behavior of the training set. A validation set can be defined to evaluate the performance of the model before testing; however, implementing a validation step reduces the number of sample points available for training and testing. Cross-validation schemes seek to mitigate the effect of overfitting by partitioning the overall training set into subsets. The need for a validation set is also eliminated based on the methodology of the cross-validation scheme.

For a k-fold cross-validation scheme, the data designated for training is divided into k folds

as illustrated in Fig. 2.9 where each fold contains an equal number of samples. The model is then trained using $k-1$ folds, and the remaining fold is used for testing. If the number of folds k equals the number of samples in the dataset, n, the concept of the leave-one-out (LOO) cross-validation scheme occurs. The length of the training sets differs between a k-fold and LOO approach. For the LOO scheme, the number of training sets employed for each fold is $n-1$, whereas a k-fold scheme uses $(k-1)\,n/k$. A LOO cross-validation scheme has been found to yield results similar to only using one training dataset because there is only a single sample point difference. A cross-validation scheme attempts to reduce the variance in the predictions by looping the training-testing scheme over the entire training dataset. The algorithms were tested using a k-fold cross-validation scheme for $k = 2$ to 98 where the $k = 98$ scheme resembled the leave-one-out approach. The accuracy for each partition was recorded to assess the effect of k and determine the most accurate number of folds for each model.

5.5 Results of Tested Machine Learning Algorithms

Variations of the eight algorithms were made by manipulating internal parameters. For instance, the number of nearest neighbors was varied for the k-NN algorithms. The mathematical formulation was also manipulated for each of the eight algorithms explored to yield an experiment containing a total of $1,249$ models. Appendix C contains a list of the machine learning algorithm variations employed.

5.5.1 Single Train-Test Data Partition Experiments

Using a 90/10 training-testing data split, the results from the $37,470$ single partition experiments (30 unique runs of each of the $1,249$ models) were examined using the metrics of accuracy, precision, recall, and F1-score as given in Section 5.4. Table 5.3 contains statistics for the accuracy of the eight algorithms examined. The mean accuracy of the models was approximately 70%. With an average standard deviation of 13.23%, the accuracy of the single split experiments exhibited a wide distribution. The maximum accuracy score for half of the algorithms was over 95%, whereas the minimum scores for half of the algorithms were below 40%. The distribution of the results suggested that the employed single partition did not yield datasets capable of representing the breadth of the classification problem. Using the randomized split, the testing or training set could have contained

Table 5.3: Accuracy statistics for the eight algorithms examined using 30 variations of each model with a single data partition for testing and training.

Algorithm	Mean	Standard Deviation	Maximum	Minimum
Naïve Bayes	74.33%	14.53%	100.00%	30.00%
Discriminant Analysis	73.50%	12.61%	100.00%	55.00%
Ensemble Method	72.50%	11.77%	90.00%	45.00%
Neural Network	71.76%	13.21%	95.90%	42.77%
Support Vector Machine	71.56%	13.51%	97.78%	44.44%
Decision Tree	70.67%	12.36%	90.00%	40.00%
Stochastic Gradient Descent	68.40%	14.32%	92.00%	32.00%
Nearest Neighbor	67.85%	13.55%	93.50%	39.12%

significantly more samples of one of the two classes. This distribution would sway the performance of the model to predict a greater number of false positives or false negatives.

5.5.2 K-fold Cross-Validation Experiments

5.5.2.1 General Algorithm Performance

The cross-validation scheme was implemented to introduce more data to each model to reduce the overall variance [37]. The spread of the results was significantly reduced from those of the single partition experiments, as shown in Tab. 5.4. The average standard deviation of the accuracy scores decreased by nearly 84% to an average value of 2.08%. The mean accuracy of the SVM and MLP algorithms was relatively unaffected by using a cross-validation scheme. The minimum and maximum accuracy scores for all the algorithms were comparable and much closer using a cross-validation scheme than a single split with a fixed ratio. The magnitudes of the mean accuracies were comparable between the single split and the cross-validation schemes at approximately 70%.

Comparing Tabs. 5.3 and 5.4, the decision tree, gradient descent and k-NN algorithms were found to perform the worst as an algorithm as a whole. Only one decision tree model was examined, and the poor performance illustrated a shortcoming of the algorithm. This algorithm has difficulty making generalized predictions for a problem, so the cross-validation scheme challenged the algorithm with varying training data. The algorithm created a decision process based on the given features in the training set. Any variations in the testing set would inhibit the ability of the trained decision tree to make an accurate prediction; therefore, a decision tree was determined to

Table 5.4: Accuracy statistics based on the algorithm employed for all the cross-validation experiments.

Algorithm	Mean	Standard Deviation	Maximum	Minimum
Support Vector Machine	72.13%	1.71%	75.59%	64.33%
Neural Network	71.53%	2.10%	75.10%	62.11%
Ensemble Method	70.92%	1.72%	74.69%	63.79%
Naïve Baye	70.54%	0.55%	72.32%	69.32%
Discriminant Analysis	70.41%	2.82%	74.48%	56.12%
Decision Tree	68.96%	1.90%	76.53%	61.30%
Stochastic Gradient Descent	67.33%	4.08%	77.21%	58.12%
Nearest Neighbor	66.73%	1.75%	70.17%	59.76%

not be an effective model formulation for this problem.

The cross-validation experiments were concerned with the mean accuracy of each respective fold. The leave-one-out scheme, which uses each sample to test the performance of the model, was compared against the mean accuracy for various numbers of folds using a parametric study. As the number of folds increased, the mean accuracy score for the respective fold was found to converge to the LOO accuracy value as shown in Fig. 5.7. In the first 30 folds, the mean fold accuracy differed the most from the LOO accuracy but eventually converged to the LOO mean. The term *efficient fold* was defined for use in this work as the number of folds less than 30 that corresponded to the smallest difference between the LOO scheme and fold mean accuracy. Figure 5.8 shows that $k = 17$ and $k = 26$ visually appeared the closest to the LOO mean for the SVM-SVC-sigmoid model. The fold corresponding to $k = 26$ was the most efficient fold with a percent difference of 1.65%.

This fold value was found to be greater than the commonly accepted values of $k = 10$ or $k = 20$ [37]. Examining the results of the efficient fold number for the examined models in Fig. 5.9 found that a large number of models yielded the best results for a fold range of $k = 25 - 29$; however, a majority of the evaluated models had efficient fold numbers less than the suggested $k = 20$. A trade-off was observed in both Figs. 5.7 and 5.8 between the mean accuracy and variance. Though greater mean accuracy values than that achieved with a LOO scheme were obtained, the variance in the accuracy scores for each partition of a respective fold scheme increased with the number of folds. Computational expenses also directly increased with the number of folds; the computational time, though, was negligible in this work due to the size of the dataset. A k-fold cross-validation scheme with a fold number less than the number of data points, i.e., a LOO scheme, yielded acceptable

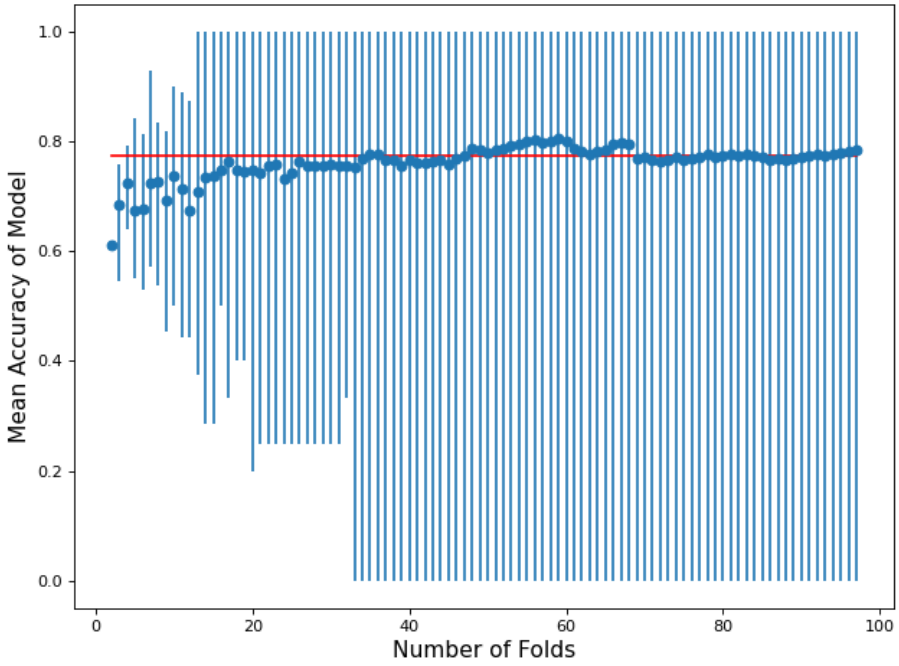


Figure 5.7: The mean accuracy for each fold of the SVC model utilizing a sigmoid function. The leave-one-out accuracy (77.55%) is shown by the red line. The error bars show that the spread of the accuracy increased as the number of folds increased.

accuracy scores.

The performance indicators discussed in Section 5.4 were visualized through confusion matrices. Figure 5.10 shows a confusion matrix for the SVM-SVC-sigmoid model with a $k = 26$ k-fold cross-validation scheme. The diagonals of the confusion matrix are related. The diagonal quadrants going from top-to-bottom and left-to-right show the correct predictions, whereas the other diagonal contains the incorrect predictions. The top-left square shows that 34 samples were correctly predicted as not having a delamination; likewise, 35 samples were correctly predicted to have a delamination. A total of 29 samples ($\approx 30\%$) were incorrectly predicted. More samples were classified as containing a delamination when the actual sample point did not have a delamination than vice versa. Only 7% of the samples that contained a delamination were classified as not containing a delamination; this was the lowest percentage for any of the classification quadrants by this model, offering promise to the applicability of the specific model.

A majority of the investigated models had the lowest prediction percentage for the cross-validation experiments in the bottom left quadrant of their confusion matrix; consider this quadrant

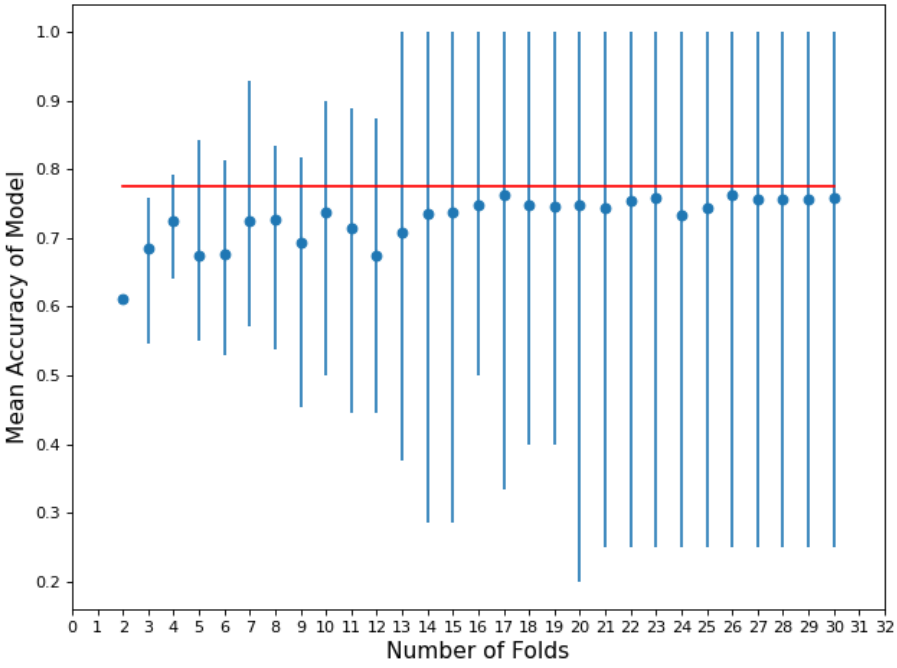


Figure 5.8: The mean accuracy for the first 30 folds of the SVC model utilizing a sigmoid function. The leave-one-out accuracy (77.55%) is shown by the red line.

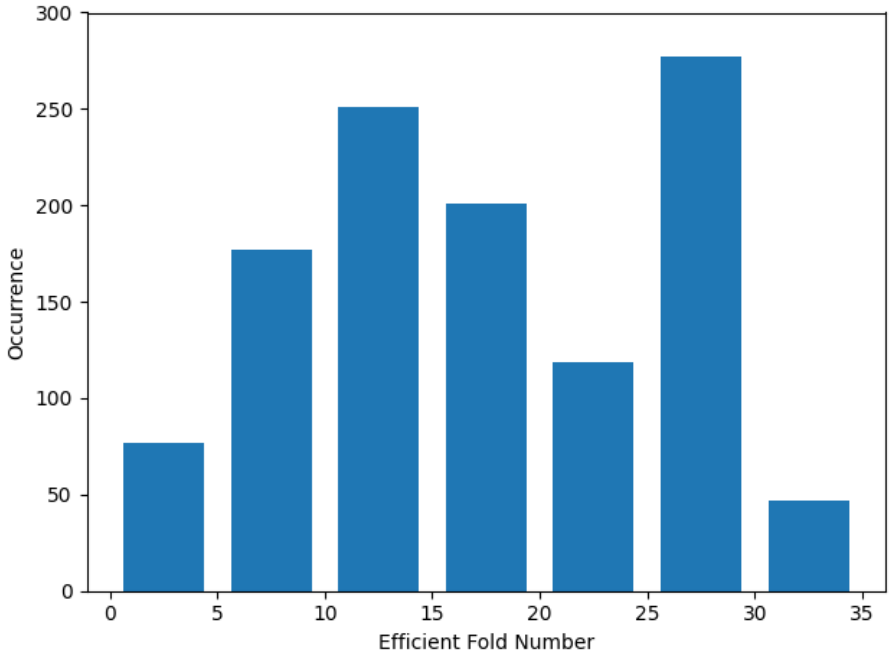


Figure 5.9: The distribution of the efficient number of folds.

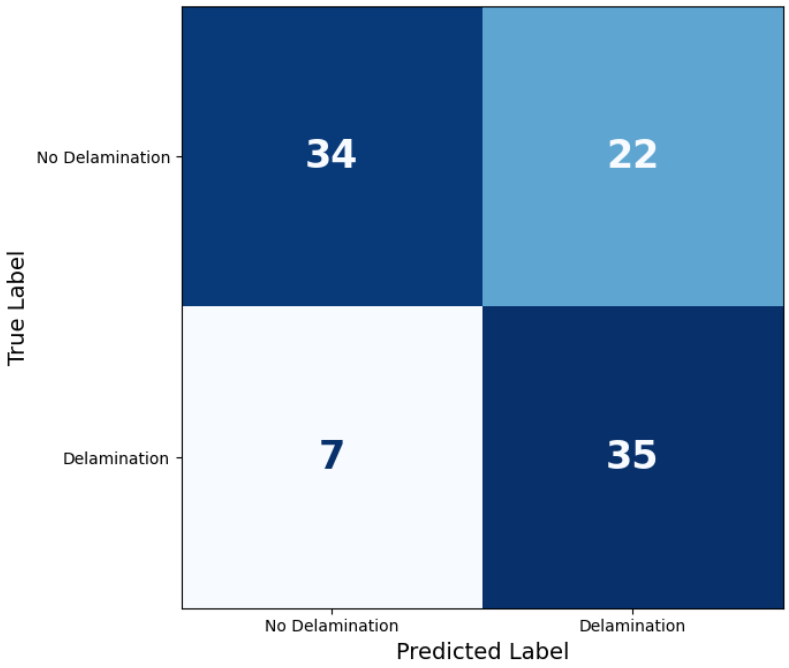


Figure 5.10: A confusion matrix for the $k = 26$ SVC model with a sigmoid function which uses the number of true positives, true negatives, false positives, and false negatives.

as the case of a false positive. An overprediction of the number of delaminations does not offer as adverse a failure effect as overpredicting the number of no delamination samples. A system predicted not to contain a delamination may be considered safe and placed back into operation. If damage were to exist, however, catastrophic, unexpected failure could occur. On the other hand, predicting a delamination would signal the necessity to evaluate the system further to determine if a delamination existed at the given point of interest. Ideally, an implemented model would reduce the number of false positives and false negatives to limit the need for a secondary examination of a material or system. Increasing the number of samples in the dataset could increase the variance in the feature values, ultimately providing more information for learning and trend detection.

5.5.2.2 Multi-layer Perceptron Performance

A total of $1,149$ MLP models were examined during this work. Three parameters were manipulated to observe their effect on the performance of this type of algorithm. The activation function was varied to include all the possible functions available within the scikit-learn package [37]. The number of hidden layers and the perceptrons in each hidden layer were parametrically studied

to examine the effect of the MLP's architecture on its performance. One, two, and three hidden layer networks were investigated, and each layer was comprised of a number of perceptions ranging from one to 100. The general architecture of the MLPs had four input perceptrons, a number of specified perceptrons in hidden layers, and two output perceptrons correlated to the two class labels:

$$\text{One hidden layer:} \quad 4 - N_i - 2, \quad N_i = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 25, 30, 40, 50, 60, 70, 75, 80, 90, 100$$

$$\text{Two hidden layers:} \quad 4 - N_i - N_i - 2, \quad N_i = 10, 20, 25, 30, 40, 50, 60, 70, 75, 80, 90, 100$$

$$\text{Three hidden layers:} \quad 4 - N_i - N_i - N_i - 2, \quad N_i = 20, 40, 60, 80, 100$$

Table C1 contains the activation functions implemented in this work that are available with [37] for a MLP classifier.

The performance of the MLP models were compared in terms of their runtime and efficient fold mean accuracy in Figs. 5.11, 5.12, and 5.13. A standardized z-score, Eq. (5.49), was implemented to transform these two quantities into a normal distribution to compare the MLP models against one another. The runtime value corresponded to the computational time required to compute all of the k-fold cross-validation schemes, not just the efficient fold. Many of the MLP models failed to converge to the optimization tolerance of $1e-4$ when using an epoch of 800 iterations. The runtime was hypothesized and observed to increase if convergence issues were obtained for a respective model.

The single hidden layer models experienced the most considerable variance in performance for both parameters, as shown in Fig. 5.11. The number of single hidden layer models run was less than 20% of the other two architectures. The number of nodes in the single hidden layer was selected for consistency with the two and three hidden layer architecture studies. Confidence in the results of the single hidden layer models was not high because of the lack of variation in the examined models. No decisive conclusion was determined as to whether the single hidden layer architecture was best suited for this classification because of the limited number of models examined.

In the case of the two hidden layer models, the runtime of the sigmoid (logistic) and hyperbolic tangent (tanh) were comparable and closest to the overall mean of all the two hidden layer models in Fig. 5.12. The tanh function models showed the second least variance among both parameters for the models only outperformed by the identity function. In contrast, the logistic function models exhibited the greatest variance for the efficient fold accuracy, and several outlier models existed with significantly poor scoring performances relative to the population's mean.

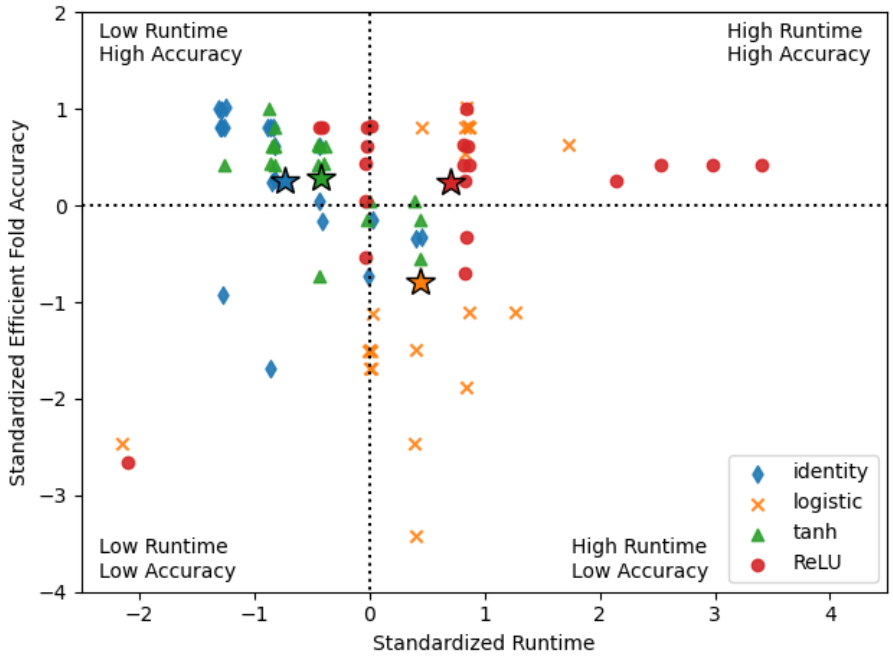


Figure 5.11: The one hidden layer MLP models were found to have a wide spread for both the accuracy and runtime values. The variance was attributed to the lack of models run.

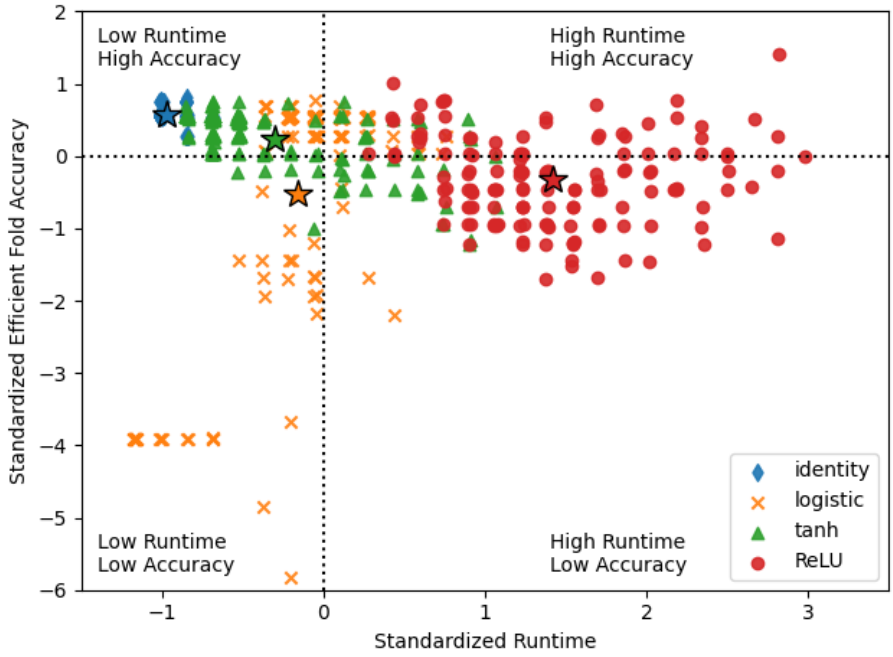


Figure 5.12: The two hidden layer MLP models were found to have a wide spread of runtime. The stars represent the mean parameters for each set.

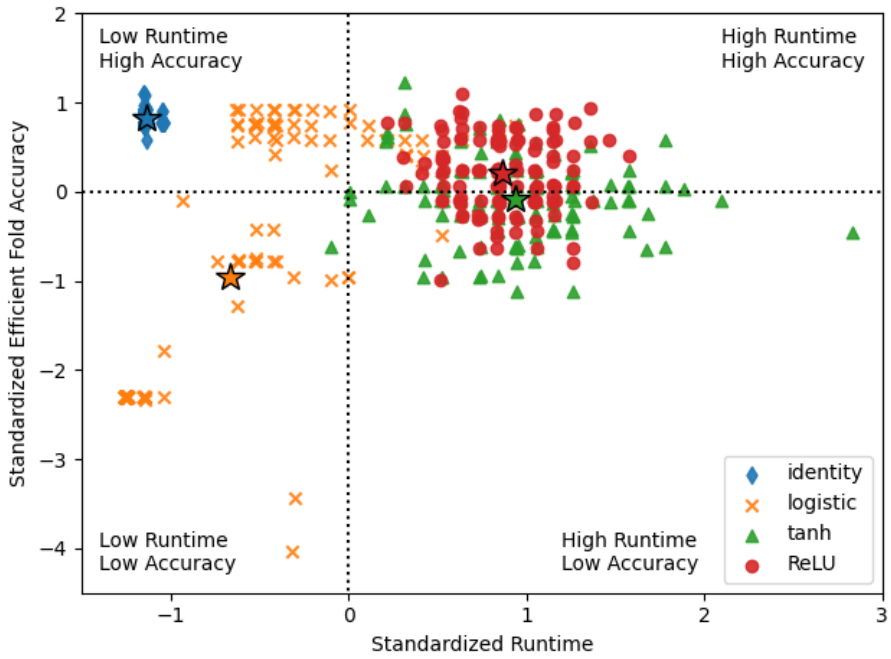


Figure 5.13: The three hidden layer MLP models had several outliers with respect to runtime and accuracy. The performance of the models for the tanh and ReLU models appear comparable based on their clusters and mean values. The stars represent the mean parameters for each set.

The effect of including negative values appeared to impact the performance and runtime of the two hidden layer models. Both the identity and hyperbolic tangent activation functions accounted for negative values instead of mapping them to zero. The models incorporating these two functions made predictions with the cumulative highest mean accuracy scores. These functions also had the shortest runtime averages. The logistic and ReLU functions, which mapped negative values to zero, produced the worst average scoring performances. Equating any negative values to zero can decrease the ability of a model to fit the given data; this behavior has been defined as the dying ReLU problem [160, 161]. In the case of two hidden layers, the need to pass negative values through the function was attributed to the normalization of the feature data implemented in the machine learning pipeline. A z-score normalization yielded both positive and negative values centered around a mean of zero; therefore, the feature vectors contained values less than zero. Mapping the negative values of features to zero inhibited the models from adequately accounting for these features during learning.

The inclusion of a negative value from the activation function played a lesser role in the

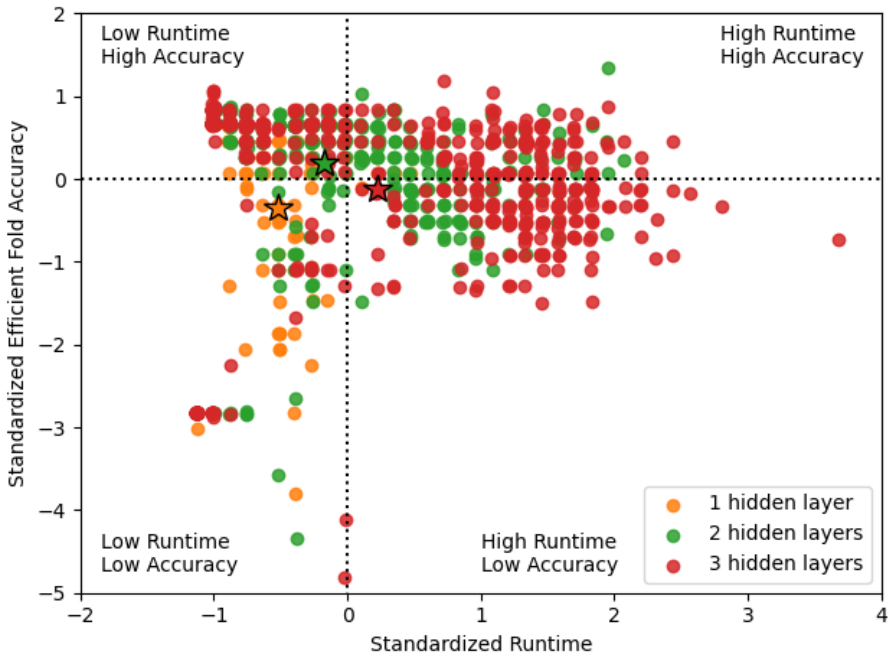


Figure 5.14: The two hidden layer MLP models had on average a greater mean accuracy using a k-fold cross-validation scheme. Several outliers existed for both architectures, but most algorithms had runtimes close to the mean of all the MLP models. The stars represent the mean parameters for each set.

scoring performance of the three hidden layer models. Both the runtime and accuracy appeared unaffected for the identity function models by introducing an additional hidden layer. All of the models implementing this activation function were clustered in the low runtime, high accuracy quadrant in Fig. 5.13. The logistic function models exhibited the same behavior for three hidden layers as it did with two hidden layers. A widespread of accuracy values were observed. A cluster of models employing the logistic activation function had accuracy values comparable to that of the identity function. However, numerous outliers lowered the overall mean accuracy value for the overall function to the lowest of the four activation functions. The performance of the ReLU and tanh functions yielded comparable results as shown by their clusters and mean normalized values in Fig. 5.13. The models employing the tanh function had a wider spread than those employing the ReLU function, and a runtime outlier was observed. While the mean accuracy of the tanh function models was second-lowest, only above that of the logistic function models, the highest accuracy for a three hidden layer model was obtained using this activation function with a $4 - 100 - 40 - 60 - 2$ architecture.

The overall mean performance for the two and three hidden layer models were comparable. Figure 5.14 shows that the mean performance of the three hidden layer models was slightly worse in both categories than that for the two hidden layer models. A negative trade-off was observed between the mean accuracy and runtime as the number of hidden layers was increased in the two and three hidden layer models. The mean runtime difference for the three hidden layer architecture and the overall mean runtime of all the MLP models was greater than that for the two hidden layer models. While computational abilities have increased, the increased number of nodes and calculations associated with hidden layers correlated to greater computational times. Ergo, the three hidden layer models predominantly resulted in greater runtimes. The hidden layer node combination of $60-100-100$ with the tanh activation function required the longest computational runtime of all the MLP models at nearly 40 minutes. This was due to convergence issues as the tolerance between the calculated and predicted labels during training was not achieved for most of the folds in the k-fold scheme. The addition of hidden layers increased the complexity of the models. The complexity associated with this two-class problem classification seemed better suited for a two hidden layer architecture.

The one hidden layer models exhibited a broad spread of accuracy values for runtimes only a standard deviation below the mean MLP runtime. The accuracy spread of the two and three hidden layer architectures were similar as the magnitudes of their maximum and minimum values were comparable. A majority of the models yielded accuracy values within one standard deviation of the MLP mean accuracy. Accuracy outliers were present for all three hidden layer architectures. The three hidden layer architecture presented the greatest number (59) of outliers that fell well below the mean accuracy of the MLP algorithm, especially in the bottom 100 accuracy values. These outliers contributed to the decrease of this architecture's overall mean accuracy being less than that of the MLP algorithm.

The computational runtime and accuracy were found to have an inverse relationship, but this relationship was considered insignificant based on the examined models. The greatest accuracy model had a runtime value at approximately two standard deviations away from the MLP algorithm mean runtime. Furthermore, two of the three worst-performing models in accuracy occurred at approximately the MLP mean runtime. Independent of architecture, the five worst-performing MLP models had a runtime within one standard deviation less than the mean MLP runtime. The inverse relationship between runtime and accuracy was observed through the examination of the clusters for

each architecture. The two hidden layer models, which had a lower mean runtime, also outperformed the three hidden layer models in terms of accuracy. This suggests that a two-hidden layer model could model the necessary relationships to predict a delamination. The choice of this architecture would reduce the increased complexities and computational times associated with sizeable neural network architectures like those used in deep learning.

5.5.2.3 Comparison of All Tested Models

Comparing the accuracy scores for each of the tested models, Fig. 5.15, the multi-layer perceptron neural network algorithm category was found to outperform the other seven algorithms. The ten models with the highest accuracy values were found to be MLPs as listed in Tab. 5.5. The MLP models contained 99 of the top 100 highest accuracy models; 59 of those models were three hidden layer MLP models. The highest non-MLP model (the SVC-sigmoid model) ranked 90^{th} for the highest accuracy. The ability of a neural network to learn and predict nonlinear and complex relationships was evident in these findings. The plots of the sensing voltage, such as Fig. 5.1, have been found to produce indiscernible induced voltage readings leading to the inability to predict the presence of a delamination. The investigated machine learning algorithms have demonstrated the ability to combine stacking sequence information about the laminate with an adjusted voltage reading to detect the presence of a delamination for a given sample point. The performance of the MLP models suggested that nonlinear relationships could be defined between the features and target labels. The relationships between the feature data and target labels were previously unknown by visual observation of the experimental data.

While the MLP algorithm resulted in the highest efficient fold accuracy values, it also resulted in the models with the lowest accuracy values listed in Tab. 5.6. Of the 100 models with the worst accuracy values, Fig. 5.17 shows that 92% of these models were neural networks using an MLP formulation. Three hidden layer models dominated the list of worst models with 59 of the worst performances. Figure 5.15 illustrates that the number of models with below-average accuracy or an outlier performance increased as additional hidden layers were added to the MLP architecture. The increased complexity and calculations associated with adding additional hidden layers were unnecessary for this classification problem; the increased complexity hindered performance.

Table 5.6 contains the ten worst performing models in terms of accuracy, and the logistic (sigmoid) activation function was employed in all of these models. The logistic function was previ-

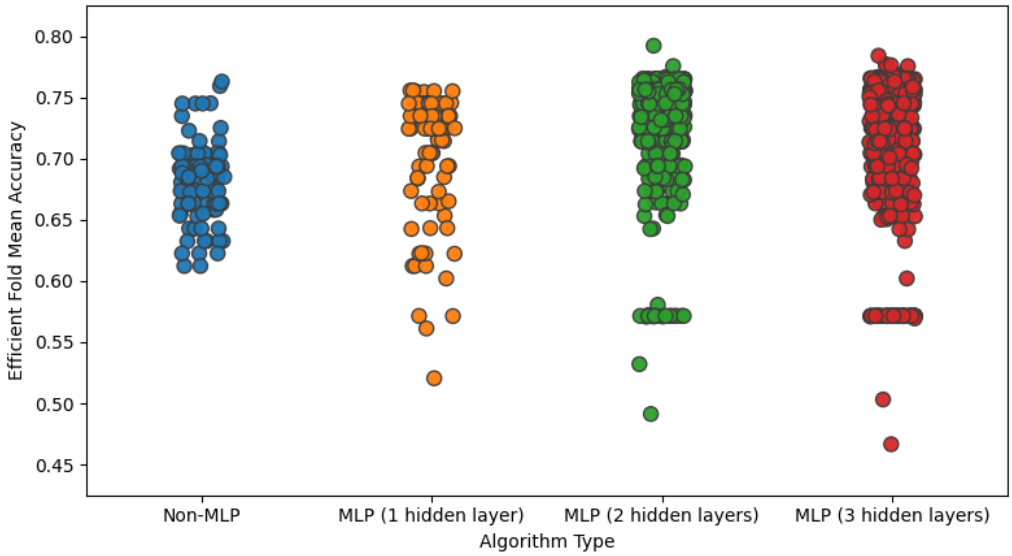


Figure 5.15: In terms of the efficient fold mean accuracy, the MLP models outperformed the non-MLP models on average. More outliers, however, were found for the MLP models.

ously shown to result in more accuracy outliers than the other activation functions. The failure to account for negative failures was attributed to this finding because seven of the top ten performing models accounted for negative input values for each perceptron/node. Contradicting the hypothesis that an increase in the number of hidden layers resulted in a decreased performance, 80% of the top ten performing models were MLPs containing three hidden layers and 59% of the top 100. Through a parametric study, no conclusion concerning the effect of the number of hidden layers on the accuracy of the MLP models was discerned based on the findings of the investigated models. No hidden layer architecture yielded greater (or worse) results than another. The parametric study demonstrated the need to investigate variations of a model to identify the best performing parameters.

Table 5.5: Top ten performing models based on accuracy.

Algorithm	Model Name (Attributes)	Efficient Fold Accuracy
MLP (2 hidden layers)	ReLU, adam, $90-100$	79.22%
MLP (3 hidden layers)	tanh, adam, $100-40-60$	78.40%
MLP (3 hidden layers)	identity, adam, $100-100-20$	77.68%
MLP (3 hidden layers)	identity, adam, $60-60-80$	77.68%
MLP (3 hidden layers)	ReLU, adam, $60-20-60$	77.62%
MLP (2 hidden layers)	ReLU, adam, $20-10$	77.55%
MLP (3 hidden layers)	identity, adam, $40-80-40$	77.54%
MLP (3 hidden layers)	identity, adam, $40-40-80$	76.86%
MLP (3 hidden layers)	identity, adam, $60-100-80$	76.67%
MLP (3 hidden layers)	identity, adam, $60-60-100$	76.67%

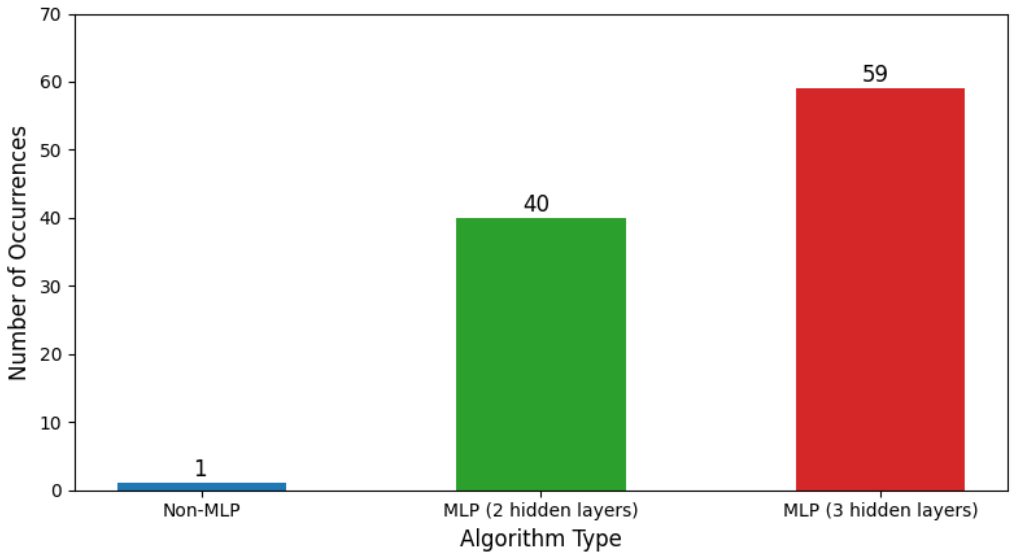


Figure 5.16: The MLP models with two and three hidden layers significantly outperformed the non-MLP and single hidden layer models in terms of greatest accuracy scores.

Table 5.6: Bottom ten worst performing models based on accuracy.

Algorithm	Model Name (Attributes)	Efficient Fold Accuracy
MLP (3 hidden layers)	logistic, adam, $40-100-20$	46.67%
MLP (2 hidden layers)	logistic, adam, $90-10$	49.14%
MLP (3 hidden layers)	logistic, adam, $60-80-80$	50.32%
MLP (1 hidden layer)	logistic, adam, 2	52.05%
MLP (2 hidden layers)	logistic, adam, $60-70$	53.21%
MLP (1 hidden layer)	ReLU, adam, 1	56.12%
MLP (3 hidden layers)	logistic, adam, $80-20-80$	56.94%
MLP (2 hidden layers)	logistic, adam, $10-60$	57.08%
MLP (2 hidden layers)	logistic, adam, $10-70$	57.10%
MLP (2 hidden layers)	logistic, adam, $30-100$	57.10%

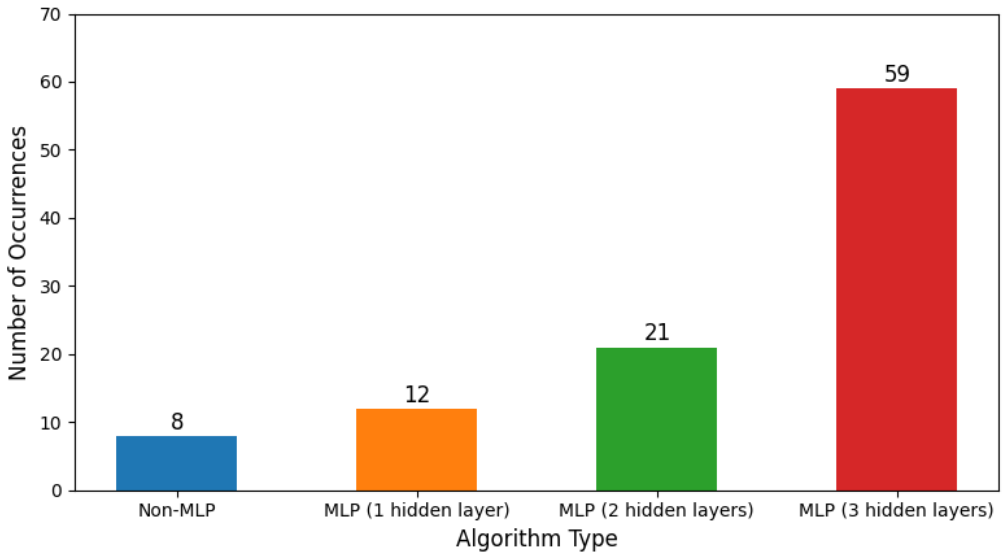


Figure 5.17: The MLP models had the most number of models with the lowest accuracy values in the bottom 100.

5.6 Conclusions

Eight general machine learning algorithms were considered in this work to predict the presence of embedded delaminations in CFRP beams. A testing dataset of 98 points was extracted from a set of $1,380$ points found in four literature sources [6, 57, 127, 155]. The final dataset was determined by analyzing each source and using a proposed adjusted voltage parameter to standardize the induced sensing voltage values. A single literature source, [57], was reasoned to be the best source for this preliminary investigation because all the employed CFRP beams were made of the same material (HexPly AS4/3501-6) and unidirectional. However, the lack of variation in CFRP beams reduced the applicability of the trained machine learning algorithms.

The k-fold cross-validation methodology utilized in this work reduced the variance in the accuracy results for each algorithm type as a whole by nearly 84%. The mean accuracy for the algorithms as a whole decreased for all but the SVM algorithms when using a cross-validation scheme to divide the dataset into training and testing data. The SVM and neural network were two of the best performing algorithms due to their high level of customizability which was examined through a parametric study. The decision tree, stochastic gradient descent, and nearest neighbor algorithms were ineffective in this preliminary investigation to predict embedded delaminations. The decision tree algorithm, for example, is not suitable for generalized problems. The introduction of varying combinations of data through a cross-validation scheme improves the generalization of a problem of interest; thus, explaining its low accuracy scores. However, the ensemble method had a greater mean accuracy because it employed multiple decision trees improving generalizability. The utilization of a cross-validation scheme was suggested to reduce variability in results while introducing "more" data to an algorithm for training; the results of the cross-validation experiments were of interest in this work.

The MLP models were found to yield the best and worst-performing models by examining each of the $1,249$ models individually. The ability of a neural network to model nonlinear relationships was presumed to increase the prediction accuracy score of an ML algorithm as 99 of the top 100 models utilized the MLP formulation. The greatest accuracy achieved was 79.22% with a two hidden layer MLP architecture using the rectified linear unit activation function (ReLU, adam, $4 - 90 - 100 - 2$). On the contrary, the MLP formulation yielded 85 of the 100 worst-performing ML models. The architecture of the MLP was found to influence the performance of the models.

Increasing the number of hidden layers in the neural network increased the complexity and calculations involved with prediction making. A direct relationship was observed between the number of hidden layers and the number of worst-performing outliers. The MLP models had 12, 21, and 59 of these worst-performing outliers for one, two, and three hidden layers, respectively.

The accuracy of the MLP models was compared against a standardized runtime quantity to correlate the computational time with prediction making. The one hidden layer models performed below the mean accuracy and runtime of all the examined MLP models. However, only 84 MLP with one hidden layer were examined compared to over 550 for both the two and three hidden layer models. The difference in the number of models and comparison in performance concluded that a single hidden layer MLP was not adequate to predict the presence of a delamination using the obtained data features. The two hidden layer MLPs outperformed the three hidden layer models by having a mean runtime z-score of -0.178 and mean accuracy z-score of 0.200, whereas the three hidden layer MLP mean z-scores were 0.222 and -0.113, respectively. While an increase in the number of hidden layers resulted in a greater number of worst accuracy outliers, no significant conclusive difference was discerned in this preliminary investigation between the two and three hidden layer MLPs.

The average accuracy score of all the $1,249$ examined models was approximately 71%, which was lower than results from other machine learning applications with NDE. The parametric study employed in this work illustrated the spread of possible results with an MLP due to changing numbers of nodes and hidden layers. A study on the architecture of a neural network was found to influence the decision making regarding its application to a problem. No one activation function or number of hidden layers presented prediction accuracies that significantly outperformed other combinations of parameters. Further investigation of the architecture of MLP models would aid in better determining a suitable formulation for a delamination predicting MLP.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This work aimed to gain insight into the capability of predicting the presence of delaminations within composite materials using embedded magnetostrictive particles. Implementing a non-destructive evaluation method allows for damage detection without significant, if any, alterations to a system. The discussed NDE method involved embedding particles of the magnetostrictive material Terfenol-D between carbon fiber reinforced polymer laminae to yield a smart, damage sensing composite material. The magnetostrictive particles coupled its mechanical and magnetic properties such that a change in the local stress field due to the presence of a delamination or other damage precursor resulted in changes in its magnetic permeability. Previous experimental investigations of this NDE method [6, 7, 57, 150] observed no discernible pattern in the induced sensing voltage for the locations along the experimental CFRP beams containing embedded delaminations. Alternative detection and prediction methods were considered in this work to address the inability to predict the presence of a delamination in experimental CFRP beams embedded with Terfenol-D particles. Three areas were examined in this work: i) an analytical modeling approach to determine induced sensing voltages that involved the characterization of Terfenol-D, ii) the determination of effective property values to quantify the stress in the sensing lamina, and iii) the implementation of machine learning algorithms to predict the presence of embedded delaminations.

A previously proposed analytical model [53] was examined in this work to compute the induced stress in an embedded magnetostrictive material; the induced stress could then be correlated

to an induced sensing voltage. The model employed nonlinear relationships to characterize the stress-strain and strain-magnetic field intensity relationships. Both of the previously proposed models in [53] were found to err in quantifying the nonlinear relationships. New relationships were proposed in this work in the form of Fourier series for their ability to fit the experimental stress-strain and strain-magnetic field intensity data from [120] as well as the derived compliance term and piezomagnetic coefficient. A three-term Fourier series for the stress-strain relationship characterized the entire domain of the experimental data presented in [120] with a maximum RMSE on the order of 10^{-1}. A five-term Fourier series captured the strain-magnetic field intensity data with an RMSE on the order of 10^{-6}. However, the Fourier series were only viable to characterize the domain used to fit the experimental data; outside the domain, the functions oscillated due to the trigonometric functions involved in the series.

The iterative stress calculation previously proposed in [53] failed to reasonably converge using the newly proposed nonlinear relationships. Convergence was found to be unlikely using the formulation presented in [53] because one of the nonlinear relationships, the magnetic strain ϵ_H, was a function of the magnetic field intensity. As the magnetic field intensity increased, this strain value increased; therefore, decreasing the likelihood of convergence. The structure of the composite examined was determined to have a negligible effect on the calculation of the induced stress. Two elastic terms were utilized in the iterative scheme to represent the makeup of the composite material. The presence of a delamination had little impact on the value of these terms. The calculation of these terms differed from that of the stiffness terms found using the classical lamination theory; the CLT calculated values were 222% greater than those calculated using the formulation presented in [53]. From this investigation, it was concluded that an alternative method for calculating the induced stress in a magnetostrictive material be found, especially since the method proposed in [53] was limited to include only the effects of an applied magnetic field.

To improve the modeling of the sensing lamina, effective properties were examined for combining the properties of the embedded Terfenol-D particles and the host matrix Hexcel 3501-6. Five widely recognized approximations were applied to the determination of the elastic, shear, and bulk moduli along with the Poisson's ratio. The moduli demonstrated the bounding problem associated with effective properties where approximations yield upper and lower bounds for possible values. The Poisson's ratio, though, failed to follow previously suggested bounds, especially the Voigt-Reuss bounds. Fifteen previously proposed coefficient of thermal expansion approximations were employed

in this work to quantify the parameter. Nearly half of the approximations estimated the relationship between the volume fraction of inclusions and the value of the coefficient of thermal expansion to be nearly linear. Using the classical lamination theory, the effect of the effective coefficient of thermal expansion was examined for modeling the fabrication of the sensing lamina, specifically the cooling step. The Voigt approximation for the elastic properties yielded the most compressive, i.e., the greatest magnitude, induced thermal stress, whereas the other four approximations examined predicted comparable magnitudes. A direct correlation between the value of the effective coefficient of thermal expansion and the magnitude of the induced stress was observed. The choice of approximations to model the sensing lamina was concluded to impact the magnitude of the induced thermal stress significantly.

The major contribution of this work was the consideration of machine learning for the prediction of embedded delaminations using the magnetostrictive particle NDE method. A dataset of 98 points was taken from a previous experimental study [57]. Eight general machine learning formulations were considered. A parametric study was conducted to manipulate various parameters for some formulations, and a total of $1,249$ unique models were examined. The mean accuracy of the $1,249$ models was approximately 71%, which was less than values found in previous ML applications to NDE. The k-fold cross-validation scheme implemented reduced the variance in the obtained accuracy by nearly 84% through the introduction of combinations of the dataset to the models. Based on the reduction in the variance of the scores, it was concluded to examine the cross-validation results further and consider the utilization of this approach in future investigations. The SVM and neural network formulations yielded the highest accuracy scores for an algorithm at 72.13% and 71.53%, respectively. The neural network formulation was specifically considered because of the number of definable parameters to tailor a model.

The activation function, number of hidden layers, and the number of nodes in each hidden layer were manipulated to consider their effect on the prediction accuracy. The number of hidden layers and performance were correlated; as the number of hidden layers increased, the number of accuracy outliers increased (in terms of both best and worst performance). The three hidden layer MLP models produced 59% of the top 100 performing models as well as 59% of the worst 100 performing models. Though the three hidden layer MLP models had the greatest number of top-performing models, the $4-90-100-2$ MLP utilizing the rectified linear unit activation function had the greatest accuracy at 79.22%. The two hidden layer models were found to perform better in terms

of accuracy and runtime with z-scores of 0.200 and -0.178, respectively, compared to that of the entire set of MLP models. This preliminary study offered two main conclusions regarding the architecture of an MLP. First, based on this work, a one hidden layer model failed to account for the complexity involved with the prediction making of embedded delaminations using the investigated features. Second, the two and three hidden layer MLPs showed comparable performance, and no conclusion as to a preferred architecture was obtained. Though, the two hidden layer MLPs outperformed the three hidden layer models in terms of both average accuracy and runtime. A parametric study should be considered to optimize the architecture and parameters of an MLP for implementation. From this work, an MLP neural network was concluded to be the best machine learning formulation to apply to the embedded magnetostrictive NDE method as it can account for nonlinearities through the combination of the feature values using linear relationships. Through this work, a preliminary ML investigation was shown to predict the presence of embedded delaminations using induced sensing voltages obtained from previous experiments. The use of a ML algorithm for damage detection was concluded to be a viable combination with the magnetostrictive particle NDE method to predict the presence of delaminations considering the inability to identify delaminations using plots of the induced sensing voltage.

6.2 Future Work

By applying machine learning to the embedded magnetostrictive particle NDE method, phenomena were observed that could impact the prediction making of an algorithm. The data introduced to the algorithm through training and testing is essential as this is the only information given to a model. Each sample point along the length of a composite beam would ideally be the same except for the presence of delamination and Terfenol-D. The distribution of Terfenol-D particles was previously found to negatively affect the performance of the discussed NDE method due to localized accumulations of particles. Distribution methods were previously suggested to mitigate these effects for the particles, such as a trenching method that distributes particles using a channel apparatus. Alternatively, a magnetostrictive film was employed to eliminate the concerns of an uncontrollable distribution of particles. An independent fabrication process was utilized in producing the films to allow for complete control of their creation. This film sought to yield a more even distribution of the magnetostrictive material to obtain more consistent induced sensing voltage readings across the

length of a beam. Any damage or fabrication mishaps could be better illustrated using the film technique.

Further investigation is necessary to determine if the selection of a sensing film reduces any mechanical effects that may arise through the use of particles. When loaded, the particles in a sensing lamina would act as stress concentrations, and cracks could propagate from the interface between the Terfenol-D particles and the host matrix. Research into the effect of the inclusion of particles is paramount to determine if they adversely impact the mechanical performance of a composite. The sensitivity of the NDE method is also necessary to quantify to determine how the magnetostrictive phenomenon responds to any cracks or voids surrounding the Terfenol-D particles due to their embedding.

An increased amount of uniform testing data would aid the machine learning algorithms in making predictions. The extracted data was inconsistent between sources in the information presented regarding the examined features; this led to a reduction in the amount of applicable data. A single testing setup should be employed to collect additional uniform experimental data to continue this preliminary investigation. Increasing the number of sample points with embedded delaminations would improve the quality (and quantity) of information provided to machine learning algorithms about this target label. Signal noise could also be introduced into the data to add variance to the values in the feature vectors. The induced sensing voltage, or adjusted RMS voltage, would be suitable for introducing noise as the other features were integer or binary values. An analytical model could also provide useful data to machine learning algorithms. The investigated analytical model proposed in [53] failed to converge in this work and was limited in application. Accounting for external mechanical loads and differentiating the initial prestress of the Terfenol-D from that caused by external loads would expand the model. A surrogate neural network approach could also be applied to generate data. Using inputs such as the presence of Terfenol-D or a delamination, a surrogate network could produce sensing voltages or laminate information. The data produced through the surrogate network would be combined to create a single sample point for use in a single trained network to predict the presence of a delamination.

An investigation into the feasibility of accounting for information through the feature vectors was started in this work but deemed unnecessary by extracting uniform experimental data from only one source. Inclusion of the constituent material properties and the stacking sequence of the composite beam would increase the applicability of the machine learning algorithms. Utilization of

stiffness terms or values from the **ABD** matrix were considered; however, the curse of dimensionality was feared with the introduction of additional features, especially with the limited dataset size. The use of terms from the **ABD** matrix would account for the constituent materials, the stacking sequence, and the presence of a delamination. Further study is necessary to identify the terms most impacted by the inclusion of a delamination region and the stacking sequence. Not all 18 independent terms would be suitable as features. Increasing the features to account for additional information about the laminate of interest would improve the applicability of the trained algorithm to include different materials and stacking sequences than those investigated in this work. However, the training data must contain samples for the possible examined laminates to avoid significant classification errors. An increase in the number of sample points is advisable for any increase in the number of features.

The cross-validation results presented were mean values for one model using k folds. Multiple experiments of the same model formulation with the same number of folds would yield results that would improve the confidence of the reported mean accuracy values. A more extensive parametric study of the number of perceptrons in each hidden layer could yield a more accurate MLP model. Arbitrary numbers of perceptrons were chosen for each hidden layer. However, a grid search technique can be computed using Python to determine the optimal parameters for a specified metric by looping through a set of given parameter values [37]. Alternative packages for Python offer additional capabilities for predicting with machine learning algorithms by increasing the user-defined parameters of the algorithms. TensorFlow [162], for instance, implements user-defined neural networks to solve problems. Deep learning neural networks are well suited for study using TensorFlow as it allows users to define the characteristics and formulation of each layer of the neural network. While the results obtained in this work were acceptable for a preliminary investigation into the application of machine learning with the discussed NDE method, further manipulation of the parameters of the algorithms could increase the prediction accuracy of the models to detect delaminations.

Appendices

Appendix A Approximations for the Effective Coefficient of Thermal Expansion

The following are approximations used to calculate the effective coefficient of thermal expansion as a function of the volume fractions of the inclusion and host material. Blackburn approximation [154]:

$$\overline{\alpha} = \alpha_p + \frac{\frac{3}{2}(1-\nu_p) V_m (\alpha_m - \alpha_p)}{\frac{1}{2}(1+\nu_p) + V_m (1-2\nu_p) + (1-2\nu_m)\frac{E_p}{E_m} V_p} \tag{A.1}$$

Turner approximation [163]:

$$\overline{\alpha} = \frac{V_m \alpha_m K_m + V_p \alpha_p K_p}{V_m K_m + V_p K_p} \tag{A.2}$$

Cribb approximation [164]:

$$\overline{\alpha} = q_1 \alpha_m + q_2 \alpha_p \tag{A.3}$$

$$q_1 = \frac{K_m (K_c - K_p)}{K_c (K_m - K_p)} \tag{A.4}$$

$$q_2 = \frac{K_p (K_m - K_c)}{K_c (K_m - K_p)} \tag{A.5}$$

Kerner approximation [165]:

$$\overline{\alpha} = \alpha_p V_p + \alpha_m V_m + V_p V_m (\alpha_m - \alpha_p) q \tag{A.6}$$

$$q = \frac{\frac{1}{K_m} - \frac{1}{K_p}}{\frac{V_p}{K_p} + \frac{V_m}{K_m} + \frac{3}{4G_m}} \tag{A.7}$$

Wang and Kwei approximation [151]:

$$\overline{\alpha} = \alpha_m - V_p q (\alpha_m - \alpha_p) \tag{A.8}$$

$$q = \frac{\left(\frac{3E_p}{E_m}\right) V_p}{\frac{E_p}{E_m}\left[2V_p (1-2\nu_m) + (1+\nu_m)\right] + 2V_m (1-2\nu_p)} \tag{A.9}$$

The above expressions, specifically for q, were found to be inconsistent with that originally presented in [166]:

$$\theta = \frac{3\left(\frac{E_p}{E_m}\right)(1-\nu_m)}{\frac{E_p}{E_m}\left[2V_p\left(1-2\nu_m\right)+\left(1+\nu_m\right)\right]+2V_m\left(1-2\nu_p\right)} \tag{A.10}$$

$$\overline{\alpha} = \alpha_m\left[1-V_p\left(1-\frac{\alpha_p}{\alpha_m}\right)\theta\right] \tag{A.11}$$

Thomas approximation [151]:

$$\overline{\alpha}^a = \alpha_m^a V_m + \alpha_p^a V_p \tag{A.12}$$

where $-1 \leq a \leq 1$ can result in the Voigt and Reuss approximations, respectively. Tummala-Friedberg approximation [167]:

$$\overline{\alpha} = \alpha_m - V_p q\left(\alpha_m - \alpha_p\right) \tag{A.13}$$

$$q = \frac{\frac{1+\nu_m}{2E_m}}{\frac{1+\nu_m}{2E_m}+\frac{1-2\nu_p}{E_p}} \tag{A.14}$$

Fahmi-Ragai approximation [168]:

$$\overline{\alpha} = \alpha_m - \frac{3V_p\left(\alpha_m-\alpha_p\right)\left(1-\nu_m\right)}{2\left(1-2\nu_p\right)\left(-V_p\right)\frac{E_m}{E_p}+2V_p\left(1-2\nu_m\right)+\left(1+\nu_m\right)} \tag{A.15}$$

Appendix B Data used in Machine Learning

Table B1 contains the 98-point dataset used in the preliminary machine learning investigation.

Table B1: Extracted data used in the machine learning investigation.

Source	Fiber Material	Matrix Material	Plies Above	Plies Below	Stacking Sequence	Terfenol-D	V_{peak}	V_{RMS}	$V_{RMS,adjusted}$	Delamination
Rudd 2014	AS4	3501-6	2	1	$[0_2/\mathrm{m}/\mathrm{d}/0]$	1	0.414	0.293	1.030	0
Rudd 2014	AS4	3501-6	1	2	$[0/\mathrm{m}/0_2]$	1	0.414	0.293	1.027	0
Rudd 2014	AS4	3501-6	3	3	$[0_3/\mathrm{m}/\mathrm{d}/0_3]$	0	0.402	0.284	0.996	0
Rudd 2014	AS4	3501-6	4	4	$[0_4/\mathrm{m}/\mathrm{d}/0_4]$	1	0.409	0.289	1.017	1
Rudd 2014	AS4	3501-6	2	2	$[0_2/\mathrm{m}/0_2]$	0	0.403	0.285	1.002	0
Rudd 2014	AS4	3501-6	2	2	$[0_2/\mathrm{m}/\mathrm{d}/0_2]$	1	0.411	0.291	1.019	1
Rudd 2014	AS4	3501-6	3	3	$[0_3/\mathrm{m}/\mathrm{d}/0_3]$	0	0.402	0.284	1.000	0
Rudd 2014	AS4	3501-6	1	1	$[0/\mathrm{m}/\mathrm{d}/0]$	1	0.407	0.288	1.014	1
Rudd 2014	AS4	3501-6	7	7	$[0_7/\mathrm{m}/0_7]$	0	0.402	0.284	0.999	0
Rudd 2014	AS4	3501-6	7	7	$[0_7/\mathrm{m}/\mathrm{d}/0_7]$	1	0.413	0.292	1.024	1
Rudd 2014	AS4	3501-6	1	1	$[0/\mathrm{m}/\mathrm{d}/0]$	1	0.406	0.287	1.011	0
Rudd 2014	AS4	3501-6	2	2	$[0_2/\mathrm{m}/\mathrm{d}/0_2]$	1	0.410	0.290	1.016	0
Rudd 2014	AS4	3501-6	7	7	$[0_7/\mathrm{m}/\mathrm{d}/0_7]$	0	0.403	0.285	0.999	0

Continued on next page

Table B1 – *Continued from previous page*

Source	Fiber Material	Matrix Material	Plies Above	Plies Below	Stacking Sequence	Terfenol-D	V_{peak}	V_{RMS}	$V_{RMS,adjusted}$	Delamination
Rudd 2014	AS4	3501-6	6	6	$[0_6/\mathrm{m}/\mathrm{d}/0_6]$	1	0.411	0.291	1.021	1
Rudd 2014	AS4	3501-6	5	5	$[0_5/\mathrm{m}/\mathrm{d}/0_5]$	0	0.402	0.284	0.999	0
Rudd 2014	AS4	3501-6	6	6	$[0_6/\mathrm{m}/0_6]$	0	0.402	0.284	0.999	0
Rudd 2014	AS4	3501-6	7	7	$[0_7/\mathrm{m}/\mathrm{d}/0_7]$	1	0.413	0.292	1.024	1
Rudd 2014	AS4	3501-6	2	1	$[0_2/\mathrm{m}/\mathrm{d}/0]$	1	0.410	0.290	1.020	1
Rudd 2014	AS4	3501-6	4	3	$[0_4/\mathrm{m}/\mathrm{d}/0_3]$	1	0.409	0.289	1.016	1
Rudd 2014	AS4	3501-6	1	1	$[0/\mathrm{m}/0]$	0	0.403	0.285	0.999	0
Rudd 2014	AS4	3501-6	1	2	$[0/\mathrm{m}/\mathrm{d}/0_2]$	1	0.407	0.288	1.014	1
Rudd 2014	AS4	3501-6	4	3	$[0_4/\mathrm{m}/\mathrm{d}/0_3]$	1	0.409	0.289	1.016	0
Rudd 2014	AS4	3501-6	1	1	$[0/\mathrm{m}/0]$	1	0.414	0.293	1.027	0
Rudd 2014	AS4	3501-6	3	3	$[0_3/\mathrm{m}/0_3]$	0	0.402	0.284	0.997	0
Rudd 2014	AS4	3501-6	5	4	$[0_5/\mathrm{m}/0_4]$	1	0.411	0.291	1.024	0
Rudd 2014	AS4	3501-6	6	6	$[0_6/\mathrm{m}/\mathrm{d}/0_6]$	1	0.411	0.291	1.021	1
Rudd 2014	AS4	3501-6	2	2	$[0_2/\mathrm{m}/\mathrm{d}/0_2]$	1	0.410	0.290	1.021	0
Rudd 2014	AS4	3501-6	3	3	$[0_3/\mathrm{m}/\mathrm{d}/0_3]$	0	0.402	0.284	0.996	0
Rudd 2014	AS4	3501-6	5	4	$[0_5/\mathrm{m}/0_4]$	1	0.410	0.290	1.021	0

Continued on next page

Table B1 – *Continued from previous page*

Source	Fiber Material	Matrix Material	Plies Above	Plies Below	Stacking Sequence	Terfenol-D	V_{peak}	V_{RMS}	$V_{RMS,adjusted}$	Delamination
Rudd 2014	AS4	3501-6	3	3	$[0_3/\mathrm{m}/\mathrm{d}/0_3]$	1	0.410	0.290	1.021	1
Rudd 2014	AS4	3501-6	4	3	$[0_4/\mathrm{m}/\mathrm{d}/0_3]$	0	0.402	0.284	0.998	0
Rudd 2014	AS4	3501-6	1	1	$[0/\mathrm{m}/\mathrm{d}/0]$	1	0.413	0.292	1.024	0
Rudd 2014	AS4	3501-6	3	2	$[0_3/\mathrm{m}/\mathrm{d}/0_2]$	1	0.410	0.290	1.020	1
Rudd 2014	AS4	3501-6	5	4	$[0_5/\mathrm{m}/\mathrm{d}/0_4]$	1	0.413	0.292	1.027	0
Rudd 2014	AS4	3501-6	6	5	$[0_6/\mathrm{m}/\mathrm{d}/0_5]$	1	0.410	0.290	1.020	1
Rudd 2014	AS4	3501-6	5	5	$[0_5/\mathrm{m}/\mathrm{d}/0_5]$	1	0.410	0.290	1.020	1
Rudd 2014	AS4	3501-6	7	7	$[0_7/\mathrm{m}/0_7]$	1	0.410	0.290	1.020	0
Rudd 2014	AS4	3501-6	7	7	$[0_7/\mathrm{m}/0_7]$	1	0.411	0.291	1.024	0
Rudd 2014	AS4	3501-6	4	4	$[0_4/\mathrm{m}/\mathrm{d}/0_4]$	1	0.407	0.288	1.013	1
Rudd 2014	AS4	3501-6	5	4	$[0_5/\mathrm{m}/0_4]$	1	0.410	0.290	1.021	0
Rudd 2014	AS4	3501-6	5	4	$[0_5/\mathrm{m}/\mathrm{d}/0_4]$	1	0.410	0.290	1.020	1
Rudd 2014	AS4	3501-6	3	2	$[0_3/\mathrm{m}/0_2]$	0	0.403	0.285	1.000	0
Rudd 2014	AS4	3501-6	1	2	$[0/\mathrm{m}/0_2]$	0	0.405	0.286	1.002	0
Rudd 2014	AS4	3501-6	3	3	$[0_3/\mathrm{m}/\mathrm{d}/0_3]$	1	0.407	0.288	1.010	1
Rudd 2014	AS4	3501-6	5	4	$[0_5/\mathrm{m}/\mathrm{d}/0_4]$	1	0.410	0.290	1.020	1

Continued on next page

Table B1 – *Continued from previous page*

Source	Fiber Material	Matrix Material	Plies Above	Plies Below	Stacking Sequence	Terfenol-D	V_{peak}	V_{RMS}	$V_{RMS,adjusted}$	Delamination
Rudd 2014	AS4	3501-6	3	3	$[0_3/\mathrm{m}/\mathrm{d}/0_3]$	0	0.402	0.284	0.996	0
Rudd 2014	AS4	3501-6	3	3	$[0_3/\mathrm{m}/\mathrm{d}/0_3]$	1	0.407	0.288	1.010	0
Rudd 2014	AS4	3501-6	6	5	$[0_6/\mathrm{m}/\mathrm{d}/0_5]$	1	0.410	0.290	1.020	1
Rudd 2014	AS4	3501-6	6	7	$[0_6/\mathrm{m}/\mathrm{d}/0_7]$	0	0.402	0.284	0.999	0
Rudd 2014	AS4	3501-6	2	2	$[0_2/\mathrm{m}/\mathrm{d}/0_2]$	0	0.403	0.285	0.998	0
Rudd 2014	AS4	3501-6	1	1	$[0/\mathrm{m}/\mathrm{d}/0]$	1	0.410	0.290	1.017	1
Rudd 2014	AS4	3501-6	6	6	$[0_6/\mathrm{m}/0_6]$	0	0.405	0.286	1.006	0
Rudd 2014	AS4	3501-6	4	4	$[0_4/\mathrm{m}/\mathrm{d}/0_4]$	1	0.409	0.289	1.017	1
Rudd 2014	AS4	3501-6	6	7	$[0_6/\mathrm{m}/\mathrm{d}/0_7]$	1	0.411	0.291	1.024	1
Rudd 2014	AS4	3501-6	3	2	$[0_3/\mathrm{m}/0_2]$	1	0.411	0.291	1.021	0
Rudd 2014	AS4	3501-6	3	3	$[0_3/\mathrm{m}/\mathrm{d}/0_3]$	1	0.409	0.289	1.013	1
Rudd 2014	AS4	3501-6	5	5	$[0_5/\mathrm{m}/\mathrm{d}/0_5]$	1	0.410	0.290	1.020	1
Rudd 2014	AS4	3501-6	7	7	$[0_7/\mathrm{m}/0_7]$	1	0.409	0.289	1.017	0
Rudd 2014	AS4	3501-6	3	3	$[0_3/\mathrm{m}/0_3]$	1	0.416	0.294	1.032	0
Rudd 2014	AS4	3501-6	6	7	$[0_6/\mathrm{m}/\mathrm{d}/0_7]$	1	0.411	0.291	1.024	1
Rudd 2014	AS4	3501-6	5	4	$[0_5/\mathrm{m}/\mathrm{d}/0_4]$	1	0.411	0.291	1.023	1

Continued on next page

Table B1 – *Continued from previous page*

Source	Fiber Material	Matrix Material	Plies Above	Plies Below	Stacking Sequence	Terfenol-D	V_{peak}	V_{RMS}	$V_{RMS,adjusted}$	Delamination
Rudd 2014	AS4	3501-6	4	4	$[0_4/\mathrm{m}/0_4]$	1	0.409	0.289	1.020	0
Rudd 2014	AS4	3501-6	1	1	$[0/\mathrm{m}/0]$	0	0.403	0.285	0.999	0
Rudd 2014	AS4	3501-6	4	3	$[0_4/\mathrm{m}/\mathrm{d}/0_3]$	0	0.402	0.284	0.998	0
Rudd 2014	AS4	3501-6	1	2	$[0/\mathrm{m}/\mathrm{d}/0_2]$	1	0.407	0.288	1.014	1
Rudd 2014	AS4	3501-6	6	7	$[0_6/\mathrm{m}/\mathrm{d}/0_7]$	1	0.413	0.292	1.027	0
Rudd 2014	AS4	3501-6	2	2	$[0_2/\mathrm{m}/\mathrm{d}/0_2]$	0	0.402	0.284	0.999	0
Rudd 2014	AS4	3501-6	6	6	$[0_6/\mathrm{m}/0_6]$	0	0.402	0.284	0.999	0
Rudd 2014	AS4	3501-6	2	1	$[0_2/\mathrm{m}/\mathrm{d}/0]$	1	0.410	0.290	1.020	1
Rudd 2014	AS4	3501-6	3	2	$[0_3/\mathrm{m}/\mathrm{d}/0_2]$	0	0.403	0.285	0.999	0
Rudd 2014	AS4	3501-6	4	4	$[0_4/\mathrm{m}/0_4]$	1	0.411	0.291	1.027	0
Rudd 2014	AS4	3501-6	1	1	$[0/\mathrm{m}/\mathrm{d}/0]$	1	0.410	0.290	1.021	1
Rudd 2014	AS4	3501-6	5	4	$[0_5/\mathrm{m}/\mathrm{d}/0_4]$	1	0.411	0.291	1.023	0
Rudd 2014	AS4	3501-6	3	2	$[0_3/\mathrm{m}/\mathrm{d}/0_2]$	1	0.409	0.289	1.017	1
Rudd 2014	AS4	3501-6	2	2	$[0_2/\mathrm{m}/0_2]$	1	0.410	0.290	1.019	0
Rudd 2014	AS4	3501-6	3	2	$[0_3/\mathrm{m}/\mathrm{d}/0_2]$	1	0.413	0.292	1.027	0
Rudd 2014	AS4	3501-6	4	3	$[0_4/\mathrm{m}/\mathrm{d}/0_3]$	1	0.411	0.291	1.021	1

Continued on next page

Table B1 – *Continued from previous page*

Source	Fiber Material	Matrix Material	Plies Above	Plies Below	Stacking Sequence	Terfenol-D	V_{peak}	V_{RMS}	$V_{RMS,adjusted}$	Delamination
Rudd 2014	AS4	3501-6	4	3	$[0_4/\mathrm{m}/0_3]$	1	0.409	0.289	1.018	0
Rudd 2014	AS4	3501-6	1	1	[0/m/d/0]	0	0.403	0.285	0.999	0
Rudd 2014	AS4	3501-6	4	3	$[0_4/\mathrm{m}/\mathrm{d}/0_3]$	1	0.410	0.290	1.019	1
Rudd 2014	AS4	3501-6	4	3	$[0_4/\mathrm{m}/\mathrm{d}/0_3]$	1	0.413	0.292	1.025	1
Rudd 2014	AS4	3501-6	6	7	$[0_6/\mathrm{m}/0_7]$	1	0.413	0.292	1.030	0
Rudd 2014	AS4	3501-6	6	5	$[0_6/\mathrm{m}/0_5]$	1	0.410	0.290	1.024	0
Rudd 2014	AS4	3501-6	3	3	$[0_3/\mathrm{m}/\mathrm{d}/0_3]$	1	0.409	0.289	1.017	1
Rudd 2014	AS4	3501-6	4	3	$[0_4/\mathrm{m}/\mathrm{d}/0_3]$	1	0.413	0.292	1.025	1
Rudd 2014	AS4	3501-6	4	3	$[0_4/\mathrm{m}/\mathrm{d}/0_3]$	0	0.406	0.287	1.009	0
Rudd 2014	AS4	3501-6	3	2	$[0_3/\mathrm{m}/0_2]$	0	0.403	0.285	1.000	0
Rudd 2014	AS4	3501-6	2	2	$[0_2/\mathrm{m}/\mathrm{d}/0_2]$	1	0.411	0.291	1.019	1
Rudd 2014	AS4	3501-6	5	4	$[0_5/\mathrm{m}/0_4]$	0	0.402	0.284	0.999	0
Rudd 2014	AS4	3501-6	2	2	$[0_2/\mathrm{m}/\mathrm{d}/0_2]$	1	0.410	0.290	1.021	1
Rudd 2014	AS4	3501-6	3	3	$[0_3/\mathrm{m}/\mathrm{d}/0_3]$	1	0.409	0.289	1.017	1
Rudd 2014	AS4	3501-6	3	2	$[0_3/\mathrm{m}/\mathrm{d}/0_2]$	1	0.409	0.289	1.013	1
Rudd 2014	AS4	3501-6	6	7	$[0_6/\mathrm{m}/0_7]$	0	0.402	0.284	1.002	0

Continued on next page

Table B1 – *Continued from previous page*

Source	Fiber Material	Matrix Material	Plies Above	Plies Below	Stacking Sequence	Terfenol-D	V_{peak}	V_{RMS}	$V_{RMS,adjusted}$	Delamination
Rudd 2014	AS4	3501-6	2	2	$[0_2/\mathrm{m}/\mathrm{d}/0_2]$	1	0.410	0.290	1.021	1
Rudd 2014	AS4	3501-6	3	2	$[0_3/\mathrm{m}/\mathrm{d}/0_2]$	1	0.407	0.288	1.010	1
Rudd 2014	AS4	3501-6	6	7	$[0_6/\mathrm{m}/\mathrm{d}/0_7]$	1	0.411	0.291	1.024	0
Rudd 2014	AS4	3501-6	4	4	$[0_4/\mathrm{m}/0_4]$	1	0.407	0.288	1.016	0
Rudd 2014	AS4	3501-6	1	1	$[0/\mathrm{m}/\mathrm{d}/0]$	1	0.410	0.290	1.017	1

Appendix C Machine Learning Models

A total of $1,249$ models were examined in this work for using eight algorithms found in the scikit-learn package [37] for Python. The generalized script presented in Appendix D.4 was modified to accommodate the various attributes manipulated for each algorithm.

Table C1: Summary of the manipulations made with each algorithm to yield the tested models.

Algorithm	Number of Models	Model Attribute(s) Changed
Discriminant Analysis (DA)	2	Linear DA (LDA), Quadratic DA (QDA)
Decision Tree (DT)	1	
Ensemble Method	2	Random Forest, Extra Trees
k-Nearest Neighbor (k-NN)	80	Algorithms: BallTree, KDTree, brute, auto; $k = 2 - 22$
Naïve Bayes (NB)	1	Gaussian distribution (GNB)
Stochastic Gradient Descent (SGD)	5	Loss functions: hinge, log, modified huber, squared hinge, perceptron
Support Vector Machine (SVM)	9	Classes: C-support vector classification (SVC), Nu-support vector classification (NuSVC), Linear support vector classification (LinearSVC); Kernel: linear, polynomial, radial basis function (rbf), sigmoid
Multi-layer Perceptron MLP	1,149	Activation functions: identity, logistic, tanh, rectified linear unit (ReLU); Solver function: proposed stochastic gradient descent (adam); Hidden layers: one (N_1 for $N_i = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 25, 30, 40, 50, 60, 70, 75, 80, 90, 100$), two ($N_1 - N_2$ for $N_i = 10, 20, 25, 30, 40, 50, 60, 70, 75, 80, 90, 100$) and three ($N_1 - N_2 - N_3$ for $N_i = 20, 40, 60, 80, 100$)

Appendix D Selected Python Scripts

D.1 Analytical Model Script

The iterative stress-calculating scheme presented in Krishnamurthy et al. [53] was scripted using Python. The script utilized the nonlinear relationships The Python packages discussed in [37, 125, 126, 169, 170] were employed in this script.

D.1.1 Characterization of Terfenol-D

```
import os
import numpy as np
import pandas as pd
import scipy as sp
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
from sklearn import metrics

import Error_Calculations


colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b', '#e377c2', '#7f7f7f', '#
    ↪ bcbd22', '#17becf']

stress_strain_labels_OG = ['H=0 kA/m','H=30 kA/m','H=60 kA/m','H=90 kA/m','H=120 kA/m']
stress_strain_labels = ['0 kA/m','30 kA/m','60 kA/m','90 kA/m','120 kA/m']
stress_strain_index = ['0','30','60','90','120']

strain_mf_labels = ['6.9 MPa','9.6 MPa','12.4 MPa','17.9 MPa','24.1 MPa', '38.0 MPa', '51.7 MPa']
prestress_index = ['6.9','9.6','12.4','17.9','24.1', '38.0', '51.7']

fourier3_coefficients = ['a0', 'a1', 'a2', 'a3', 'b1', 'b2', 'b3', 'omega']
fourier5_coefficients = ['a0', 'a1', 'a2', 'a3', 'a4', 'a5', 'b1', 'b2', 'b3', 'b4', 'b5', 'omega']
poly4_coefficients = ['a0', 'a1', 'a2', 'a3', 'a4']


def fourier1(x,a0,a1,b1,omega):
    return a0 + a1*np.cos(omega*x) + b1*np.sin(omega*x)

def fourier2(x,a0,a1,a2,b1,b2,omega):
    return a0 + a1*np.cos(omega*x) + b1*np.sin(omega*x) + a2*np.cos(2*omega*x) + b2*np.sin(2*omega*x)

def fourier3(x,a0,a1,a2,a3,b1,b2,b3,omega):
    return a0 + a1*np.cos(omega*x) + b1*np.sin(omega*x) + a2*np.cos(2*omega*x) + b2*np.sin(2*omega*x)
        ↪ + a3*np.cos(3*omega*x) + b3*np.sin(3*omega*x)

def fourier4(x,a0,a1,a2,a3,a4,b1,b2,b3,b4,omega):
    return a0 + a1*np.cos(omega*x) + b1*np.sin(omega*x) + a2*np.cos(2*omega*x) + b2*np.sin(2*omega*x)
        ↪ + a3*np.cos(3*omega*x) + b3*np.sin(3*omega*x) + a4*np.cos(4*omega*x) + b4*np.sin(4*omega*x
        ↪ )

def fourier5(x,a0,a1,a2,a3,a4,a5,b1,b2,b3,b4,b5,omega):
    return a0 + a1*np.cos(omega*x) + b1*np.sin(omega*x) + a2*np.cos(2*omega*x) + b2*np.sin(2*omega*x)
        ↪ + a3*np.cos(3*omega*x) + b3*np.sin(3*omega*x) + a4*np.cos(4*omega*x) + b4*np.sin(4*omega*x
        ↪ ) + a5*np.cos(5*omega*x) + b5*np.sin(5*omega*x)

def poly4(x,a0,a1,a2,a3,a4):
    return a0 + a1*x + a2*pow(x,2) + a3*pow(x,3) + a4*pow(x,4)
```

```
# input data folder
butler_main_folder = os.path.join(os.getcwd(),'Butler_Plots')
butler_plot_names = os.listdir(butler_main_folder)
plot_type = butler_plot_names[0]
print('...selecting file from >>> ' + plot_type)

bulter_plot_specific_folder = os.path.join(butler_main_folder,plot_type)
butler_plot_lines = os.listdir(bulter_plot_specific_folder)


# cycle through files for plot
i = 0
master_ss = pd.DataFrame(columns=fourier3_coefficients,index=stress_strain_index)
master_ss_sigma = pd.DataFrame(columns=fourier3_coefficients,index=stress_strain_index)
master_smf_poly4 = pd.DataFrame(columns=poly4_coefficients,index=prestress_index)
master_smf_fourier3 = pd.DataFrame(columns=fourier3_coefficients,index=prestress_index)
master_smf_fourier5 = pd.DataFrame(columns=fourier5_coefficients,index=prestress_index)

for files in butler_plot_lines:

    butler_line = butler_plot_lines[0]
    butler_line_filename = os.path.join(bulter_plot_specific_folder,files)
    df = pd.read_csv(butler_line_filename)
    header_list = list(df)


    # stress-strain data
    if 'stress' in plot_type.lower() and 'strain' in plot_type.lower():

        X_data = df[header_list[0]]*pow(10,-6)
        y_data = df[header_list[1]]*pow(10,6)

        plt.figure('Butler Stress-Strain Plot Data')
        plt.plot(X_data,y_data*pow(10,-6), color=colors[i], marker=None, linestyle='-')
        plt.xlim([0,2000e-6])
        plt.ylim([0,100e6*pow(10,-6)])
        plt.xlabel('Strain')
        plt.ylabel('Stress (MPa)')
        plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
        plt.legend(stress_strain_labels, loc='upper left')
        plt.tight_layout()
        plt.savefig('Plots/Stress-Strain/Butler_SSData.png')

        if 'line1' in files.lower():
            n = 1.818
            alpha_e = 52.287e-12
            beta_e = -1.218 * pow(10,-6 * n -6)
            y_kr_model = np.linspace(0,100e6,101)
            X_kr_model = alpha_e * y_kr_model + beta_e * pow(y_kr_model,n)

            df_krishnamurthy = pd.read_csv('Krishnamurthy_Exp.csv')
            X_kr_exp = df_krishnamurthy[list(df)[0]]*pow(10,-6)
            y_kr_exp = df_krishnamurthy[list(df)[1]]*pow(10,6)

            plt.figure('Krishnamurthy Ramberg-Osgood')
            plt.plot(X_data,y_data*pow(10,-6),'o')
            plt.plot(X_kr_exp, y_kr_exp*pow(10,-6),'s')
            plt.plot(X_kr_model, y_kr_model*pow(10,-6), color=colors[2], linestyle='--', linewidth=2)
            plt.xlim([0,1200e-6])
            plt.ylim([0,40e6*pow(10,-6)])
            plt.xlabel('Strain')
            plt.ylabel('Stress (MPa)')
            plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
```

```
    plt.legend(['Experimental Data (H$_{0}$=0 kA/m)','Krishnamurthy Experimental Data (H$_{0}$
        ↪ =0 kA/m)','Krishnamurthy Ramberg-Osgood Model'], loc='upper left')
    plt.tight_layout()
    plt.savefig('Plots/Stress-Strain/Krishnamurthy_Ramberg-Osgood.png')

    plt.figure('Krishnamurthy Ramberg-Osgood Divergence')
    plt.plot(X_data,y_data*pow(10,-6),'o')
    plt.plot(X_kr_model, y_kr_model*pow(10,-6), color=colors[2], linestyle='--', linewidth=2)
    plt.xlim([0,2000e-6])
    plt.ylim([0,100e6*pow(10,-6)])
    plt.xlabel('Strain')
    plt.ylabel('Stress (MPa)')
    plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
    plt.legend(['Experimental Data (H$_{0}$=0 kA/m)','Krishnamurthy Ramberg-Osgood Model'], loc
        ↪ ='lower right')
    plt.tight_layout()
    plt.savefig('Plots/Stress-Strain/Krishnamurthy_Ramberg-Osgood_Divergence.png')


    res = []
    for idx in range(0, len(y_data)) :
        if y_data[idx] > 40e6:
            res.append(idx)
    y_data_error = y_data[y_data < 40e6]
    y_data_error[0]=0

    X_butler_error = X_data[X_data < X_data[res[0]]]
    X_krButler_model_error = alpha_e * y_data_error + beta_e * pow(y_data_error,n)
    X_kr_exp_error = X_kr_exp
    X_kr_exp_error[0]=0
    X_krExp_model_error = alpha_e * X_kr_exp_error + beta_e * pow(X_kr_exp_error,n)


X_model = np.linspace(0,X_data.values[-1],101)
fs3coef, pcov = curve_fit(fourier3, X_data, y_data, maxfev=10000)
fs3coef_inv, pcov = curve_fit(fourier3, y_data, X_data, maxfev=10000, p0
    ↪ =[0,0,0,0,0,0,0,3.14298675609527e-08])


if 'line1' in files.lower():
    ys_kr_model = np.linspace(0,40e6,41)
    s_butler_exp = np.diff(X_butler_error) / np.diff(y_data_error)
    s_kr_exp = np.diff(X_kr_exp) / np.diff(y_kr_exp)
    s_kr_model = alpha_e + beta_e * n * pow(ys_kr_model,n-1)
    Xs_kr_model = alpha_e * ys_kr_model + beta_e * pow(ys_kr_model,n)
    s_fourier3 = np.diff(X_model) / np.diff(fourier3(X_model, *fs3coef))

    plt.figure('Compliance -H = 0 kA/m')
    plt.plot(y_data_error[:-1]*pow(10,-6),s_butler_exp,'s')
    plt.plot(y_kr_exp[:-1]*pow(10,-6),s_kr_exp,'o')
    plt.plot(ys_kr_model*pow(10,-6),s_kr_model)
    plt.plot(fourier3(X_model, *fs3coef)[:-1]*pow(10,-6),s_fourier3)
    plt.xlim([0,40e6*pow(10,-6)])
    plt.xlabel('Stress (MPa)')
    plt.ylabel('Compliance Term (1/Pa)')
    plt.legend(['Experimental Data (H$_{0}$=0 kA/m)','Krishnamurthy Experimental Data (H$_{0}$
        ↪ =0 kA/m)','Krishnamurthy Ramberg-Osgood Model','Three-term Fourier Series'], loc='
        ↪ upper right')
    plt.tight_layout()
    plt.savefig('Plots/Stress-Strain/Compliance_H0.png')


plt.figure('Stress-Strain Fourier3 Series')
plt.plot(X_data,y_data*pow(10,-6), color=colors[i], marker='o', linestyle='none', label=None)
```

```
plt.plot(X_model,fourier3(X_model, *fs3coef)*pow(10,-6),color=colors[i], linestyle='-', label=
    ↪ stress_strain_labels[i])
plt.xlim([0,2000e-6])
plt.ylim([0,100e6*pow(10,-6)])
plt.xlabel('Strain')
plt.ylabel('Stress (MPa)')
plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
plt.legend(loc='upper left')
plt.tight_layout()
plt.savefig('Plots/Stress-Strain/Fourier3_SScurvefit.png')

plt.figure('Fourier3 -sigma (experimental data) input')
plt.plot(X_data,y_data*pow(10,-6), color=colors[i], marker='o', linestyle='none', label=None)
plt.plot(fourier3(y_data, *fs3coef_inv),y_data*pow(10,-6), color=colors[i], marker=None,
    ↪ linestyle='-', label=stress_strain_labels[i])
plt.xlim([0,2000e-6])
plt.ylim([0,100e6*pow(10,-6)])
plt.xlabel('Strain')
plt.ylabel('Stress (MPa)')
plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
plt.legend(loc='upper left')
plt.tight_layout()
plt.savefig('Plots/Stress-Strain/Fourier3_sigma-expdata.png')

y_model = np.linspace(0,100e6,101)
plt.figure('Fourier3 -sigma (model data) input')
plt.plot(X_data,y_data*pow(10,-6), color=colors[i], marker='o', linestyle='none', label=None)
plt.plot(fourier3(y_model, *fs3coef_inv),y_model*pow(10,-6), color=colors[i], marker=None,
    ↪ linestyle='-', label=stress_strain_labels[i])
plt.xlim([0,2000e-6])
plt.ylim([0,100e6*pow(10,-6)])
plt.xlabel('Strain (x10e-6)')
plt.ylabel('Stress (MPa)')
plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
plt.legend(loc='upper left')
plt.tight_layout()
plt.savefig('Plots/Stress-Strain/Fourier3_sigma-model.png')


s_butler_exp = np.diff(X_data) / np.diff(y_data)
s_fourier3 = np.diff(X_model) / np.diff(fourier3(X_model, *fs3coef))

plt.figure('Compliance')
plt.plot(y_data[:-1]*pow(10,-6),s_butler_exp, color=colors[i], marker='o', linestyle='none',
    ↪ label=None)
plt.plot(fourier3(X_model, *fs3coef)[:-1]*pow(10,-6),s_fourier3, color=colors[i], linestyle='-
    ↪ ', label=stress_strain_labels[i])
plt.xlim([0,100e6*pow(10,-6)])
plt.xlabel('Stress (MPa)')
plt.ylabel('Compliance Term (1/Pa)')
plt.legend(loc='upper right')
plt.tight_layout()
plt.savefig('Plots/Stress-Strain/Compliance.png')


master_ss.loc[stress_strain_index[i]] = fs3coef
master_ss.to_csv('Stress-Strain_Fourier3_Coefficients.csv')

master_ss_sigma.loc[stress_strain_index[i]] = fs3coef_inv
master_ss_sigma.to_csv('Stress-Strain_Fourier3inv_Coefficients.csv')

i += 1
```

```
elif 'strain' in plot_type.lower() and 'magnetic' in plot_type.lower() and 'field' in plot_type.
    ↪ lower():

    X_model = np.linspace(0,1500,1501)
    X_data = df[header_list[0]]
    y_data = df[header_list[1]]*pow(10,-3)

    X_reduced=[]
    y_reduced=[]
    for j in range(0,len(X_data),3):
        X_reduced.append(X_data[j])
        y_reduced.append(y_data[j])

    X_reduced = np.asarray(X_reduced)
    y_reduced = np.asarray(y_reduced)

    plt.figure('Butler Strain-Magnetic Field Plot Data')
    plt.plot(X_data,y_data, color=colors[i], marker=None, linestyle='-')
    plt.xlim([0,1500])
    plt.xticks(np.linspace(0,1500,4))
    plt.yticks(np.linspace(0,1.5e-3,4))
    plt.ylim([0,1.5e-3])
    plt.xlabel('Magnetic Field Intensity (Oersted)')
    plt.ylabel('Strain')
    plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
    plt.legend(strain_mf_labels, loc='upper left')
    plt.tight_layout()
    plt.savefig('Plots/Strain-MagneticField/Butler_SMFData.png')


    df_krishnamurthy_poly4 = pd.read_csv('Krishnamurthy_SMFpoly4coef.csv')
    y_krpoly4 = df_krishnamurthy_poly4.loc[i][1] + df_krishnamurthy_poly4.loc[i][2] * X_model +
        ↪ df_krishnamurthy_poly4.loc[i][3] * pow(X_model,2) + df_krishnamurthy_poly4.loc[i][4] *
        ↪  pow(X_model,3) + df_krishnamurthy_poly4.loc[i][5] * pow(X_model,4)

    p4coef, pcov = curve_fit(poly4, X_data, y_data)
    fs3coef, pcov = curve_fit(fourier3, X_data, y_data,maxfev=10000,p0
        ↪ =[0,0,0,0,0,0,0,0.00209553137817916])
    fs5coef, pcov = curve_fit(fourier5, X_data, y_data,maxfev=10000,p0
        ↪ =[0,0,0,0,0,0,0,0,0,0,0,0.00209553137817916])


    plt.figure('Krishnamurthy Poly4')
    plt.plot(X_model,y_krpoly4)
    plt.xlim([0,1500])
    plt.ylim([0,1.500e3])
    plt.xticks(np.linspace(0,1500,4))
    plt.yticks(np.linspace(0,1.500e3,4))
    plt.xlabel('Magnetic Field Intensity (Oersted)')
    plt.ylabel('Strain')
    plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
    plt.legend(strain_mf_labels, loc='upper left')
    plt.tight_layout()
    plt.savefig('Plots/Strain-MagneticField/Krishnamurthy_Poly4.png')

    plt.figure('Krishnamurthy Poly4 + Experimental Data')
    plt.plot(X_reduced,y_reduced*pow(10,6), color=colors[i], marker='o', linestyle='none', label=
        ↪ None)
    plt.plot(X_model,y_krpoly4, color=colors[i], linestyle='-', label=strain_mf_labels[i])
    plt.xlim([0,1500])
    plt.ylim([0,1.500e3])
    plt.xticks(np.linspace(0,1500,4))
    plt.yticks(np.linspace(0,1.500e3,4))
    plt.xlabel('Magnetic Field Intensity (Oersted)')
```

```
plt.ylabel('Strain')
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
plt.legend(loc='upper left')
plt.tight_layout()
plt.savefig('Plots/Strain-MagneticField/Krishnamurthy_Poly4_ExpData.png')

plt.figure('Poly4 Curve-fit')
plt.plot(X_reduced,y_reduced, color=colors[i], marker='o', linestyle='none', label=None)
plt.plot(X_model, poly4(X_model, *p4coef), color=colors[i], linestyle='-', label=
    ↪ strain_mf_labels[i])
plt.xlim([0,1500])
plt.ylim([0,1.5e-3])
plt.xticks(np.linspace(0,1500,4))
plt.yticks(np.linspace(0,1.5e-3,4))
plt.xlabel('Magnetic Field Intensity (Oersted)')
plt.ylabel('Strain')
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
plt.legend(loc='upper left')
plt.tight_layout()
plt.savefig('Plots/Strain-MagneticField/Poly4_SMFcurvefit.png')

plt.figure('Fourier3 Curve-fit')
plt.plot(X_reduced,y_reduced, color=colors[i], marker='o', linestyle='none', label=None)
plt.plot(X_model, fourier3(X_model, *fs3coef), color=colors[i], linestyle='-', label=
    ↪ strain_mf_labels[i])
plt.xlim([0,1500])
plt.ylim([0,1.5e-3])
plt.xticks(np.linspace(0,1500,4))
plt.yticks(np.linspace(0,1.5e-3,4))
plt.xlabel('Magnetic Field Intensity (Oersted)')
plt.ylabel('Strain')
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
plt.legend(loc='upper left')
plt.tight_layout()
plt.savefig('Plots/Strain-MagneticField/Fourier3_SMFcurvefit.png')

plt.figure('Fourier5 Curve-fit')
plt.plot(X_reduced,y_reduced, color=colors[i], marker='o', linestyle='none', label=None)
plt.plot(X_model, fourier5(X_model, *fs5coef), color=colors[i], linestyle='-', label=
    ↪ strain_mf_labels[i])
plt.xlim([0,1500])
plt.ylim([0,1.5e-3])
plt.xticks(np.linspace(0,1500,4))
plt.yticks(np.linspace(0,1.5e-3,4))
plt.xlabel('Magnetic Field Intensity (Oersted)')
plt.ylabel('Strain')
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
plt.legend(loc='upper left')
plt.tight_layout()
plt.savefig('Plots/Strain-MagneticField/Fourier5_SMFcurvefit.png')

if '6.9' in files.lower():
    dc_butler_exp = np.diff(y_reduced) / np.diff(X_reduced)
    dc_krpoly4 = df_krishnamurthy_poly4.loc[i][2] + 2*df_krishnamurthy_poly4.loc[i][3] *
        ↪ X_model + 3*df_krishnamurthy_poly4.loc[i][4] * pow(X_model,2) + 4*
        ↪ df_krishnamurthy_poly4.loc[i][5] * pow(X_model,3)
    s_fourier5 = np.diff(fourier5(X_model, *fs5coef)) / np.diff(X_model)
    s_poly4 = np.diff(poly4(X_model, *p4coef)) / np.diff(X_model)

    plt.figure('Piezomagnetic Coefficient 6.9 MPa')
    plt.plot(X_reduced[:-1],dc_butler_exp,'s')
    plt.plot(X_model,dc_krpoly4*pow(10,-6))
    plt.plot(X_model[:-1],s_poly4)
    plt.plot(X_model[:-1],s_fourier5)
```

```
    plt.xlim([0,1500])
    plt.ylim([0,4e-6])
    plt.xticks(np.linspace(0,1500,4))
    plt.yticks(np.linspace(0,4e-6,9))
    plt.xlabel('Magnetic Field Intensity (Oersted)')
    plt.ylabel('Piezomagnetic Coefficient (m/A)')
    plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
    plt.legend(['Experimental Data ('r'$\sigma_0$''=6.9 MPa)','Krishnamurthy Polynomial (x10e-
        ↪ 6)','Proposed Quartic Polynomial','Five-term Fourier Series'], loc='upper right')
    plt.tight_layout()
    plt.savefig('Plots/Strain-MagneticField/PiezomagneticCoefficient_6-9MPa.png')


dc_butler_exp = np.diff(y_reduced) / np.diff(X_reduced)
s_fourier5 = np.diff(fourier5(X_model, *fs5coef)) / np.diff(X_model)
s_poly4 = np.diff(poly4(X_model, *p4coef)) / np.diff(X_model)

plt.figure('Piezomagnetic Coefficient Fourier5')
plt.plot(X_reduced[:-1],dc_butler_exp, color=colors[i], marker='o', linestyle='none', label=
    ↪ None)
plt.plot(X_model[:-1],s_fourier5, color=colors[i], linestyle='-', label=strain_mf_labels[i])
plt.xlim([0,1500])
plt.ylim([0,3.5e-6])
plt.xticks(np.linspace(0,1500,4))
plt.yticks(np.linspace(0,3.5e-6,8))
plt.xlabel('Magnetic Field Intensity (Oersted)')
plt.ylabel('Piezomagnetic Coefficient (m/A)')
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
plt.legend(loc='upper right')
plt.tight_layout()
plt.savefig('Plots/Strain-MagneticField/PiezomagneticCoefficient_fourier5.png')

plt.figure('Piezomagnetic Coefficient Poly4')
plt.plot(X_reduced[:-1],dc_butler_exp, color=colors[i], marker='o', linestyle='none', label=
    ↪ None)
plt.plot(X_model[:-1],s_poly4, color=colors[i], linestyle='-', label=strain_mf_labels[i])
plt.xlim([0,1500])
plt.ylim([0,3.5e-6])
plt.xticks(np.linspace(0,1500,4))
plt.yticks(np.linspace(0,3.5e-6,8))
plt.xlabel('Magnetic Field Intensity (Oersted)')
plt.ylabel('Piezomagnetic Coefficient (m/A)')
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
plt.legend(loc='upper right')
plt.tight_layout()
plt.savefig('Plots/Strain-MagneticField/PiezomagneticCoefficient_poly4.png')



master_smf_poly4.loc[prestress_index[i]] = p4coef
master_smf_fourier3.loc[prestress_index[i]] = fs3coef
master_smf_fourier5.loc[prestress_index[i]] = fs5coef
master_smf_poly4.to_csv('Strain-MagneticField_Poly4_Coefficients.csv')
master_smf_fourier3.to_csv('Strain-MagneticField_Fourier3_Coefficients.csv')
master_smf_fourier5.to_csv('Strain-MagneticField_Fourier5_Coefficients.csv')

i += 1
```

D.1.2 Iterative Stress Scheme

```
import os
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from bulkmaterialprop import as4_35016,terfenolD
from interpolate_EBmodel_Krishnamurthy import *


################################################
# input parameters
H0 = 0
H = 10
################################################


plies_above = 4
plies_below = 4
plies_ms = 1
num_plies = plies_above + plies_ms + plies_below

ss_angle = [0] * (plies_above + plies_ms + plies_below)
ss_material = ['cfrp'] * plies_above + ['ms'] * plies_ms + ['cfrp'] * plies_below


total_thickness = (plies_above + plies_below) * as4_35016.thickness + plies_ms * terfenolD.thickness
total_thickness = round(total_thickness,10)
z = []
z.append(-0.5 * total_thickness)

A1 = 0
A2 = 0
mslayer = 0
for i in range(0,num_plies):
    if 'cfrp' in ss_material[i] and mslayer == 0:
        new_z = z[i] + as4_35016.thickness
        A1 = A1 + as4_35016.E1 * (new_z -z[i])
    elif 'ms' in ss_material[i]:
        new_z = z[i] + terfenolD.thickness
        mslayer = 1
    elif 'cfrp' in ss_material[i] and mslayer == 1:
        new_z = z[i] + as4_35016.thickness
        A2 = A2 + as4_35016.E1 * (new_z -z[i])
    z.append(round(new_z,10))


n = 1.818
alpha_e = 52.287e-12
beta_e = -1.218 * pow(10,-6 * n -6)

sigmaMax = 55
step = 0.1
sigma = np.arange(0,sigmaMax+step,step) * pow(10,6)
sigma_tr = sigma + step

LHS2_fourier3 = []
RHS1_krpoly4 = []
RHS2_krpoly4 = []
RHS1_npoly4 = []
RHS2_npoly4 = []
RHS_nfourier5 = []

LHS1 = np.asarray((sigma_tr * terfenolD.thickness) / (A1 + A2))
LHS2_kr = np.asarray(alpha_e * sigma_tr + beta_e * pow(sigma_tr,n))
```

```
for i in range(0,len(sigma)):
    LHS2_fourier3.append(SSfourier3_sum(sigma_tr[i],H0))
    esp_Krpoly4, esp_Krpoly4_a0 = interpolateKrPoly4(sigma[i],H)
    esp_new_poly4, esp_new_poly4_a0 = interpolateNewPoly4(sigma[i],H)
    esp_new_fourier5 = interpolateNewFourier5(sigma[i],H)

    RHS1_krpoly4.append(esp_Krpoly4)
    RHS2_krpoly4.append(esp_Krpoly4_a0)
    RHS1_npoly4.append(esp_new_poly4)
    RHS2_npoly4.append(esp_new_poly4_a0)
    RHS_nfourier5.append(esp_new_fourier5)


LHS2_fourier3 = np.asarray(LHS2_fourier3)
RHS1_krpoly4 = np.asarray(RHS1_krpoly4)
RHS2_krpoly4 = np.asarray(RHS2_krpoly4)
RHS1_npoly4 = np.asarray(RHS1_npoly4)
RHS2_npoly4 = np.asarray(RHS2_npoly4)
RHS_nfourier5 = np.asarray(RHS_nfourier5)


LHS_kr = np.asarray(LHS1) -np.asarray(LHS2_kr)
LHS_fourier3 = np.asarray(LHS1) -np.asarray(LHS2_fourier3)

diff1_kr = LHS_kr -RHS1_krpoly4
diff2_kr = LHS_kr -RHS2_krpoly4
diff1_fourier3 = LHS_fourier3 -RHS1_krpoly4
diff2_fourier3 = LHS_fourier3 -RHS2_krpoly4


plt.figure('EB -Iteration Krishnamurthy')
plt.plot(sigma*pow(10,-6),LHS_kr)
plt.plot(sigma*pow(10,-6),RHS1_krpoly4)
plt.plot(sigma*pow(10,-6),RHS2_krpoly4)
plt.xlim([0,sigmaMax])
plt.xticks(np.linspace(0,sigmaMax,12))
plt.xlabel('Stress (MPa)')
plt.ylabel('Strain')
plt.legend(['LHS','RHS (without 'r'$\alpha_0$'')','RHS (with 'r'$\alpha_0$'')'], loc='upper right')
plt.tight_layout()
plt.savefig('Plots/Krishnamurthy_EBModel/EB_model_KR.png')

plt.figure('EB -Iteration Proposed Models')
plt.plot(sigma*pow(10,-6),LHS_fourier3)
plt.plot(sigma*pow(10,-6),RHS1_npoly4)
plt.plot(sigma*pow(10,-6),RHS2_npoly4)
plt.xlim([0,sigmaMax])
plt.ylim([-1.4e-3,0.6e-3])
plt.xticks(np.linspace(0,sigmaMax,12))
plt.xlabel('Stress (MPa)')
plt.ylabel('Strain')
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
plt.legend(['LHS','RHS (without 'r'$\alpha_0$'')','RHS (with 'r'$\alpha_0$'')'], loc='upper right')
plt.tight_layout()
plt.savefig('Plots/Krishnamurthy_EBModel/EB_model_poly4.png')

plt.figure('EB -Iteration Proposed Fourier Models')
plt.plot(sigma*pow(10,-6),LHS_fourier3)
plt.plot(sigma*pow(10,-6),RHS_nfourier5)
plt.xlim([0,sigmaMax])
plt.ylim([-1.4e-3,0.4e-3])
plt.xticks(np.linspace(0,sigmaMax,12))
plt.xlabel('Stress (MPa)')
plt.ylabel('Strain')
```

```
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
plt.legend(['LHS','RHS'], loc='upper right')
plt.tight_layout()
plt.savefig('Plots/Krishnamurthy_EBModel/EB_model_fourier.png')

ymax = 0.2e-3
ymin = -1.4e-3
plt.figure('EB -Iteration ALL Proposed Models')
plt.plot(sigma*pow(10,-6),LHS_fourier3)
plt.plot(sigma*pow(10,-6),RHS1_npoly4)
plt.plot(sigma*pow(10,-6),RHS2_npoly4)
plt.plot(sigma*pow(10,-6),RHS_nfourier5)
plt.xlim([0,sigmaMax])
plt.ylim([ymin,ymax])
plt.xticks(np.linspace(0,sigmaMax,12))
plt.yticks(np.linspace(ymin,ymax,9))
plt.xlabel('Stress (MPa)')
plt.ylabel('Strain')
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
plt.legend(['LHS','RHS -poly4 (without 'r'$\alpha_0$'')','RHS -poly4 (with 'r'$\alpha_0$'')','RHS -
    ↪ Fourier5'], loc='lower left')
plt.tight_layout()
plt.savefig('Plots/Krishnamurthy_EBModel/EB_model.png')

ymax = 10e-5
ymin = -12e-5
plt.figure('EB -Iteration ALL Proposed Models RHS')
plt.plot(sigma*pow(10,-6),LHS_fourier3)
plt.plot(sigma*pow(10,-6),RHS1_npoly4)
plt.plot(sigma*pow(10,-6),RHS2_npoly4)
plt.plot(sigma*pow(10,-6),RHS_nfourier5)
plt.xlim([0,sigmaMax])
plt.ylim([ymin,ymax])
plt.xticks(np.linspace(0,sigmaMax,12))
plt.yticks(np.linspace(ymin,ymax,9))
plt.xlabel('Stress (MPa)')
plt.ylabel('Strain')
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
plt.legend(['LHS','RHS -poly4 (without 'r'$\alpha_0$'')','RHS -poly4 (with 'r'$\alpha_0$'')','RHS -
    ↪ Fourier5'], loc='lower right')
plt.tight_layout()
plt.savefig('Plots/Krishnamurthy_EBModel/EB_model_RHS.png')


plt.figure('LHS terms')
plt.plot(sigma*pow(10,-6),LHS1,'-')
plt.plot(sigma*pow(10,-6),LHS2_kr,'-')
plt.plot(sigma*pow(10,-6),LHS2_fourier3,'-')
plt.xlim([0,sigmaMax])
plt.ylim([0,1.4e-3])
plt.xticks(np.linspace(0,sigmaMax,12))
plt.xlabel('Stress (MPa)')
plt.ylabel('Strain')
plt.legend(['LHS1 (stiffness term)','Ramberg-Osgood','Fourier3'], loc='upper left')
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
plt.tight_layout()
plt.savefig('Plots/Krishnamurthy_EBModel/EB_LHSstrain.png')

plt.figure('LHS terms total')
plt.plot(sigma*pow(10,-6),LHS_kr,'-')
plt.plot(sigma*pow(10,-6),LHS_fourier3,'-')
plt.xlim([0,sigmaMax])
plt.ylim([-1.4e-3,0])
plt.xticks(np.linspace(0,sigmaMax,12))
```

```
plt.xlabel('Stress (MPa)')
plt.ylabel('Strain')
plt.legend(['Ramberg-Osgood','Fourier3'], loc='upper right')
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
plt.tight_layout()
plt.savefig('Plots/Krishnamurthy_EBModel/EB_LHSstrain_total.png')

plt.figure('Difference in Poly4')
plt.plot(sigma*pow(10,-6),RHS1_krpoly4)
plt.plot(sigma*pow(10,-6),RHS2_krpoly4)
plt.xlim([0,sigmaMax])
plt.ylim([-55,55])
plt.xticks(np.linspace(0,sigmaMax,12))
plt.xlabel('Stress (MPa)')
plt.ylabel('Strain')
plt.legend(['Krishnamurthy Poly4 (without 'r'$\alpha_0$'')','Krishnamurthy Poly4 (with 'r'$\alpha_0$''
    ↪ )'], loc='upper right')
plt.tight_layout()
plt.savefig('Plots/Krishnamurthy_EBModel/KrPoly4_Comparison.png')
```

D.1.3 Interpolation of the Magnetic Strain Relationship Coefficients

A linear interpolation method was applied to calculate the coefficients of the strain-magnetic field intensity relationship. The functions examined in this script include the polynomial presented in [53], an improved polynomial, and two Fourier series (three and five term series). The linear interpolation scheme was taken from [53].

$$\alpha_i = \alpha_{i1} \quad \sigma < \sigma_1 \tag{D.16}$$

$$\alpha_{ij}(\sigma) = \alpha_{ij} + \frac{\alpha_{i(j+1)} - \alpha_{ij}}{\sigma_{j+1} - \sigma_j}(\sigma - \sigma_j) \quad \sigma_j < \sigma < \sigma_{j+1} \tag{D.17}$$

$$i = 0, 1, 2, 3, 4 \quad j = 1, 2, 3, 4, 5, 6, 7 \tag{D.18}$$

```
import numpy as np
import pandas as pd


def SSfourier3_sum(stress,H0):
    df_ss_fourier3 = pd.read_csv('Stress-Strain_Fourier3inv_Coefficients.csv')
    bias_dict = {0:0, 30:1, 60:2, 90:3, 120:4}
    stress_strain_index = ['0','30','60','90','120']

    df2_row = df_ss_fourier3.iloc[[bias_dict[H0]]]
    a0 = df2_row['a0'][0]
    a1 = df2_row['a1'][0]
    a2 = df2_row['a2'][0]
    a3 = df2_row['a3'][0]
    b1 = df2_row['b1'][0]
    b2 = df2_row['b2'][0]
    b3 = df2_row['b3'][0]
    omega = df2_row['omega'][0]

    sum = a0 + a1*np.cos(omega*stress) + b1*np.sin(omega*stress) + a2*np.cos(2*omega*stress) + b2*np.
        ↪ sin(2*omega*stress) + a3*np.cos(3*omega*stress) + b3*np.sin(3*omega*stress)
    return sum
```

```
def interpolateKrPoly4(stress,H):
    df_kr_poly4 = pd.read_csv('Krishnamurthy_SMFpoly4coef.csv')
    prestress_index = ['6.9','9.6','12.4','17.9','24.1', '38.0', '51.7']
    prestress = [6.9, 9.6, 12.4, 17.9, 24.1, 38.0, 51.7]

    stress = round(stress * pow(10,-6),1)
    stress_index = df_kr_poly4.index[df_kr_poly4['stress'] == stress].tolist()

    if stress_index:
        print('Prestress level: ' + prestress_index[stress_index[0]])
        a0 = df_kr_poly4['a0'][stress_index]
        a1 = df_kr_poly4['a1'][stress_index]
        a2 = df_kr_poly4['a2'][stress_index]
        a3 = df_kr_poly4['a3'][stress_index]
        a4 = df_kr_poly4['a4'][stress_index]

    elif stress < prestress[0]:
        print('below zero')
        a0 = df_kr_poly4['a0'][0]
        a1 = df_kr_poly4['a1'][0]
        a2 = df_kr_poly4['a2'][0]
        a3 = df_kr_poly4['a3'][0]
        a4 = df_kr_poly4['a4'][0]

    elif stress > prestress[6]:
        print('Maximum stress exceeded.')
        a0 = 0
        a1 = 0
        a2 = 0
        a3 = 0
        a4 = 0

    elif not stress_index:
        print('...interpolating...')
        for i in range(0,6):
            is_between = prestress[i] <= stress <= prestress[i+1]

            if is_between:
                a0 = df_kr_poly4['a0'][i] + ((df_kr_poly4['a0'][i+1] -df_kr_poly4['a0'][i]) / (
                    ↪ prestress[i+1] -prestress[i])) * (stress -prestress[i])
                a1 = df_kr_poly4['a1'][i] + ((df_kr_poly4['a1'][i+1] -df_kr_poly4['a1'][i]) / (
                    ↪ prestress[i+1] -prestress[i])) * (stress -prestress[i])
                a2 = df_kr_poly4['a2'][i] + ((df_kr_poly4['a2'][i+1] -df_kr_poly4['a2'][i]) / (
                    ↪ prestress[i+1] -prestress[i])) * (stress -prestress[i])
                a3 = df_kr_poly4['a3'][i] + ((df_kr_poly4['a3'][i+1] -df_kr_poly4['a3'][i]) / (
                    ↪ prestress[i+1] -prestress[i])) * (stress -prestress[i])
                a4 = df_kr_poly4['a4'][i] + ((df_kr_poly4['a4'][i+1] -df_kr_poly4['a4'][i]) / (
                    ↪ prestress[i+1] -prestress[i])) * (stress -prestress[i])


    epsilonH = a1 * H + a2 * pow(H,2) + a3 * pow(H,3) + a4 * pow(H,4)
    epsilonH_a0 = a0 + a1 * H + a2 * pow(H,2) + a3 * pow(H,3) + a4 * pow(H,4)
    return(epsilonH, epsilonH_a0)


def interpolateNewPoly4(stress,H):
    df_smf_poly4 = pd.read_csv('Strain-MagneticField_Poly4_Coefficients.csv')
    prestress_index = ['6.9','9.6','12.4','17.9','24.1', '38.0', '51.7']
    prestress = [6.9, 9.6, 12.4, 17.9, 24.1, 38.0, 51.7]

    stress = round(stress * pow(10,-6),1)
    stress_index = df_smf_poly4.index[df_smf_poly4['stress'] == stress].tolist()
```

```
    if stress_index:
        print('Prestress level: ' + prestress_index[stress_index[0]])
        a0 = df_smf_poly4['a0'][stress_index]
        a1 = df_smf_poly4['a1'][stress_index]
        a2 = df_smf_poly4['a2'][stress_index]
        a3 = df_smf_poly4['a3'][stress_index]
        a4 = df_smf_poly4['a4'][stress_index]

    elif stress < prestress[0]:
        print('below zero')
        a0 = df_smf_poly4['a0'][0]
        a1 = df_smf_poly4['a1'][0]
        a2 = df_smf_poly4['a2'][0]
        a3 = df_smf_poly4['a3'][0]
        a4 = df_smf_poly4['a4'][0]

    elif stress > prestress[6]:
        print('Maximum stress exceeded.')
        a0 = 0
        a1 = 0
        a2 = 0
        a3 = 0
        a4 = 0

    elif not stress_index:
        print('...interpolating...')
        for i in range(0,6):
            is_between = prestress[i] <= stress <= prestress[i+1]

            if is_between:
                a0 = df_smf_poly4['a0'][i] + ((df_smf_poly4['a0'][i+1] -df_smf_poly4['a0'][i]) / (
                    ↪ prestress[i+1] -prestress[i])) * (stress -prestress[i])
                a1 = df_smf_poly4['a1'][i] + ((df_smf_poly4['a1'][i+1] -df_smf_poly4['a1'][i]) / (
                    ↪ prestress[i+1] -prestress[i])) * (stress -prestress[i])
                a2 = df_smf_poly4['a2'][i] + ((df_smf_poly4['a2'][i+1] -df_smf_poly4['a2'][i]) / (
                    ↪ prestress[i+1] -prestress[i])) * (stress -prestress[i])
                a3 = df_smf_poly4['a3'][i] + ((df_smf_poly4['a3'][i+1] -df_smf_poly4['a3'][i]) / (
                    ↪ prestress[i+1] -prestress[i])) * (stress -prestress[i])
                a4 = df_smf_poly4['a4'][i] + ((df_smf_poly4['a4'][i+1] -df_smf_poly4['a4'][i]) / (
                    ↪ prestress[i+1] -prestress[i])) * (stress -prestress[i])


    epsilonH = a1 * H + a2 * pow(H,2) + a3 * pow(H,3) + a4 * pow(H,4)
    epsilonH_a0 = a0 + a1 * H + a2 * pow(H,2) + a3 * pow(H,3) + a4 * pow(H,4)
    return(epsilonH, epsilonH_a0)


def interpolateNewFourier5(stress,H):
    df_smf_fourier5 = pd.read_csv('Strain-MagneticField_Fourier5_Coefficients.csv')
    prestress_index = ['6.9','9.6','12.4','17.9','24.1', '38.0', '51.7']
    prestress = [6.9, 9.6, 12.4, 17.9, 24.1, 38.0, 51.7]

    stress = round(stress * pow(10,-6),1)
    stress_index = df_smf_fourier5.index[df_smf_fourier5['stress'] == stress].tolist()

    if stress_index:
        print('Prestress level: ' + prestress_index[stress_index[0]])
        a0 = df_smf_fourier5['a0'][stress_index]
        a1 = df_smf_fourier5['a1'][stress_index]
        a2 = df_smf_fourier5['a2'][stress_index]
        a3 = df_smf_fourier5['a3'][stress_index]
        a4 = df_smf_fourier5['a4'][stress_index]
        a5 = df_smf_fourier5['a5'][stress_index]
        b1 = df_smf_fourier5['b1'][stress_index]
```

```
    b2 = df_smf_fourier5['b2'][stress_index]
    b3 = df_smf_fourier5['b3'][stress_index]
    b4 = df_smf_fourier5['b4'][stress_index]
    b5 = df_smf_fourier5['b5'][stress_index]
    omega = df_smf_fourier5['omega'][stress_index]


elif stress < prestress[0]:
    print('below zero')
    a0 = df_smf_fourier5['a0'][0]
    a1 = df_smf_fourier5['a1'][0]
    a2 = df_smf_fourier5['a2'][0]
    a3 = df_smf_fourier5['a3'][0]
    a4 = df_smf_fourier5['a4'][0]
    a5 = df_smf_fourier5['a5'][0]
    b1 = df_smf_fourier5['b1'][0]
    b2 = df_smf_fourier5['b2'][0]
    b3 = df_smf_fourier5['b3'][0]
    b4 = df_smf_fourier5['b4'][0]
    b5 = df_smf_fourier5['b5'][0]
    omega = df_smf_fourier5['omega'][0]


elif stress > prestress[6]:
    print('Maximum stress exceeded.')
    a0 = 0
    a1 = 0
    a2 = 0
    a3 = 0
    a4 = 0
    a5 = 0
    b1 = 0
    b2 = 0
    b3 = 0
    b4 = 0
    b5 = 0
    omega = 0


elif not stress_index:
    print('...interpolating...')
    for i in range(0,6):
        is_between = prestress[i] <= stress <= prestress[i+1]

        if is_between:
            a0 = df_smf_fourier5['a0'][i] + ((df_smf_fourier5['a0'][i+1] -df_smf_fourier5['a0'][i])
                ↪  / (prestress[i+1] -prestress[i])) * (stress -prestress[i])
            a1 = df_smf_fourier5['a1'][i] + ((df_smf_fourier5['a1'][i+1] -df_smf_fourier5['a1'][i])
                ↪  / (prestress[i+1] -prestress[i])) * (stress -prestress[i])
            a2 = df_smf_fourier5['a2'][i] + ((df_smf_fourier5['a2'][i+1] -df_smf_fourier5['a2'][i])
                ↪  / (prestress[i+1] -prestress[i])) * (stress -prestress[i])
            a3 = df_smf_fourier5['a3'][i] + ((df_smf_fourier5['a3'][i+1] -df_smf_fourier5['a3'][i])
                ↪  / (prestress[i+1] -prestress[i])) * (stress -prestress[i])
            a4 = df_smf_fourier5['a4'][i] + ((df_smf_fourier5['a4'][i+1] -df_smf_fourier5['a4'][i])
                ↪  / (prestress[i+1] -prestress[i])) * (stress -prestress[i])
            a5 = df_smf_fourier5['a5'][i] + ((df_smf_fourier5['a5'][i+1] -df_smf_fourier5['a5'][i])
                ↪  / (prestress[i+1] -prestress[i])) * (stress -prestress[i])
            b1 = df_smf_fourier5['b1'][i] + ((df_smf_fourier5['b1'][i+1] -df_smf_fourier5['b1'][i])
                ↪  / (prestress[i+1] -prestress[i])) * (stress -prestress[i])
            b2 = df_smf_fourier5['b2'][i] + ((df_smf_fourier5['b2'][i+1] -df_smf_fourier5['b2'][i])
                ↪  / (prestress[i+1] -prestress[i])) * (stress -prestress[i])
            b3 = df_smf_fourier5['b3'][i] + ((df_smf_fourier5['b3'][i+1] -df_smf_fourier5['b3'][i])
                ↪  / (prestress[i+1] -prestress[i])) * (stress -prestress[i])
```

```
            b4 = df_smf_fourier5['b4'][i] + ((df_smf_fourier5['b4'][i+1] -df_smf_fourier5['b4'][i])
                ↪  / (prestress[i+1] -prestress[i])) * (stress -prestress[i])
            b5 = df_smf_fourier5['b5'][i] + ((df_smf_fourier5['b5'][i+1] -df_smf_fourier5['b5'][i])
                ↪  / (prestress[i+1] -prestress[i])) * (stress -prestress[i])
            omega = df_smf_fourier5['omega'][i] + ((df_smf_fourier5['omega'][i+1] -df_smf_fourier5[
                ↪ 'omega'][i]) / (prestress[i+1] -prestress[i])) * (stress -prestress[i])

    epsilonH = a0 + a1*np.cos(omega*H) + b1*np.sin(omega*H) + a2*np.cos(2*omega*H) + b2*np.sin(2*omega
        ↪ *H) + a3*np.cos(3*omega*H) + b3*np.sin(3*omega*H) + a4*np.cos(4*omega*H) + b4*np.sin(4*
        ↪ omega*H) + a5*np.cos(5*omega*H) + b5*np.sin(5*omega*H)
    return epsilonH
```

D.2 Effective Property Scripts

D.2.1 Effective Elastic Property Calculations

```
import numpy as np

# voigt approximation
def voigt(propertyA,propertyB,vfA,vfB):
    effective_property = propertyA * vfA + propertyB * vfB
    return effective_property

# reuss approximation
def reuss(propertyA,propertyB,vfA,vfB):
    inv = (vfA / propertyA) + (vfB / propertyB)
    effective_property = 1 / inv
    return effective_property

# hashin-shtrikman (lower bound)
def hashin_shtrikman_lower(K_td,G_td,K_matrix,G_matrix,vf_td,vf_matrix):
    K = K_matrix + np.divide(vf_td,((1 / (K_td -K_matrix)) + (vf_matrix / (K_matrix + G_matrix))))
    G = G_matrix + np.divide(vf_td,(1 / (G_td -G_matrix)) + ((vf_matrix * (K_matrix + 2 * G_matrix)) /
        ↪  (2 * G_matrix * (K_matrix + G_matrix))))

    E_effective = (9 * K * G) / (3 * K + G)
    NU_effective = (3 * K -2 * G) / (2 * (3 * K + G))
    G_effective = E_effective / (2 * (1 + NU_effective))
    K_effective = E_effective / (3 * (1 -2 * NU_effective))
    return E_effective, NU_effective, G_effective, K_effective

# hashin-shtrikman (upper bound)
def hashin_shtrikman_upper(K_td,G_td,K_matrix,G_matrix,vf_td,vf_matrix):
    K = K_td + np.divide(vf_matrix,((1 / (K_matrix -K_td)) + (vf_td / (K_td + G_td))))
    G = G_td + np.divide(vf_matrix,(1 / (G_matrix -G_td)) + ((vf_td * (K_td + 2 * G_td)) / (2 * G_td *
        ↪  (K_td + G_td))))

    E_effective = (9 * K * G) / (3 * K + G)
    NU_effective = (3 * K -2 * G) / (2 * (3 * K + G))
    G_effective = E_effective / (2 * (1 + NU_effective))
    K_effective = E_effective / (3 * (1 -2 * NU_effective))
    return E_effective, NU_effective, G_effective, K_effective


# mori-tanaka
def mori_tanaka(K_td,G_td,K_matrix,G_matrix,vf_td,vf_matrix):
    K = (K_td -K_matrix) * (vf_td / (1 + vf_matrix * ((K_td -K_matrix) / (K_matrix + (4/3) * G_matrix)
        ↪ ))) + K_matrix
    G = (G_td -G_matrix) * (vf_td / (1 + vf_matrix * ((G_td -G_matrix) / (G_matrix + ((G_matrix * (9 *
        ↪  K_matrix + 8 * G_matrix)) / (6 * (K_matrix + 2 * G_matrix))))))) + G_matrix
```

```
    E_effective = (9 * K * G) / (3 * K + G)
    NU_effective = (3 * K -2 * G) / (2 * (3 * K + G))
    G_effective = G
    K_effective = K

    return E_effective, NU_effective, G_effective, K_effective
```

D.2.2 Effective Elastic Properties

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from bulkmaterialprop import matrix35016
from bulkmaterialprop import terfenolD
import elastic_property_approx as elastic_properties

class Eeff:
    voigt = []

class Geff:
    voigt = []

class NUeff:
    voigt = []

class Keff:
    voigt = []

# set volume fraction
vf_td = np.linspace(0,1,101)
vf_matrix = 1 -vf_td

## approximations
elastic_property_labels = ['Voigt', 'Reuss', 'Hill', 'Hashin-Shtrikman (Upper Bound)', 'Hashin-
    ↪ Shtrikman (Lower Bound)', 'Mori-Tanaka']
elastic_property_labels2 = ['Voigt -elasticity','Voigt -verbatim','Reuss -elasticity','Reuss -verbatim
    ↪ ', 'Hill', 'Hashin-Shtrikman (Upper Bound)', 'Hashin-Shtrikman (Lower Bound)', 'Mori-Tanaka']

# VOIGT
Geff.voigt = elastic_properties.voigt(terfenolD.G,matrix35016.G,vf_td,vf_matrix)
Keff.voigt = elastic_properties.voigt(terfenolD.K,matrix35016.K,vf_td,vf_matrix)
Eeff.voigt_elasticity = (9 * Keff.voigt * Geff.voigt) / (3 * Keff.voigt + Geff.voigt)
NUeff.voigt_elasticity = (3 * Keff.voigt -2 * Geff.voigt) / (2 * (3 * Keff.voigt + Geff.voigt))
Eeff.voigt = elastic_properties.voigt(terfenolD.E,matrix35016.E,vf_td,vf_matrix)
NUeff.voigt = elastic_properties.voigt(terfenolD.nu,matrix35016.nu,vf_td,vf_matrix)


# REUSS
Geff.reuss = elastic_properties.reuss(terfenolD.G,matrix35016.G,vf_td,vf_matrix)
Keff.reuss = elastic_properties.reuss(terfenolD.K,matrix35016.K,vf_td,vf_matrix)
Eeff.reuss_elasticity = (9 * Keff.reuss * Geff.reuss) / (3 * Keff.reuss + Geff.reuss)
NUeff.reuss_elasticity = (3 * Keff.reuss -2 * Geff.reuss) / (6 * Keff.reuss + 2 * Geff.reuss)
Eeff.reuss = elastic_properties.reuss(terfenolD.E,matrix35016.E,vf_td,vf_matrix)
NUeff.reuss = elastic_properties.reuss(terfenolD.nu,matrix35016.nu,vf_td,vf_matrix)


# HILL
Geff.hill = (Geff.voigt + Geff.reuss) / 2
```

```
Keff.hill = (Keff.voigt + Keff.reuss) / 2
Eeff.hill = (Eeff.voigt + Eeff.reuss) / 2
NUeff.hill = (NUeff.voigt + NUeff.reuss) / 2


# HASHIN-SHTRIKMAN
# # # upper bound
Eeff.hsub, NUeff.hsub, Geff.hsub, Keff.hsub = elastic_properties.hashin_shtrikman_upper(terfenolD.K,
    ↪ terfenolD.G,matrix35016.K,matrix35016.G,vf_td,vf_matrix)

# # # lower bound
Eeff.hslb, NUeff.hslb, Geff.hslb, Keff.hslb = elastic_properties.hashin_shtrikman_lower(terfenolD.K,
    ↪ terfenolD.G,matrix35016.K,matrix35016.G,vf_td,vf_matrix)


# MORI-TANAKA
Eeff.mori_tanaka, NUeff.mori_tanaka, Geff.mori_tanaka, Keff.mori_tanaka = elastic_properties.
    ↪ mori_tanaka(terfenolD.K,terfenolD.G,matrix35016.K,matrix35016.G,vf_td,vf_matrix)


## EXPORT

K = np.array([Keff.voigt, Keff.reuss, Keff.hill, Keff.hsub, Keff.hslb, Keff.mori_tanaka])
K = np.transpose(K)

K = pd.DataFrame(K, columns=elastic_property_labels, index=vf_td)
K.to_csv('K_approximations.csv')


vf_td_reduced=[]
Eeff_hsub_reduced=[]
for j in range(0,len(vf_td_reduced),3):
    vf_td_reduced.append(vf_td_reduced[j])
    Eeff_hsub_reduced.append(Eeff.hsub[j])

vf_td_reduced = np.asarray(vf_td_reduced)
Eeff_hsub_reduced = np.asarray(Eeff_hsub_reduced)


## EFFECTIVE PROPERTY PLOTS

AR = 8/5
width = 12
fig_size = [width, width/AR]
text_label_x = 0.602
ymin_value = 0

# elastic modulus -elasticity
ymax_value = int(35)
plt.figure(num=1, figsize=fig_size)
plt.plot(vf_td,Eeff.voigt_elasticity*pow(10,-9))
plt.plot(vf_td,Eeff.reuss_elasticity*pow(10,-9))
plt.plot(vf_td,Eeff.hill*pow(10,-9))
plt.plot(vf_td,Eeff.hsub*pow(10,-9),'o-')
plt.plot(vf_td,Eeff.hslb*pow(10,-9),'o-')
plt.plot(vf_td,Eeff.mori_tanaka*pow(10,-9))
plt.plot(vf_td, [terfenolD.E*pow(10,-9) for _ in range(len(vf_td))],'k:')
plt.plot(vf_td, [matrix35016.E*pow(10,-9) for _ in range(len(vf_td))],'k:')
plt.xticks(np.linspace(0,1,11), fontsize=12)
plt.xlim(xmin=0, xmax=1)
plt.yticks(np.linspace(0,ymax_value,int(ymax_value/5+1)), fontsize=12)
plt.ylim(ymin=0, ymax=ymax_value)
plt.xlabel('Volume Fraction of Terfenol-D (f$_{1}$) [X 100%]', fontsize=14)
plt.ylabel('Effective Elastic Modulus (E$_{eff}$) [GPa]', fontsize=14)
```

```
plt.legend(elastic_property_labels, loc='upper left', frameon=True, framealpha=0.9, ncol=2, fontsize
    ↪ =12)
plt.text(0.75,matrix35016.E*pow(10,-9)+0.5,'E$_{3501-6}$ = 4.2 GPa', fontsize=12)
plt.text(0.75,terfenolD.E*pow(10,-9)+0.5,'E$_{Terfenol-D}$ = 30 GPa', fontsize=12)
plt.plot([0.64, 0.64],[0, ymax_value],'k--', alpha=0.5)
plt.text(text_label_x,terfenolD.E*pow(10,-9)+2,'RCP', fontsize=12)
plt.fill_betweenx(y=[ymin_value, ymax_value],x1=[0.64,0.64], x2=[1,1], color='#b3b3b3', alpha=0.3,
    ↪ hatch='/')
plt.tight_layout()
filename1 = 'Plots/EffectiveE.png'
plt.savefig(filename1)


# elastic modulus
plt.figure(num=5, figsize=fig_size)
plt.plot(vf_td,Eeff.voigt_elasticity*pow(10,-9),'#1f77b4')
plt.plot(vf_td,Eeff.voigt*pow(10,-9),'#1f77b4',linestyle='--')
plt.plot(vf_td,Eeff.reuss_elasticity*pow(10,-9),'#ff7f0e')
plt.plot(vf_td,Eeff.reuss*pow(10,-9),'#ff7f0e',linestyle='--')
plt.plot(vf_td, [terfenolD.E*pow(10,-9) for _ in range(len(vf_td))],'k:')
plt.plot(vf_td, [matrix35016.E*pow(10,-9) for _ in range(len(vf_td))],'k:')
plt.xticks(np.linspace(0,1,11), fontsize=12)
plt.xlim(xmin=0, xmax=1)
plt.yticks(np.linspace(0,ymax_value,int(ymax_value/5+1)), fontsize=12)
plt.ylim(ymin=0, ymax=ymax_value)
plt.xlabel('Volume Fraction of Terfenol-D (f$_{1}$) [X 100%]', fontsize=14)
plt.ylabel('Effective Elastic Modulus (E$_{eff}$) [GPa]', fontsize=14)
plt.legend(['Voigt -elasticity','Voigt -model','Reuss -elasticity','Reuss -model'], loc='upper left',
    ↪ frameon=True, framealpha=0.9, ncol=2, fontsize=12)
plt.text(0.75,4.7,'E$_{3501-6}$ = 4.2 GPa', fontsize=12)
plt.text(0.75,30.5,'E$_{Terfenol-D}$ = 30 GPa', fontsize=12)
plt.plot([0.64, 0.64],[0, ymax_value],'k--', alpha=0.5)
plt.text(text_label_x,terfenolD.E*pow(10,-9)+2,'RCP', fontsize=12)
plt.fill_betweenx(y=[ymin_value, ymax_value],x1=[0.64,0.64], x2=[1,1], color='#b3b3b3', alpha=0.3,
    ↪ hatch='/')
plt.tight_layout()
filename5 = 'Plots/EffectiveE_modelverbatim.png'
plt.savefig(filename5)


# shear modulus
ymax_value = 12
plt.figure(num=2, figsize=fig_size)
plt.plot(vf_td,Geff.voigt*pow(10,-9))
plt.plot(vf_td,Geff.reuss*pow(10,-9))
plt.plot(vf_td,Geff.hill*pow(10,-9))
plt.plot(vf_td,Geff.hsub*pow(10,-9),'o-')
plt.plot(vf_td,Geff.hslb*pow(10,-9),'o-')
plt.plot(vf_td,Geff.mori_tanaka*pow(10,-9))
plt.plot(vf_td, [terfenolD.G*pow(10,-9) for _ in range(len(vf_td))],'k:')
plt.plot(vf_td, [matrix35016.G*pow(10,-9) for _ in range(len(vf_td))],'k:')
plt.xticks(np.linspace(0,1,11), fontsize=12)
plt.xlim(xmin=0, xmax=1)
plt.yticks(np.linspace(0,ymax_value,int(ymax_value/2+1)), fontsize=12)
plt.ylim(ymin=0, ymax=ymax_value)
plt.xlabel('Volume Fraction of Terfenol-D (f$_{1}$) [X 100%]', fontsize=14)
plt.ylabel('Effective Shear Modulus (G$_{eff}$) [GPa]', fontsize=14)
plt.legend(elastic_property_labels, loc='upper left', frameon=True, framealpha=0.9, ncol=2, fontsize
    ↪ =12)
plt.text(0.75,matrix35016.G*pow(10,-9)+0.2,'G$_{3501-6}$ = 1.567 GPa', fontsize=12)
plt.text(0.75,terfenolD.G*pow(10,-9)+0.2,'G$_{Terfenol-D}$ = 10.385 GPa', fontsize=12)
plt.plot([0.64, 0.64],[0, ymax_value],'k--', alpha=0.5)
plt.text(text_label_x,terfenolD.G*pow(10,-9)+0.75,'RCP', fontsize=12)
```

```
plt.fill_betweenx(y=[ymin_value, ymax_value],x1=[0.64,0.64], x2=[1,1], color='#b3b3b3', alpha=0.3,
    ↪ hatch='/')
plt.tight_layout()
filename2 = 'Plots/EffectiveG.png'
plt.savefig(filename2)


# bulk modulus
ymax_value = 110
plt.figure(num=3, figsize=fig_size)
plt.plot(vf_td,Keff.voigt*pow(10,-9))
plt.plot(vf_td,Keff.reuss*pow(10,-9))
plt.plot(vf_td,Keff.hill*pow(10,-9))
plt.plot(vf_td,Keff.hsub*pow(10,-9),'o-')
plt.plot(vf_td,Keff.hslb*pow(10,-9),'o-')
plt.plot(vf_td,Keff.mori_tanaka*pow(10,-9))
plt.plot(vf_td, [terfenolD.K*pow(10,-9) for _ in range(len(vf_td))],'k:')
plt.plot(vf_td, [matrix35016.K*pow(10,-9) for _ in range(len(vf_td))],'k:')
plt.xticks(np.linspace(0,1,11), fontsize=12)
plt.xlim(xmin=0, xmax=1)
plt.yticks(np.linspace(0,ymax_value,int(ymax_value/10+1)), fontsize=12)
plt.ylim(ymin=0, ymax=ymax_value)
plt.xlabel('Volume Fraction of Terfenol-D (f$_{1}$) [X 100%]', fontsize=14)
plt.ylabel('Effective Bulk Modulus (K$_{eff}$) [GPa]', fontsize=14)
plt.legend(elastic_property_labels, loc='upper left', frameon=True, framealpha=0.9, ncol=2, fontsize
    ↪ =12)
plt.text(0.75,matrix35016.K*pow(10,-9)+1.4,'K$_{3501-6}$ = 4.375 GPa', fontsize=12)
plt.text(0.75,terfenolD.K*pow(10,-9)+1.4,'K$_{Terfenol-D}$ = 90 GPa', fontsize=12)
plt.plot([0.64, 0.64],[0, ymax_value],'k--', alpha=0.5)
plt.text(text_label_x,terfenolD.K*pow(10,-9)+10,'RCP', fontsize=12)
plt.fill_betweenx(y=[ymin_value, ymax_value],x1=[0.64,0.64], x2=[1,1], color='#b3b3b3', alpha=0.3,
    ↪ hatch='/')
plt.tight_layout()
filename3 = 'Plots/EffectiveK.png'
plt.savefig(filename3)


# poisson's ratio -elasticity
ymin_value = 0.32
ymax_value = 0.48
step = int((ymax_value -ymin_value) / 0.02 +2)
plt.figure(num=4, figsize=fig_size)
plt.plot(vf_td,NUeff.voigt_elasticity,'#1f77b4')
plt.plot(vf_td,NUeff.voigt,'#1f77b4',linestyle='--')
plt.plot(vf_td,NUeff.reuss_elasticity,'#ff7f0e')
plt.plot(vf_td,NUeff.reuss,'#ff7f0e',linestyle='--')
plt.plot(vf_td,NUeff.hill,'#2ca02c')
plt.plot(vf_td,NUeff.hsub,'#d62728',marker='o',linestyle='-')
plt.plot(vf_td,NUeff.hslb,'#9467bd',marker='o',linestyle='-')
plt.plot(vf_td,NUeff.mori_tanaka,'#8c564b')
plt.plot(vf_td, [terfenolD.nu for _ in range(len(vf_td))],'k:')
plt.plot(vf_td, [matrix35016.nu for _ in range(len(vf_td))],'k:')
plt.xticks(np.linspace(0,1,11), fontsize=12)
plt.xlim(xmin=0, xmax=1)
plt.yticks(np.linspace(ymin_value,ymax_value,step), fontsize=12)
plt.ylim(ymin=ymin_value, ymax=ymax_value)
plt.xlabel('Volume Fraction of Terfenol-D (f$_{1}$) [X 100%]', fontsize=14)
plt.ylabel('Effective Poisson''s Ratio (''r'$ \nu_{eff}$'') [-]', fontsize=14)
plt.legend(elastic_property_labels2, loc='upper left', frameon=True, framealpha=0.9, ncol=2, fontsize
    ↪ =12)
plt.text(0.75,matrix35016.nu+0.0025,r'$\nu$''$_{3501-6}$ = 0.34', fontsize=12)
plt.text(0.75,terfenolD.nu+0.0025,r'$\nu$''$_{Terfenol-D}$ = 0.44', fontsize=12)
plt.plot([0.64, 0.64],[0, ymax_value],'k--', alpha=0.5)
plt.text(text_label_x,terfenolD.nu+0.02,'RCP', fontsize=12)
```

```
plt.fill_betweenx(y=[ymin_value, ymax_value],x1=[0.64,0.64], x2=[1,1], color='#b3b3b3', alpha=0.3,
    ↪ hatch='/')
plt.tight_layout()
filename4 = 'Plots/EffectiveNU_comparison.png'
plt.savefig(filename4)


# poisson's ratio
ymin_value = 0.32
ymax_value = 0.46
step = int((ymax_value -ymin_value) / 0.02 + 1)
plt.figure(num=6, figsize=fig_size)
plt.plot(vf_td,NUeff.voigt_elasticity,'#1f77b4')
plt.plot(vf_td,NUeff.voigt,'#1f77b4',linestyle='--')
plt.plot(vf_td,NUeff.reuss_elasticity,'#ff7f0e')
plt.plot(vf_td,NUeff.reuss,'#ff7f0e',linestyle='--')
plt.plot(vf_td, [terfenolD.nu for _ in range(len(vf_td))],'k:')
plt.plot(vf_td, [matrix35016.nu for _ in range(len(vf_td))],'k:')
plt.xticks(np.linspace(0,1,11), fontsize=12)
plt.xlim(xmin=0, xmax=1)
plt.yticks(np.linspace(ymin_value,ymax_value,step), fontsize=12)
plt.ylim(ymin=ymin_value, ymax=ymax_value)
plt.xlabel('Volume Fraction of Terfenol-D (f$_{1}$) [X 100%]', fontsize=14)
plt.ylabel('Effective Poisson''s Ratio ('r'$ \nu_{eff}$'') [-]', fontsize=14)
plt.legend(['Voigt -elasticity','Voigt -verbatim','Reuss -elasticity','Reuss -verbatim'], loc='upper
    ↪ left', frameon=True, framealpha=0.9, ncol=2, fontsize=12)
plt.text(0.75,matrix35016.nu+0.0025,r'$\nu$''$_{3501-6}$ = 0.34', fontsize=12)
plt.text(0.75,terfenolD.nu+0.0025,r'$\nu$''$_{Terfenol-D}$ = 0.44', fontsize=12)
plt.plot([0.64, 0.64],[0, ymax_value],'k--', alpha=0.5)
plt.text(text_label_x,terfenolD.nu+0.008,'RCP', fontsize=12)
plt.fill_betweenx(y=[ymin_value, ymax_value],x1=[0.64,0.64], x2=[1,1], color='#b3b3b3', alpha=0.3,
    ↪ hatch='/')
plt.tight_layout()
filename6 = 'Plots/EffectiveNU_modelverbatim.png'
plt.savefig(filename6)
```

D.2.3 Effective Coefficient of Thermal Expansion

```
import os
import shutil
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from bulkmaterialprop import matrix35016
from bulkmaterialprop import terfenolD
import elastic_property_approx as elastic_properties

class alpha:
    voigt = []

class q1_cribb:
    voigt = []

class q2_cribb:
    voigt = []

##
# set volume fraction
vf_td = np.linspace(0,1,101)
```

```
vf_matrix = 1 -vf_td
K = pd.read_csv('K_approximations.csv')
K_voigt = K['Voigt']
K_reuss = K['Reuss']
K_hill = K['Hill']
K_hsupper = K['Hashin-Shtrikman (Upper Bound)']
K_hslower = K['Hashin-Shtrikman (Lower Bound)']
K_mt = K['Mori-Tanaka']



## approximations

# VOIGT
alpha.voigt = elastic_properties.voigt(terfenolD.CTE,matrix35016.CTE,vf_td,vf_matrix)

# REUSS
alpha.reuss = elastic_properties.reuss(terfenolD.CTE,matrix35016.CTE,vf_td,vf_matrix)
alpha.reuss2=(vf_td*terfenolD.E*terfenolD.CTE+vf_matrix*matrix35016.E*matrix35016.CTE)/(vf_td*
    ↪ terfenolD.E+vf_matrix*matrix35016.E)

# BLACKBURN
alpha.blackburn = terfenolD.CTE + (((3/2) * (1 -terfenolD.nu) * vf_matrix * (matrix35016.CTE -
    ↪ terfenolD.CTE)) / ((1 / 2) * (1 + terfenolD.nu) + vf_matrix * (1 -2 * terfenolD.nu) + (1 -2 *
    ↪ matrix35016.nu) * (terfenolD.E / matrix35016.E) * vf_td))

# TURNER
alpha.turner = (vf_matrix * matrix35016.CTE * matrix35016.K + vf_td * terfenolD.CTE * terfenolD.K) / (
    ↪ vf_matrix * matrix35016.K + vf_td * terfenolD.K)

# CRIBB
q1_cribb.voigt = (matrix35016.K * (K_voigt -terfenolD.K)) / (K_voigt * (matrix35016.K -terfenolD.K))
q1_cribb.reuss = (matrix35016.K * (K_reuss -terfenolD.K)) / (K_reuss * (matrix35016.K -terfenolD.K))
q1_cribb.hill = (matrix35016.K * (K_hill -terfenolD.K)) / (K_hill * (matrix35016.K -terfenolD.K))
q1_cribb.mt = (matrix35016.K * (K_mt -terfenolD.K)) / (K_mt * (matrix35016.K -terfenolD.K))
q1_cribb.hsub = (matrix35016.K * (K_hsupper -terfenolD.K)) / (K_hsupper * (matrix35016.K -terfenolD.K)
    ↪ )
q1_cribb.hslb = (matrix35016.K * (K_hslower -terfenolD.K)) / (K_hslower * (matrix35016.K -terfenolD.K)
    ↪ )

q2_cribb.voigt = (terfenolD.K * (matrix35016.K -K_voigt)) / (K_voigt * (matrix35016.K -terfenolD.K))
q2_cribb.reuss = (terfenolD.K * (matrix35016.K -K_reuss)) / (K_reuss * (matrix35016.K -terfenolD.K))
q2_cribb.hill = (terfenolD.K * (matrix35016.K -K_hill)) / (K_hill * (matrix35016.K -terfenolD.K))
q2_cribb.mt = (terfenolD.K * (matrix35016.K -K_mt)) / (K_mt * (matrix35016.K -terfenolD.K))
q2_cribb.hsub = (terfenolD.K * (matrix35016.K -K_hsupper)) / (K_hsupper * (matrix35016.K -terfenolD.K)
    ↪ )
q2_cribb.hslb = (terfenolD.K * (matrix35016.K -K_hslower)) / (K_hslower * (matrix35016.K -terfenolD.K)
    ↪ )

alpha.cribb_voigt = q1_cribb.voigt * matrix35016.CTE + q2_cribb.voigt * terfenolD.CTE
alpha.cribb_reuss = q1_cribb.reuss * matrix35016.CTE + q2_cribb.reuss * terfenolD.CTE
alpha.cribb_hill = q1_cribb.hill *matrix35016.CTE + q2_cribb.hill * terfenolD.CTE
alpha.cribb_mt = q1_cribb.mt *matrix35016.CTE + q2_cribb.mt * terfenolD.CTE
alpha.cribb_hsub = q1_cribb.hsub * matrix35016.CTE + q2_cribb.hsub * terfenolD.CTE
alpha.cribb_hslb = q1_cribb.hslb * matrix35016.CTE + q2_cribb.hslb * terfenolD.CTE

# KERNER
q_kerner = ((1 / matrix35016.K) -(1 / terfenolD.K)) / ((vf_td / terfenolD.K) + (vf_matrix /
    ↪ matrix35016.K) + (3 / (4 * matrix35016.G)))
alpha.kerner = terfenolD.CTE * vf_td + matrix35016.CTE * vf_matrix + vf_td * vf_matrix * (matrix35016.
    ↪ CTE -terfenolD.CTE) * q_kerner

# WANG-KWEI
```

```
q_wk = (3 * (terfenolD.E / matrix35016.E) * (1 -matrix35016.nu)) / ((terfenolD.E / matrix35016.E) * (2
    ↪  * vf_td * (1 -2 * matrix35016.nu) + (1 + matrix35016.nu)) + 2 * vf_matrix * (1 -2 * terfenolD
    ↪ .nu))
alpha.wang_kwei = matrix35016.CTE -vf_td * q_wk * (matrix35016.CTE -terfenolD.CTE)
alpha.wang_kwei2 = matrix35016.CTE * (1 -vf_td * (1 -(terfenolD.CTE / matrix35016.CTE)) * q_wk)

#THOMAS
a=1
alpha.thomas = pow(pow(matrix35016.CTE,a) * vf_matrix + pow(terfenolD.CTE,a) * vf_td, (1 / a))

# TUMMALA-FRIEDBERG
q_tf = ((1 + matrix35016.nu) / (2 * matrix35016.E)) / ((1 + matrix35016.nu) / (2 * matrix35016.E) + (1
    ↪  -2 * terfenolD.nu) / (terfenolD.E))
alpha.tummala_friedberg = matrix35016.CTE -vf_td * q_tf * (matrix35016.CTE -terfenolD.CTE)

# FAHMI-RAGAI
alpha.fahmi_ragai = matrix35016.CTE -((3 * vf_td * (matrix35016.CTE -terfenolD.CTE) * (1 -matrix35016.
    ↪ nu)) / (2 * (1 -2 * terfenolD.nu) * (-1 * vf_td) * (matrix35016.E / terfenolD.E) + 2 * vf_td *
    ↪  (1 -2 * matrix35016.nu) + (1 + matrix35016.nu)))


#####
cte_approximations1 = ['Voigt','Reuss','Blackburn','Turner','Kerner','Wang-Kwei','Thomas (a=1)','
    ↪ Tummala-Friedberg','Fahmi-Ragai']
cte_approximations2 = ['Cribb (Voigt)','Cribb (Reuss)','Cribb (Hill)','Cribb (Mori-Tanaka)','Cribb (HS
    ↪  -Upper)','Cribb (HS -Lower)']
cte_approximations_total = cte_approximations1 + cte_approximations2

AR = 8/5
width = 12
fig_size = [width, width/AR]
legend_location = 'upper right'

ymax_value = 4.8e-5
ymin_value = 1e-5
step = 9

plt.figure(num=1, figsize=fig_size)
plt.plot(vf_td,alpha.voigt,'v-')
plt.plot(vf_td,alpha.reuss2,'p-')
plt.plot(vf_td,alpha.blackburn,'>-')
plt.plot(vf_td,alpha.turner,'<-')
plt.plot(vf_td,alpha.kerner,'1-')
plt.plot(vf_td,alpha.wang_kwei,'h-')
# plt.plot(vf_td,alpha.wang_kwei2)
plt.plot(vf_td,alpha.thomas,'2-')
plt.plot(vf_td,alpha.tummala_friedberg,'^-')
plt.plot(vf_td,alpha.fahmi_ragai,'+-')
#
plt.plot(vf_td, [terfenolD.CTE for _ in range(len(vf_td))],'k:')
plt.plot(vf_td, [matrix35016.CTE for _ in range(len(vf_td))],'k:')
#
plt.xticks(np.linspace(0,1,11), fontsize=12)
plt.xlim(xmin=0, xmax=1)
plt.yticks(np.linspace(ymin_value,ymax_value,step), fontsize=12)
plt.ylim(ymin=ymin_value, ymax=ymax_value)
plt.xlabel('Volume Fraction of Terfenol-D (f$_{1}$) [X 100%]', fontsize=14)
plt.ylabel('Effective Coefficient of Thermal Expansion ('r'$\alpha_{eff}$'') ['r'$^{\circ}$''C$^{-1}$]
    ↪ ', fontsize=14)
plt.legend(cte_approximations1, loc=legend_location, frameon=True, framealpha=0.9, fontsize=12)
plt.text(0.3,matrix35016.CTE-1.2e-6,r'$\alpha_{3501-6}$''= 4.5e-5 'r'$^{\circ}$''C$^{-1}$', fontsize
    ↪ =12)
plt.text(0.045,terfenolD.CTE+0.3e-6,r'$\alpha_{Terfenol-D}$''= 1.2e-5 'r'$^{\circ}$''C$^{-1}$',
    ↪ fontsize=12)
```

```
plt.plot([0.64, 0.64],[ymin_value, ymax_value],'k--', alpha=0.5)
plt.text(0.602,matrix35016.CTE-0.75e-5,'RCP', fontsize=12)
plt.fill_betweenx(y=[ymin_value, ymax_value],x1=[0.64,0.64], x2=[1,1], color='#b3b3b3', alpha=0.3,
    ↪ hatch='/')
plt.tight_layout()
filename1 = 'Plots/EffectiveCTE_elasticapprox.png'
plt.savefig(filename1)


plt.figure(num=2, figsize=fig_size)
plt.plot(vf_td,alpha.cribb_voigt,'o-')
plt.plot(vf_td,alpha.cribb_reuss,'v-')
plt.plot(vf_td,alpha.cribb_hill,'s-')
plt.plot(vf_td,alpha.cribb_mt,'|-')
plt.plot(vf_td,alpha.cribb_hsub,'*-')
plt.plot(vf_td,alpha.cribb_hslb,'D-')
#
plt.plot(vf_td, [terfenolD.CTE for _ in range(len(vf_td))],'k:')
plt.plot(vf_td, [matrix35016.CTE for _ in range(len(vf_td))],'k:')
#
plt.xticks(np.linspace(0,1,11), fontsize=12)
plt.xlim(xmin=0, xmax=1)
plt.yticks(np.linspace(ymin_value,ymax_value,step), fontsize=12)
plt.ylim(ymin=ymin_value, ymax=ymax_value)
plt.xlabel('Volume Fraction of Terfenol-D (f$_{1}$) [X 100%]', fontsize=14)
plt.ylabel('Effective Coefficient of Thermal Expansion ('r'$\alpha_{eff}$'') ['r'$^{\circ}$''C$^{-1}$]
    ↪ ', fontsize=14)
plt.legend(cte_approximations2, loc=legend_location, frameon=True, framealpha=0.9, fontsize=12)
plt.text(0.3,matrix35016.CTE-1.2e-6,r'$\alpha_{3501-6}$''= 4.5e-5 'r'$^{\circ}$''C$^{-1}$', fontsize
    ↪ =12)
plt.text(0.045,terfenolD.CTE+0.3e-6,r'$\alpha_{Terfenol-D}$''= 1.2e-5 'r'$^{\circ}$''C$^{-1}$',
    ↪ fontsize=12)
plt.plot([0.64, 0.64],[ymin_value, ymax_value],'k--', alpha=0.5)
plt.text(0.602,matrix35016.CTE-0.75e-5,'RCP', fontsize=12)
plt.fill_betweenx(y=[ymin_value, ymax_value],x1=[0.64,0.64], x2=[1,1], color='#b3b3b3', alpha=0.3,
    ↪ hatch='/')
plt.tight_layout()
filename2 = 'Plots/EffectiveCTE_cribbapprox.png'
plt.savefig(filename2)


plt.figure(num=3, figsize=fig_size)
plt.plot(vf_td,alpha.voigt,'v-')
plt.plot(vf_td,alpha.reuss2,'p-')
plt.plot(vf_td,alpha.blackburn,'>-')
plt.plot(vf_td,alpha.turner,'<-')
plt.plot(vf_td,alpha.kerner,'1-')
plt.plot(vf_td,alpha.wang_kwei,'h-')
plt.plot(vf_td,alpha.thomas,'2-')
plt.plot(vf_td,alpha.tummala_friedberg,'^-')
plt.plot(vf_td,alpha.fahmi_ragai,'+-')
#
plt.plot(vf_td,alpha.cribb_voigt,'o-')
plt.plot(vf_td,alpha.cribb_reuss,'v-')
plt.plot(vf_td,alpha.cribb_hill,'s-')
plt.plot(vf_td,alpha.cribb_mt,'|-')
plt.plot(vf_td,alpha.cribb_hsub,'*-')
plt.plot(vf_td,alpha.cribb_hslb,'D-')
#
plt.plot(vf_td, [terfenolD.CTE for _ in range(len(vf_td))],'k:')
plt.plot(vf_td, [matrix35016.CTE for _ in range(len(vf_td))],'k:')
#
plt.xticks(np.linspace(0,1,11), fontsize=12)
plt.xlim(xmin=0, xmax=1)
```

```
plt.yticks(np.linspace(ymin_value,ymax_value,step), fontsize=12)
plt.ylim(ymin=ymin_value, ymax=ymax_value)
plt.xlabel('Volume Fraction of Terfenol-D (f$_{1}$) [X 100%]', fontsize=14)
plt.ylabel('Effective Coefficient of Thermal Expansion ('r'$\alpha_{eff}$'') ['r'$^{\circ}$''C$^{-1}$]
    ↪ ', fontsize=14)
plt.legend(cte_approximations_total, loc=legend_location, frameon=True, framealpha=0.9, fontsize=12)
plt.text(0.3,matrix35016.CTE-1.2e-6,r'$\alpha_{3501-6}$''= 4.5e-5 'r'$^{\circ}$''C$^{-1}$', fontsize
    ↪ =12)
plt.text(0.045,terfenolD.CTE+0.3e-6,r'$\alpha_{Terfenol-D}$''= 1.2e-5 'r'$^{\circ}$''C$^{-1}$',
    ↪ fontsize=12)
plt.plot([0.64, 0.64],[ymin_value, ymax_value],'k--', alpha=0.5)
plt.text(0.602,matrix35016.CTE-0.75e-5,'RCP', fontsize=12)
plt.fill_betweenx(y=[ymin_value, ymax_value],x1=[0.64,0.64], x2=[1,1], color='#b3b3b3', alpha=0.3,
    ↪ hatch='/')
plt.tight_layout()
filename3 = 'Plots/EffectiveCTE_allapprox.png'
plt.savefig(filename3)
```

D.3 Classical Lamination Theory Script

The classical lamination theory script was developed from a script formulation previously presented in [171].

D.3.1 Classical Lamination Theory

The classical lamination theory was scripted as a function of the volume fraction of the inclusion, Terfenol-D.

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import animation
import CLT_matprops as CLT_matprops
import clt_functions as clt
from clt_functions_vf import DataExtraction, clt_form_vf


coord_sys = 'MCS'
## coord_sys = 'LCS'

var = 'stress'
# var = 'strain'

axis = 0
# axis = 1
# axis = 2

parameter_string = coord_sys + '_' + var + '_' + str(axis)

if var == 'strain':
    if axis == 0:
        ylabel_string = 'Longitudinal Strain' ' ('r'$\epsilon_{1}$'')'
        ymin_value = -0.006
        ymax_value = 0.002
        factor = 0.8
        RCP_text_y = factor * ymax_value
    elif axis == 1:
        ylabel_string = 'Transverse Strain' ' ('r'$\epsilon_{2}$'')'
```

```
        ymin_value = -0.006
        ymax_value = 0
        RCP_text_y = ymax_value -0.0005
    else:
        ylabel_string = 'In-plane Shear Strain' ' ('r'$\gamma_{12}$'')'
        ymin_value = -0.004
        ymax_value = 0.004
        factor = 0.8
        RCP_text_y = factor * ymax_value
else:
    if axis == 0:
        ylabel_string = 'Longitudinal Stress' ' ('r'$\sigma_{1}$'')' ' [MPa]'
        ymin_value = -120
        ymax_value = 40
        factor = 0.8
        RCP_text_y = factor * ymax_value
    elif axis == 1:
        ylabel_string = 'Transverse Stress' ' ('r'$\sigma_{2}$'')' ' [MPa]'
        ymin_value = -120
        ymax_value = 10
        RCP_text_y = 0
    else:
        ylabel_string = 'In-plane Shear Stress' ' ('r'$\tau_{12}$'')' ' [MPa]'
        ymin_value = -40
        ymax_value = 40
        RCP_text_y = ymax_value -5

legend_string = ['HexPly AS4/3501-6','Hexcel 3501-6','Terfenol-D (bulk)','Voigt','Reuss', 'Hashin-
    ↪ Shtrikman (Upper)', 'Hashin-Shtrikman (Lower)', 'Mori-Tanaka']

local_data0 = np.empty((0,2), float)
local_data1 = np.empty((0,2), float)
local_data2 = np.empty((0,2), float)
local_data4 = np.empty((0,2), float)
local_data5 = np.empty((0,2), float)
local_data6 = np.empty((0,2), float)
local_data7 = np.empty((0,2), float)
local_data8 = np.empty((0,2), float)


vf_td = np.around(np.linspace(0,1,101),2)
for vf in vf_td:
    mat_id_num = 0
    vf_datapoint = clt_form_vf(vf,mat_id_num,coord_sys,var,axis)
    local_data0 = np.append(local_data0, [vf_datapoint], axis=0)

    mat_id_num = 1
    vf_datapoint = clt_form_vf(vf,mat_id_num,coord_sys,var,axis)
    local_data1 = np.append(local_data1, [vf_datapoint], axis=0)

    mat_id_num = 2
    vf_datapoint = clt_form_vf(vf,mat_id_num,coord_sys,var,axis)
    local_data2 = np.append(local_data2, [vf_datapoint], axis=0)

    mat_id_num = 4
    vf_datapoint = clt_form_vf(vf,mat_id_num,coord_sys,var,axis)
    local_data4 = np.append(local_data4, [vf_datapoint], axis=0)

    mat_id_num = 5
    vf_datapoint = clt_form_vf(vf,mat_id_num,coord_sys,var,axis)
    local_data5 = np.append(local_data5, [vf_datapoint], axis=0)

    mat_id_num = 6
    vf_datapoint = clt_form_vf(vf,mat_id_num,coord_sys,var,axis)
```

```
    local_data6 = np.append(local_data6, [vf_datapoint], axis=0)

    mat_id_num = 7
    vf_datapoint = clt_form_vf(vf,mat_id_num,coord_sys,var,axis)
    local_data7 = np.append(local_data7, [vf_datapoint], axis=0)

    mat_id_num = 8
    vf_datapoint = clt_form_vf(vf,mat_id_num,coord_sys,var,axis)
    local_data8 = np.append(local_data8, [vf_datapoint], axis=0)



AR = 8/5
width = 12
fig_size = [width, width/AR]
plt.figure(num=1, figsize=fig_size)

if var == 'stress':
    plt.plot(local_data0[:,0],local_data0[:,1]*pow(10,-6))
    plt.plot(local_data2[:,0],local_data2[:,1]*pow(10,-6))
    plt.plot(local_data1[:,0],local_data1[:,1]*pow(10,-6))
    plt.plot(local_data4[:,0],local_data4[:,1]*pow(10,-6))
    plt.plot(local_data5[:,0],local_data5[:,1]*pow(10,-6))
    plt.plot(local_data6[:,0],local_data6[:,1]*pow(10,-6))
    plt.plot(local_data7[:,0],local_data7[:,1]*pow(10,-6))
    plt.plot(local_data8[:,0],local_data8[:,1]*pow(10,-6))
else:
    plt.plot(local_data0[:,0],local_data0[:,1])
    plt.plot(local_data2[:,0],local_data2[:,1])
    plt.plot(local_data1[:,0],local_data1[:,1])
    plt.plot(local_data4[:,0],local_data4[:,1])
    plt.plot(local_data5[:,0],local_data5[:,1])
    plt.plot(local_data6[:,0],local_data6[:,1])
    plt.plot(local_data7[:,0],local_data7[:,1])
    plt.plot(local_data8[:,0],local_data8[:,1])

plt.xticks(np.linspace(0,1,11), fontsize=12)
plt.xlim(xmin=0, xmax=1)
plt.ylim(ymin=ymin_value, ymax=ymax_value)
plt.xlabel('Volume Fraction of Terfenol-D (f$_{1}$) [X 100%]', fontsize=14)
plt.ylabel(ylabel_string, fontsize=14)
plt.legend(legend_string, loc='upper left', frameon=True, framealpha=0.9, ncol=2, fontsize=12)
plt.plot([0.64, 0.64],[ymin_value, ymax_value],'k--', alpha=0.5)
plt.text(0.65,RCP_text_y,'RCP', fontsize=12)
plt.fill_betweenx(y=[ymin_value, ymax_value],x1=[0.64,0.64], x2=[1,1], color='#b3b3b3', alpha=0.3,
    ↪ hatch='/')
plt.tight_layout()
filename = 'Plots/CLT_vf/' + parameter_string + '.png'
plt.savefig(filename)
```

D.3.2 Material Property Values

The effective property approximations were used to approximate the elastic property and coefficient of thermal expansion values.

```
import numpy as np

from bulkmaterialprop import matrix35016
from bulkmaterialprop import terfenolD
from bulkmaterialprop import as4_35016
```

```
import elastic_property_approx as elastic_properties

prepreg = {
    'E1' : as4_35016.E1,
    'E2' : as4_35016.E2,
    'n12' : as4_35016.nu12,
    'G12' : as4_35016.G12,
    'a1' : as4_35016.CTE1,
    'a2' : as4_35016.CTE2,
    'thickness' : as4_35016.thickness,
}

td_bulk = {
    'E1' : terfenolD.E,
    'E2' : terfenolD.E,
    'n12' : terfenolD.nu,
    'G12' : terfenolD.G,
    'a1' : terfenolD.CTE,
    'a2' : terfenolD.CTE,
    'thickness' : terfenolD.thickness,
}

epoxy = {
    'E1' : matrix35016.E,
    'E2' : matrix35016.E,
    'n12' : matrix35016.nu,
    'G12' : matrix35016.G,
    'a1' : matrix35016.CTE,
    'a2' : matrix35016.CTE,
    'thickness' : matrix35016.thickness,
}

delam = {
    'E1' : 100,
    'E2' : 100,
    'n12' : 0,
    'G12' : 0,
    'a1' : 0,
    'a2' : 0,
    'thickness' : 0.01e-3,
}

##################################################

def alphaCTE(vf):
    vf_td = vf
    vf_matrix = 1 -vf

    K_voigt = elastic_properties.voigt(terfenolD.K,matrix35016.K,vf,1-vf)
    K_reuss = elastic_properties.reuss(terfenolD.K,matrix35016.K,vf,1-vf)
    K_hill = (K_voigt + K_reuss) / 2
    E, NU, G, K_mt = elastic_properties.mori_tanaka(terfenolD.K,terfenolD.G,matrix35016.K,matrix35016.
        ↪ G,vf_td,vf_matrix)
    E, NU, G, K_hsupper = elastic_properties.hashin_shtrikman_upper(terfenolD.K,terfenolD.G,
        ↪ matrix35016.K,matrix35016.G,vf_td,vf_matrix)
    E, NU, G, K_hslower = elastic_properties.hashin_shtrikman_lower(terfenolD.K,terfenolD.G,
        ↪ matrix35016.K,matrix35016.G,vf_td,vf_matrix)

    selection = 2
    if selection == 0:
        # voigt
        alpha = elastic_properties.voigt(terfenolD.CTE,matrix35016.CTE,vf_td,vf_matrix)

    elif selection == 1:
```

```
    # reuss
    alpha = (vf_td*terfenolD.E*terfenolD.CTE+vf_matrix*matrix35016.E*matrix35016.CTE)/(vf_td*
        ↪ terfenolD.E+vf_matrix*matrix35016.E)

elif selection == 2:
    # blackburn
    alpha = terfenolD.CTE + (((3/2) * (1 -terfenolD.nu) * vf_matrix * (matrix35016.CTE -terfenolD.
        ↪ CTE)) / ((1 / 2) * (1 + terfenolD.nu) + vf_matrix * (1 -2 * terfenolD.nu) + (1 -2 *
        ↪ matrix35016.nu) * (terfenolD.E / matrix35016.E) * vf_td))

elif selection == 3:
    # turner
    alpha = (vf_matrix * matrix35016.CTE * matrix35016.K + vf_td * terfenolD.CTE * terfenolD.K) /
        ↪ (vf_matrix * matrix35016.K + vf_td * terfenolD.K)

elif selection == 4:
    # kerner
    q_kerner = ((1 / matrix35016.K) -(1 / terfenolD.K)) / ((vf_td / terfenolD.K) + (vf_matrix /
        ↪ matrix35016.K) + (3 / (4 * matrix35016.G)))
    alpha = terfenolD.CTE * vf_td + matrix35016.CTE * vf_matrix + vf_td * vf_matrix * (matrix35016
        ↪ .CTE -terfenolD.CTE) * q_kerner

elif selection == 5:
    # wang-kwei
    q_wk = (3 * (terfenolD.E / matrix35016.E) * (1 -matrix35016.nu)) / ((terfenolD.E / matrix35016
        ↪ .E) * (2 * vf_td * (1 -2 * matrix35016.nu) + (1 + matrix35016.nu)) + 2 * vf_matrix *
        ↪ (1 -2 * terfenolD.nu))
    alpha = matrix35016.CTE -vf_td * q_wk * (matrix35016.CTE -terfenolD.CTE)

elif selection == 6:
    # thomas
    a=1
    alpha = pow(pow(matrix35016.CTE,a) * vf_matrix + pow(terfenolD.CTE,a) * vf_td, (1 / a))

elif selection == 7:
    # timmala-friedberg
    q_tf = ((1 + matrix35016.nu) / (2 * matrix35016.E)) / ((1 + matrix35016.nu) / (2 * matrix35016
        ↪ .E) + (1 -2 * terfenolD.nu) / (terfenolD.E))
    alpha = matrix35016.CTE -vf_td * q_tf * (matrix35016.CTE -terfenolD.CTE)

elif selection == 8:
    # fahmi-ragai
    alpha = matrix35016.CTE -((3 * vf_td * (matrix35016.CTE -terfenolD.CTE) * (1 -matrix35016.nu))
        ↪  / (2 * (1 -2 * terfenolD.nu) * (-1 * vf_td) * (matrix35016.E / terfenolD.E) + 2 *
        ↪ vf_td * (1 -2 * matrix35016.nu) + (1 + matrix35016.nu)))

elif selection == 9:
    # cribb-voigt
    q1_cribb = (matrix35016.K * (K_voigt -terfenolD.K)) / (K_voigt * (matrix35016.K -terfenolD.K))
    q2_cribb = (terfenolD.K * (matrix35016.K -K_voigt)) / (K_voigt * (matrix35016.K -terfenolD.K))
    alpha = q1_cribb * matrix35016.CTE + q2_cribb * terfenolD.CTE

elif selection == 10:
    # cribb-reuss
    q1_cribb = (matrix35016.K * (K_reuss -terfenolD.K)) / (K_reuss * (matrix35016.K -terfenolD.K))
    q2_cribb = (terfenolD.K * (matrix35016.K -K_reuss)) / (K_reuss * (matrix35016.K -terfenolD.K))
    alpha = q1_cribb * matrix35016.CTE + q2_cribb * terfenolD.CTE

elif selection == 11:
    # cribb-hill
    q1_cribb = (matrix35016.K * (K_hill -terfenolD.K)) / (K_hill * (matrix35016.K -terfenolD.K))
    q2_cribb = (terfenolD.K * (matrix35016.K -K_hill)) / (K_hill * (matrix35016.K -terfenolD.K))
    alpha = q1_cribb *matrix35016.CTE + q2_cribb * terfenolD.CTE
```

```
    elif selection == 12:
        # cribb-mt
        q1_cribb = (matrix35016.K * (K_mt -terfenolD.K)) / (K_mt * (matrix35016.K -terfenolD.K))
        q2_cribb = (terfenolD.K * (matrix35016.K -K_mt)) / (K_mt * (matrix35016.K -terfenolD.K))
        alpha = q1_cribb *matrix35016.CTE + q2_cribb * terfenolD.CTE

    elif selection == 13:
        # cribb-hsub
        q1_cribb = (matrix35016.K * (K_hsupper -terfenolD.K)) / (K_hsupper * (matrix35016.K -terfenolD
            ↪ .K))
        q2_cribb = (terfenolD.K * (matrix35016.K -K_hsupper)) / (K_hsupper * (matrix35016.K -terfenolD
            ↪ .K))
        alpha = q1_cribb * matrix35016.CTE + q2_cribb * terfenolD.CTE

    elif selection == 14:
        # cribb-hslb
        q1_cribb = (matrix35016.K * (K_hslower -terfenolD.K)) / (K_hslower * (matrix35016.K -terfenolD
            ↪ .K))
        q2_cribb = (terfenolD.K * (matrix35016.K -K_hslower)) / (K_hslower * (matrix35016.K -terfenolD
            ↪ .K))
        alpha = q1_cribb * matrix35016.CTE + q2_cribb * terfenolD.CTE

    return alpha

##################################################

def td_voigt_pc(vf):
    vf_td = vf
    vf_matrix = 1 -vf

    a = alphaCTE(vf)

    G = elastic_properties.voigt(terfenolD.G,matrix35016.G,vf_td,vf_matrix)
    K = elastic_properties.voigt(terfenolD.K,matrix35016.K,vf_td,vf_matrix)

    td_voigt = {
        'E1' : elastic_properties.voigt(terfenolD.E,matrix35016.E,vf_td,vf_matrix),
        'E2' : elastic_properties.voigt(terfenolD.E,matrix35016.E,vf_td,vf_matrix),
        'n12' : (3 * K -2 * G) / (2 * (3 * K + G)),
        'G12' : G,
        'a1' : a,
        'a2' : a,
        'thickness' : terfenolD.thickness,
    }

    return td_voigt

def td_reuss_pc(vf):
    vf_td = vf
    vf_matrix = 1 -vf

    a = alphaCTE(vf)

    G = elastic_properties.reuss(terfenolD.G,matrix35016.G,vf_td,vf_matrix)
    K = elastic_properties.reuss(terfenolD.K,matrix35016.K,vf_td,vf_matrix)

    td_reuss = {
        'E1' : (9 * K * G) / (3 * K + G),
        'E2' : (9 * K * G) / (3 * K + G),
        'n12' : (3 * K -2 * G) / (2 * (3 * K + G)),
        'G12' : G,
        'a1' : a,
        'a2' : a,
        'thickness' : terfenolD.thickness,
```

```
    }

    return td_reuss

def td_hsub_pc(vf):
    vf_td = vf
    vf_matrix = 1 -vf

    a = alphaCTE(vf)

    E, NU, G, K = elastic_properties.hashin_shtrikman_upper(terfenolD.K,terfenolD.G,matrix35016.K,
        ↪ matrix35016.G,vf_td,vf_matrix)

    td_hsub = {
        'E1' : E,
        'E2' : E,
        'n12' : NU,
        'G12' : G,
        'a1' : a,
        'a2' : a,
        'thickness' : terfenolD.thickness,
    }

    return td_hsub

def td_hslb_pc(vf):
    vf_td = vf
    vf_matrix = 1 -vf

    a = alphaCTE(vf)

    E, NU, G, K = elastic_properties.hashin_shtrikman_lower(terfenolD.K,terfenolD.G,matrix35016.K,
        ↪ matrix35016.G,vf_td,vf_matrix)

    td_hslb = {
        'E1' : E,
        'E2' : E,
        'n12' : NU,
        'G12' : G,
        'a1' : a,
        'a2' : a,
        'thickness' : terfenolD.thickness,
    }

    return td_hslb

def td_mt_pc(vf):
    vf_td = vf
    vf_matrix = 1 -vf

    a = alphaCTE(vf)

    E, NU, G, K = elastic_properties.mori_tanaka(terfenolD.K,terfenolD.G,matrix35016.K,matrix35016.G,
        ↪ vf_td,vf_matrix)

    td_mt = {
        'E1' : E,
        'E2' : E,
        'n12' : NU,
        'G12' : G,
        'a1' : a,
        'a2' : a,
        'thickness' : terfenolD.thickness,
    }
```

```
    return td_mt
```

D.3.3 General CLT Functions

The functions defined in this script were modified from those presented in [171]

```
import numpy

class LaminateLayupError(TypeError): pass

def calc_stressCLT(mat_list, lam, F, fail_list = None, dT = 0, dM = 0):
    """ Calculates stress and strain vectors according to CLT. """

    # Get number of layers
    num = len(lam["ang"])
    # total_thk = numpy.sum(lam["thk"])

    # Z vector contains z coordinates.
    Z = assemble_Z(lam)

    # Calculates stiffness matrix based on laminate properties.
    ABD = assemble_ABD(mat_list, lam, Z, fail_list)

    # Calculates thermal, moisture resultants and adds to Forces vector
    Nt = calc_thermal_forces(mat_list, lam, Z, fail_list, dT)
    F = F + Nt
    # Nm = calc_moisture_forces(mat_list, lam, Z, fail_list, dM)
    # F = F + Nt + Nm

    # Calculates strain vector by solving eq. form AX = B
    strain_vector = numpy.linalg.solve(ABD, F)

    # Initializes strain and curvature vectors & defines values
    strains = curvatures = numpy.zeros((1, 3))
    strains = strain_vector[:3]
    curvatures = strain_vector [3:6]

    # Initializes Laminate System (LS) strain vectors (inferior and superior)
    LS_strain_inf = numpy.zeros((3, num))
    LS_strain_sup = numpy.zeros((3, num))

    # Assign Laminate System strain (Epsilon = Epsilon0 + kappa*z)
    for i in range(num):
        LS_strain_inf[:3,i] = strains + curvatures*Z[i]
        LS_strain_sup[:3,i] = strains + curvatures*Z[i+1]

    # Initialize Material System stress & strain vectors (inferior and superior)
    MS_strain_inf = numpy.zeros((3, num))
    MS_strain_sup = numpy.zeros((3, num))

    MS_stress_inf = numpy.zeros((3, num))
    MS_stress_sup = numpy.zeros((3, num))

    LS_stress_inf = numpy.zeros((3, num))
    LS_stress_sup = numpy.zeros((3, num))

    # Calculates Material System stresses and strains
    Z = numpy.flip(Z)
    for i in range(num):
        mat_id = lam["mat_id"][i]
```

```
        mat_prop = mat_list[mat_id]

        T = assemble_matrixT(lam["ang"][i])

        R = numpy.array([[1,0,0],[0,1,0],[0,0,2]])
        Rinv = numpy.linalg.inv(R)
        RTR = numpy.dot(numpy.dot(R,T), Rinv)

        Ak = numpy.dot(RTR,strains)
        Bk = numpy.dot(RTR, curvatures)

        MS_strain_sup[:,i] = Ak + Bk * Z[i+1]
        MS_strain_inf[:,i] = Ak + Bk * Z[i]


        Qbar = assemble_matrixQbar(mat_prop,lam["ang"][i])

        A1k = numpy.dot(Qbar,strains)
        B1k = numpy.dot(Qbar,curvatures)
        MS_stress_sup[:,i] = A1k + B1k * Z[i+1]
        MS_stress_inf[:,i] = A1k + B1k * Z[i]

        A2k = numpy.dot(T,A1k)
        B2k = numpy.dot(T,B1k)
        LS_stress_sup[:,i] = A2k + B2k * Z[i+1]
        LS_stress_inf[:,i] = A2k + B2k * Z[i]


    # Outputs the stresses and strains vectors.
    # LCS = Laminate System; MCS = Material Coordinate System;
    # inf = inferior; sup = superior.
    return {"LCS" : {"stress" : {"inf" : LS_stress_inf,
                                 "sup" : LS_stress_sup},
                     "strain" : {"inf" : LS_strain_inf,
                                 "sup" : LS_strain_sup}},
            "MCS" : {"stress" : {"inf" : MS_stress_inf,
                                 "sup" : MS_stress_sup},
                     "strain" : {"inf" : MS_strain_inf,
                                 "sup" : MS_strain_sup}}}



def assemble_Z(lam):
    """ Assembles Z vector which contains z coordinates of laminate. """

    if not isinstance(lam, dict):
        raise LaminateLayupError('Laminate has to be a dictionary.')

    # Get number of layers
    num = len(lam["ang"])
    total_thk = numpy.sum(lam["thk"])
    Z = numpy.zeros(num + 1)
    Z[0] = -total_thk/2

    for i in range(num):
        Z[i + 1] = Z[i] + lam["thk"][i]

    return Z


def calc_thermal_forces(mat_list, lam, Z, fail_list = None, dT = 0):
    """ Calculates force resultants due to temperature variations. """
```

```
    if (not (isinstance(mat_list, (tuple, list))) or
    not (isinstance(lam, dict)) or
    not (isinstance(Z, numpy.ndarray))):
        raise LaminateLayupError('invalid input(s) for this function')

    num = len(lam["ang"])

    Nt = numpy.zeros(6)
    a = numpy.zeros(3)
    a_LCS = numpy.zeros(3)


    for i in range(num):
        mat_id = lam["mat_id"][i]
        mat_prop = mat_list[mat_id]
        ang = lam["ang"][i]

        #alpha (material coord system) vector
        a = [mat_prop["a1"], mat_prop["a2"], 0]

        T = assemble_matrixT(ang)
        Ti = numpy.linalg.inv(T)

        a_LCS = numpy.dot(Ti, a)

        # Checks if there's failure list
        if isinstance(fail_list, list):
            Q = assemble_matrixQ(mat_prop, fail_list[i])
        else:
            Q = assemble_matrixQ(mat_prop)

        Qbar = numpy.dot(numpy.dot(Ti, Q), T)

        Nt[:3] = Nt[:3] + numpy.dot(Qbar, a_LCS) * dT * (Z[i+1] -Z[i])
        Nt[3:6] = Nt[3:6] + numpy.dot(Qbar, a_LCS) * dT * (1/2) * \ (Z[i+1]**2 -Z[i]**2)

    return Nt


def assemble_matrixT(angle):
    """ Assembles T matrix (angles transformation matrix LCS -> MCS). """

    if not isinstance(angle, (int, float)) or not (-360 <= angle <= 360):
        raise LaminateLayupError("lamina angle is not between +- 360 degrees"+
                                 " or it is not an int/float")

    #Transforms angle (degrees) to angle (radians)
    angle = numpy.pi*angle/180

    cos = numpy.cos(angle)
    sin = numpy.sin(angle)
    cs = cos*sin
    cc = cos**2
    ss = sin**2

    T = numpy.zeros((3, 3))
    T = numpy.array([[cc, ss, 2*cs ],
                     [ss, cc, -2*cs ],
                     [-cs, cs, cc-ss]])
    return T


def assemble_matrixTinv(angle):
    """ Assembles T matrix (angles transformation matrix LCS -> MCS). """
```

```
    if not isinstance(angle, (int, float)) or not (-360 <= angle <= 360):
        raise LaminateLayupError("lamina angle is not between +- 360 degrees"+
                                 " or it is not an int/float")

    #Transforms angle (degrees) to angle (radians)
    angle = numpy.pi*angle/180

    cos = numpy.cos(angle)
    sin = numpy.sin(angle)
    cs = cos*sin
    cc = cos**2
    ss = sin**2

    Tinv = numpy.zeros((3, 3))
    Tinv = numpy.array([[cc, ss, -2*cs ],
                        [ss, cc, 2*cs ],
                        [cs, -cs, cc-ss]])
    return Tinv


def assemble_matrixQ (mat_prop, fail_type = None):
    """ Assembles Q matrix (reduced elastic matrix) for a given layer. """

    if not isinstance(mat_prop, dict):
        raise TypeError('mat_prop must be a dictionary')


    E1 = mat_prop["E1"]
    E2 = mat_prop["E2"]
    n12 = mat_prop["n12"]
    G12 = mat_prop["G12"]
    n21 = n12*E2/E1

    Q11 = E1/(1 -n12*n21)
    Q12 = n12*E1*E2 / (E1 -(n12 ** 2) * E2)
    Q22 = E1*E2 / (E1 -(n12 ** 2) * E2)
    Q66 = G12

    Q = numpy.zeros((3, 3))
    Q = numpy.array([[Q11, Q12, 0],
                    [Q12, Q22, 0],
                    [0, 0, Q66]])

    return Q


def assemble_matrixQbar (mat_prop,angle):
    """ Assembles Q matrix (reduced elastic matrix) for a given layer. """

    if not isinstance(mat_prop, dict):
        raise TypeError('mat_prop must be a dictionary')


    E1 = mat_prop["E1"]
    E2 = mat_prop["E2"]
    n12 = mat_prop["n12"]
    G12 = mat_prop["G12"]
    n21 = n12*E2/E1

    Q11 = E1/(1 -n12*n21)
    Q12 = n12*E1*E2 / (E1 -(n12 ** 2) * E2)
    Q22 = E1*E2 / (E1 -(n12 ** 2) * E2)
    Q66 = G12
```

```
    Q = numpy.zeros((3, 3))
    Q = numpy.array([[Q11, Q12, 0],
                     [Q12, Q22, 0],
                     [0, 0, Q66]])


    angle = numpy.pi*angle/180

    cos = numpy.cos(angle)
    sin = numpy.sin(angle)
    cs = cos*sin
    cc = cos**2
    ss = sin**2

    T = numpy.zeros((3, 3))
    T = numpy.array([[cc, ss, 2*cs],
                     [ss, cc, -2*cs],
                     [-cs, cs, cc-ss]])
    Tinv = numpy.array([[cc, ss, -2*cs],
                        [ss, cc, 2*cs],
                        [cs, -cs, cc-ss]])
    R = numpy.array([[1,0,0],[0,1,0],[0,0,2]])
    Rinv = numpy.linalg.inv(R)

    Qbar = numpy.dot(numpy.dot(numpy.dot(numpy.dot(Tinv, Q), R), T), Rinv)

    return Qbar


def assemble_ABD(mat_list, lam, Z, fail_list = None):
    """ Assembles ABD matrix (laminate stiffness matrix). """

    if (not (isinstance(mat_list, (tuple, list))) or
    not (isinstance(lam, dict)) or
    not (isinstance(Z, numpy.ndarray))):
        raise LaminateLayupError('invalid input(s) for this function')

    num = len(lam["ang"])
    A = B = D = numpy.zeros((3,3))
    Z = numpy.flip(Z)
    for i in range(num):
        mat_id = lam["mat_id"][i]
        mat_prop = mat_list[mat_id]
        ang = lam["ang"][i]

        Qi = assemble_matrixQbar(mat_prop,ang)

        A = A + Qi*(Z[i+1] -Z[i])
        B = B + (1/2)*Qi*(Z[i+1]**2 -Z[i]**2)
        D = D + (1/3)*Qi*(Z[i+1]**3 -Z[i]**3)


    ABD = numpy.zeros((6,6))
    ABD[:3, :3] = -A
    ABD[:3, 3:6] = ABD[3:6, :3] = -B
    ABD[3:6, 3:6] = -D

    return ABD

def ABCD_matrix(ABD):
    A = ABD[0:3,0:3]
    B = ABD[0:3,3:]
    D = ABD[3:,3:]
```

```
    Astar = numpy.linalg.inv(A)
    Bstar = -1 * numpy.linalg.inv(A) * B
    Hstar = B * numpy.linalg.inv(A)
    Dstar = D -B * numpy.linalg.inv(A) * B

    Aprime = Astar -Bstar * numpy.linalg.inv(Dstar) * Hstar
    Bprime = Bstar * numpy.linalg.inv(Dstar)
    Hprime = -1 * numpy.linalg.inv(Dstar) * Hstar
    Dprime = numpy.linalg.inv(Dstar)

    AB = numpy.concatenate((Aprime,Bprime),axis=1)
    CD = numpy.concatenate((Hprime,Dprime),axis=1)
    ABCD = numpy.concatenate((AB,CD),axis=0)
    return ABCD
```

```
import numpy as np
import clt_functions as clt
import CLT_matprops as CLT_matprops


def DataExtraction(lam, res, coord_sys, axis, var):
    # Sets obligatory data for the instance.
    lam = lam

    # Gets number of layers and loadsteps
    num_layers = len(lam["ang"])

    # Isolates stress & strain results from array
    res = res

    Z = clt.assemble_Z(lam)


    X = { "inf" : np.zeros((num_layers)),
          "sup" : np.zeros((num_layers)) }
    Y = { "inf" : np.zeros((num_layers)),
          "sup" : np.zeros((num_layers)) }

    P = np.zeros((num_layers*2, 2))


    # Iterates the layers, add data to plot
    for layer in range(num_layers):
        X["inf"][layer] = res[coord_sys][var]["inf"][axis][layer]
        Y["inf"][layer] = Z[layer]
        X["sup"][layer] = res[coord_sys][var]["sup"][axis][layer]
        Y["sup"][layer] = Z[layer + 1]
        P[layer*2] = [X["inf"][layer], Y["inf"][layer]]
        P[layer*2 + 1] = [X["sup"][layer], Y["sup"][layer]]

    return P

def clt_form_vf(vf,mat_id_num,coord_sys,var,axis):

    prepreg = CLT_matprops.prepreg
    td_bulk = CLT_matprops.td_bulk
    epoxy = CLT_matprops.epoxy
    delam = CLT_matprops.delam
    td_voigt = CLT_matprops.td_voigt_pc(vf)
    td_reuss = CLT_matprops.td_reuss_pc(vf)
    td_hsub = CLT_matprops.td_hsub_pc(vf)
    td_hslb = CLT_matprops.td_hslb_pc(vf)
```

```
td_mt = CLT_matprops.td_mt_pc(vf)


##################################################

# Assembles material list
mat = [prepreg, td_bulk, epoxy, delam, td_voigt, td_reuss, td_hsub, td_hslb, td_mt, []]

##################################################

# Initializes dictionary of the laminate layup configurations
# thk is thickness; ang is angle; mat_id is material id
lam = {'thk': [], 'ang': [], 'mat_id' : []}

lam['ang'] = [0]
lam['thk'] = [prepreg['thickness']]
lam['mat_id'] = [mat_id_num]


##################################################

# Vector F with the applied generalized stress (unit N/m / N.m/m)
F = np.array([ 0, # Nx
               0, # Ny
               0, # Nxy
               0, # Mx
               0, # My
               0]) # Mxy

# F = F * pow(10,3)

##################################################

# Temperature and moisture variations
delta_T = -120
delta_M = 0

res = clt.calc_stressCLT(mat, lam, F, None, delta_T, delta_M)
P = DataExtraction(lam, res, coord_sys, axis, var)
vf_datapoint = [vf, P[0,0]]

return vf_datapoint
```

D.4 General Machine Learning Script

The script presented in this section was responsible for the SVM (SVC) machine learning algorithms. Modification of the algorithm chose required changing the models imported from the [37] package in addition to changes in the parametric parameters. For instance, a k-NN algorithm required that the number of nearest neighbors k be defined. For a MLP, the number of nodes and hidden layers were specificed in addition to the activation function. A looping structure was scripted such that the script automatically cycled through a list of given parameters. The Python packages discussed in [37, 126, 169, 170] were employed in this script.

```
import os
import sys
import shutil
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn as sklearn
from sklearn import svm
from sklearn import model_selection
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn import metrics
from sklearn.metrics import plot_confusion_matrix
# from sklearn.model_selection import KFold, cross_val_score
import pickle
import custom_confusionmatrix as cm
import model_stats




def get_model(kernel):
    main_model = svm.SVC(kernel=kernel, C=1)
    model = make_pipeline(StandardScaler(), main_model)
    # model = svm.SVC(kernel=kernel, C=1)
    return model

def evaluate_model(kernel,cv):
    model = get_model(kernel)
    score = model_selection.cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
    score = np.array([(score)])
    return score

def model_statistics(score):
    mean_fold = np.mean(score)
    min_fold = np.min(score)
    max_fold = np.max(score)

    mean_fold = np.asarray(mean_fold)
    min_fold = np.asarray(min_fold)
    max_fold = np.asarray(max_fold)

    minmean_fold = np.abs(min_fold -mean_fold)
    maxmean_fold = np.abs(max_fold -mean_fold)

    df_score = pd.DataFrame(score)

    return mean_fold, min_fold, max_fold, minmean_fold, maxmean_fold, df_score



## model parameters
kernels = ['linear', 'poly', 'rbf', 'sigmoid'] # rbf -default
kernel_type = kernels[3]
folds_consideration = 30
model_string = 'SVM_SVC' + '_' + kernel_type


## select file to import & import
author = 'Rudd'
filename = 'master_delam_' + author + '.csv'
filename = os.path.join(os.getcwd(),'ML_Datasets',filename)
df = pd.read_csv(filename)
```

```
## extract desired data
X = df.drop(['source', 'fiber_material', 'matrix_material', 'stacking_sequence', 'voltage_peak', '
    ↪ voltage_RMS', 'delamination', 'delam_length', 'delam_width'], axis=1)
X = np.array(X)
predict_label = "delamination"
y = np.array(df[predict_label])


## setup & run model
x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(X, y, test_size=0.1)
num_iterations = 30
best = 0


# model_string = 'SVM_SVC' + '_' + kernel_type
iter_string = model_string + '_' + str(num_iterations) + 'iterations.csv'
stats_header = ['iteration', 'accuracy_best', 'accuracy_current', 'precision', 'recall', 'f1_score']
stats = pd.DataFrame([])

for i in range(num_iterations):
    x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(X, y, test_size=0.1,
        ↪ random_state=None)

    model = get_model(kernel_type)
    model.fit(x_train, y_train)

    predicted = model.predict(x_test)
    acc = metrics.accuracy_score(y_test, predicted)
    f1 = metrics.f1_score(y_test, predicted)
    precision = metrics.precision_score(y_test, predicted, zero_division=0)
    recall = metrics.recall_score(y_test, predicted)

    if acc > best:
        best = acc
        iter_best_string = model_string + '_' + str(num_iterations) + 'iterations_BEST.pickle'
        with open(iter_best_string, "wb") as f:
            pickle.dump(model, f)

    iter_stats = np.array([[int(i), best, acc, precision, recall, f1]])
    stats = stats.append(pd.DataFrame(iter_stats))
    # print('Current: ', acc, "Best: ", best)


## output iteration stats to .csv file
stats.columns = stats_header
stats.to_csv(iter_string,index=False)

folds = range(2,len(X))

cv_acc = pd.DataFrame([])
cv_mean = np.array([])
cv_min = np.array([])
cv_max = np.array([])
cv_minmean = np.array([])
cv_maxmean = np.array([])


## iterate number of folds
for k in folds:
    cv = model_selection.KFold(n_splits=k, shuffle=False, random_state=None)

    model = get_model(kernel_type)
    score = evaluate_model(kernel_type, cv)
```

```
    mean_fold, min_fold, max_fold, minmean_fold, maxmean_fold, df_score = model_statistics(score)

    cv_acc = cv_acc.append(pd.DataFrame(score))
    cv_mean = np.append(cv_mean, mean_fold)
    cv_min = np.append(cv_min, min_fold)
    cv_max = np.append(cv_max, max_fold)
    cv_minmean = np.append(cv_minmean, minmean_fold)
    cv_maxmean = np.append(cv_maxmean, maxmean_fold)


# maximum number of folds (LeaveOneOut method)
ideal_score = evaluate_model(kernel_type,model_selection.LeaveOneOut())
ideal_mean, ideal_min, ideal_max, ideal_minmean, ideal_maxmean, ideal_df_score = model_statistics(
    ↪ ideal_score)


## efficient number of folds
mean_diff = np.abs(ideal_mean -cv_mean[0:folds_consideration])
eff_index = np.argmin(mean_diff)
eff_numfolds = folds[eff_index]

print('Efficient number of folds: {}'.format(eff_numfolds))


## export fold scheme performance (accuracy) per fold
cv_acc.index = folds
cv_score_string = str(model_string + '_' + str(np.max(folds)) + 'folds.csv')
cv_acc.to_csv(cv_score_string, index=True, header=None)

cv_stats = pd.DataFrame([cv_mean, cv_min, cv_max, cv_minmean, cv_maxmean]).transpose()
cv_stats.columns = ['mean', 'min', 'max', 'min-mean', 'max-mean']
cv_stats.index = folds
cv_statscore_string = str(model_string + '_' + str(np.max(folds)) + 'folds_stats.csv')
cv_stats.to_csv(cv_statscore_string, index=True)

model_stats.model_scores(model_name=model_string,eff_fold_number=eff_numfolds,eff_fold_mean=cv_mean[
    ↪ eff_index],LOO_mean=ideal_mean)


## errorbar plot (all folds)
errorbar_plot_filename = str(model_string + '_' + str(np.max(folds)) + 'folds.png')
plt.figure(num=1, figsize=(8, 6), dpi=80, facecolor='w', edgecolor='k') # create a figure object
plt.errorbar(folds, cv_mean, yerr=[cv_minmean, cv_maxmean], fmt='o')
plt.plot(folds, [ideal_mean for _ in range(len(folds))], color='r')
plt.xlabel('Number of Folds', fontsize=14)
plt.ylabel('Mean Accuracy of Model', fontsize=14)
plt.tight_layout()
plt.savefig(errorbar_plot_filename)


## errorbar plot (first 20 folds)
eff_num_iter = folds_consideration-1
errorbareff_plot_filename = str(model_string + '_2-' + str(folds_consideration) + 'folds.png')
plt.figure(num=2, figsize=(8, 6), dpi=80, facecolor='w', edgecolor='k') # create a figure object
plt.errorbar(folds[:eff_num_iter], cv_mean[:eff_num_iter], yerr=[cv_minmean[:eff_num_iter], cv_maxmean
    ↪ [:eff_num_iter]], fmt='o')
plt.plot(folds[:eff_num_iter], [ideal_mean for _ in range(len(folds[:eff_num_iter]))], color='r')
plt.xlabel('Number of Folds', fontsize=14)
plt.ylabel('Mean Accuracy of Model', fontsize=14)
plt.xticks(range(0,folds_consideration+3))
plt.tight_layout()
plt.savefig(errorbareff_plot_filename)
```

```
## confusion matrix for most efficient fold
confusion_matrix_plot_filename = str(model_string + '_ConfusionMatrix_' + str(eff_numfolds) + 'folds.
    ↪ png')
cm.plot_custom_confusion_matrix(get_model(kernel=kernel_type),X,y,cv=eff_numfolds)
plt.xlabel('Predicted Label',fontsize=14)
plt.ylabel('True Label',fontsize=14)
plt.tight_layout()
plt.savefig(confusion_matrix_plot_filename)
```

D.4.1 Model Statistics

Functions were defined to create .csv files with the accuracy metric scores of the cross-validation experiments for each algorithm.

```
import os
import shutil
import pandas as pd
import numpy as np
from datetime import datetime


## create .csv of model/cv data
columns=['model_name','efficient_fold','efficient_fold_mean_score','LOO_mean','timestamp']

def model_scores(model_name, eff_fold_number, eff_fold_mean, LOO_mean):
    model_dict = {'model_name' : model_name,
    'efficient_fold' : eff_fold_number,
    'efficient_fold_mean_score' : eff_fold_mean,
    'LOO_mean' : LOO_mean,
    'time' : datetime.now()}

    filename = os.path.join(os.getcwd(),'Scores','model_fold_data.csv')
    df = pd.read_csv(filename)
    row_index = df[df['model_name'] == model_name]

    if not row_index.empty:
        print(' > > > ' + model_name + ' entry already exists. Updating now.....')

        df.loc[df.model_name == model_name, 'efficient_fold'] = eff_fold_number
        df.loc[df.model_name == model_name, 'efficient_fold_mean_score'] = eff_fold_mean
        df.loc[df.model_name == model_name, 'LOO_mean'] = LOO_mean
        df.loc[df.model_name == model_name, 'time'] = datetime.now()

    else:
        print(' > > > ' + model_name + ' does not exist. Adding entry to file now.....')
        df = df.append(model_dict, ignore_index=True)

    ## sort df by models
    df = df.sort_values(by=['model_name'])

    ## export df
    df.to_csv(filename,index=False)


## record info about MLP
columns=['model_name','efficient_fold','efficient_fold_mean_score','LOO_mean','runtime','timestamp']
def MLP_info(model_name, eff_fold_number, eff_fold_mean, LOO_mean, runtime):
    model_dict = {'model_name' : model_name,
```

```
    'efficient_fold' : eff_fold_number,
    'efficient_fold_mean_score' : eff_fold_mean,
    'LOO_mean' : LOO_mean,
    'runtime' : runtime,
    'timestamp' : datetime.now()}

    filename = os.path.join(os.getcwd(),'Scores','MLP_fold_data.csv')
    df = pd.read_csv(filename)
    row_index = df[df['model_name'] == model_name]

    if not row_index.empty:
        print(' > > > > ' + model_name + ' entry already exists. Updating now..........')

        df.loc[df.model_name == model_name, 'efficient_fold'] = eff_fold_number
        df.loc[df.model_name == model_name, 'efficient_fold_mean_score'] = eff_fold_mean
        df.loc[df.model_name == model_name, 'LOO_mean'] = LOO_mean
        df.loc[df.model_name == model_name, 'runtime'] = runtime
        df.loc[df.model_name == model_name, 'timestamp'] = datetime.now()

    else:
        print(' > > > > ' + model_name + ' does not exist. Adding entry to file now..........')
        df = df.append(model_dict, ignore_index=True)

    ## sort df by models
    df = df.sort_values(by=['model_name'])

    ## export df
    df.to_csv(filename,index=False)
```

D.4.2 Confusion Matrix

A modification to the confusion matrix script given by [37] was made to explicitly label the axes for the given class labels.

```
from itertools import product
import numpy as np
import sklearn
from sklearn.utils import check_matplotlib_support
from sklearn.metrics import confusion_matrix
from sklearn.utils.validation import _deprecate_positional_args
from sklearn.base import is_classifier
import matplotlib as mpl

class ConfusionMatrixDisplay:
    """Confusion Matrix visualization.
    It is recommend to use :func:`~sklearn.metrics.plot_confusion_matrix` to
    create a :class:`ConfusionMatrixDisplay`. All parameters are stored as
    attributes.
    Read more in the :ref:`User Guide <visualizations>`.
    Parameters
    ----------
    confusion_matrix : ndarray of shape (n_classes, n_classes)
        Confusion matrix.
    display_labels : ndarray of shape (n_classes,), default=None
        Display labels for plot. If None, display labels are set from 0 to
        `n_classes -1`.
    Attributes
    ----------
    im_ : matplotlib AxesImage
```

```
    Image representing the confusion matrix.
text_ : ndarray of shape (n_classes, n_classes), dtype=matplotlib Text, \
        or None
    Array of matplotlib axes. `None` if `include_values` is false.
ax_ : matplotlib Axes
    Axes with confusion matrix.
figure_ : matplotlib Figure
    Figure containing the confusion matrix.
"""
def __init__(self, confusion_matrix, *, display_labels=None):
    self.confusion_matrix = confusion_matrix
    self.display_labels = display_labels

@_deprecate_positional_args
def plot(self, *, include_values=True, cmap='Blue',
         xticks_rotation='horizontal', values_format=None, ax=None):
    """Plot visualization.
    Parameters
    ----------
    include_values : bool, default=True
        Includes values in confusion matrix.
    cmap : str or matplotlib Colormap, default='viridis'
        Colormap recognized by matplotlib.
    xticks_rotation : {'vertical', 'horizontal'} or float, \
                      default='horizontal'
        Rotation of xtick labels.
    values_format : str, default=None
        Format specification for values in confusion matrix. If `None`,
        the format specification is 'd' or '.2g' whichever is shorter.
    ax : matplotlib axes, default=None
        Axes object to plot on. If `None`, a new figure and axes is
        created.
    Returns
    -------
    display : :class:`~sklearn.metrics.ConfusionMatrixDisplay`
    """
    check_matplotlib_support("ConfusionMatrixDisplay.plot")
    import matplotlib.pyplot as plt

    if ax is None:
        fig, ax = plt.subplots(figsize=(8,6))
    else:
        fig = ax.figure

    cm = self.confusion_matrix
    n_classes = cm.shape[0]
    self.im_ = ax.imshow(cm, interpolation='nearest', cmap=cmap)
    self.text_ = None
    cmap_min, cmap_max = self.im_.cmap(0), self.im_.cmap(256)

    if include_values:
        self.text_ = np.empty_like(cm, dtype=object)

        # print text with appropriate color depending on background
        thresh = (cm.max() + cm.min()) / 2.0

        for i, j in product(range(n_classes), range(n_classes)):
            color = cmap_max if cm[i, j] < thresh else cmap_min

            if values_format is None:
                text_cm = format(cm[i, j], '.2g')
                if cm.dtype.kind != 'f':
                    text_d = format(cm[i, j], 'd')
                    if len(text_d) < len(text_cm):
```

```
                        text_cm = text_d
                else:
                    text_cm = format(cm[i, j], values_format)

                self.text_[i, j] = ax.text(
                    j, i, text_cm,
                    ha="center", va="center",
                    color=color,fontsize=24,weight='bold')

        if self.display_labels is None:
            display_labels = np.arange(n_classes)
        else:
            display_labels = self.display_labels

        # fig.colorbar(self.im_, ax=ax)
        ax.set(xticks=np.arange(n_classes),
               yticks=np.arange(n_classes),
               xticklabels=display_labels,
               yticklabels=display_labels,
               ylabel="True label",
               xlabel="Predicted label")

        ax.set_ylim((n_classes -0.5, -0.5))
        plt.setp(ax.get_xticklabels(), rotation=xticks_rotation)

        self.figure_ = fig
        self.ax_ = ax
        return self


def plot_custom_confusion_matrix(model, X, y_true, cv, *, labels=None,
                         sample_weight=None, normalize=None,
                         display_labels=None, include_values=True,
                         xticks_rotation='horizontal',
                         values_format=None,
                         cmap='Blues', ax=None):

    y_pred = sklearn.model_selection.cross_val_predict(model, X, y_true, cv=cv)

    cm = sklearn.metrics.confusion_matrix(y_true, y_pred, sample_weight=sample_weight,
                         labels=labels, normalize=normalize)

    display_labels = ['No Delamination', 'Delamination']

    disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                                  display_labels=display_labels)
    return disp.plot(include_values=include_values,
                     cmap=cmap, ax=ax, xticks_rotation=xticks_rotation,
                     values_format=values_format)
```

Bibliography

[1] R. M. Jones, *Mechanics of Composite Materials*. CRC Press, 2nd ed., 1999.

[2] I. Mazínová and P. Florian, *Materials selection in mechanical design*, vol. 16. Oxford, United Kingdom; Cambridge, MA: Butterworth-Heinemann, 5th ed., 2014.

[3] M. A. Haile, A. J. Hall, J. H. Yoo, M. D. Coatney, and O. J. Myers, "Detection of damage precursors with embedded magnetostrictive particles," *Journal of Intelligent Material Systems and Structures*, vol. 27, pp. 1567–1576, Jul 2016.

[4] G. N. Weisensel, "Magnetotagging, a new method for active and passive nondestructive evaluation of stresses and defects in composite structures," *Materials Technology*, vol. 10, pp. 142–144, Jan 1995.

[5] E. d. T. de Lacheisserie, *Magnetostriction: Theory and Applications of Magnetoelasticity*, vol. 97. Boca Raton, FL: CFC Press, 1993.

[6] O. J. Myers, G. Currie, J. Rudd, D. Spayde, and N. W. Bolden, "Damage detection of unidirectional carbon fiber-reinforced laminates with embedded magnetostrictive particulates: A preliminary study," *Journal of Intelligent Material Systems and Structures*, vol. 24, pp. 991–1006, May 2013.

[7] J. Rudd and O. Myers, "Experimental fabrication and nondestructive testing of carbon fiber beams for delaminations using embedded Terfenol-D particles," *Journal of Intelligent Material Systems and Structures*, vol. 29, pp. 600–609, Mar 2018.

[8] F. T. Calkins, A. B. Flatau, and M. J. Dapino, "Overview of magnetostrictive sensor technology," *Journal of Intelligent Material Systems and Structures*, vol. 18, pp. 1057–1066, Oct 2007.

[9] R. Hill, "Elastic properties of reinforced solids: Some theoretical principles," *Journal of the Mechanics and Physics of Solids*, vol. 11, pp. 357–372, Sept 1963.

[10] R. A. Schapery, "Thermal Expansion Coefficients of Composite Materials Based on Energy Principles," *Journal of Composite Materials*, vol. 2, pp. 380–404, Jul 1968.

[11] I. Daniel, J. Whitney, and R. Pipes, "Experimental Mechanics of Fiber Reinforced Composite Materials," *Experimental Techniques*, vol. 7, pp. 25–25, Mar 1983.

[12] Z. Hashin, "Analysis of composite materials: A survey," *Journal of Applied Mechanics, Transactions ASME*, vol. 50, pp. 481–505, Sept 1983.

[13] L. X. Li and T. J. Wang, "A unified approach to predict overall properties of composite materials," *Materials Characterization*, vol. 54, pp. 49–62, Jan 2005.

[14] J. N. Reddy, *Theory and analysis of laminated composite plates*. Boca Raton, FL: CRC Press, 1999.

[15] A. T. Nettles, "Basic Mechanics of Laminated Composite Plates," Tech. Rep. October, NASA Marshall Space Flight Center, Huntsville, Alabama, United States of America, 1994.

[16] G. Kemmann and O. Myers, *An Experimental Investigation of Combined Symmetric-Asymmetric Composite Laminates*. Master of science thesis, Clemson Uniiversity, 2019.

[17] J. A. Hommema, *Magnetomechanical Behavior of Terfenol-D Particulate Composites*. Master of science thesis, University of Illinois at Urbana-Champaign, 1999.

[18] Z. Deng, *Nonlinear Modeling and Characterization of the Villari Effect and Model-guided Development of Magnetostrictive Energy Harvesters and Dampers*. Doctor of philosophy dissertation, Ohio State University, 2015.

[19] A. Clark, "Chapter 7 Magnetostrictive rare earth-Fe2 compounds," in *Handbook of Ferromagnetic Materials*, ch. Chapter 7, pp. 531–589, Silver Spring, Maryland, United States of America: North-Holland Publishing Company, 1980.

[20] I. The Institute of Electrical and Electronics Engineers, "IEEE Standard on Magnetostrictive Materials: Piezomagnetic Nomenclature," *IEEE Transactions on Sonics and Ultrasonics*, vol. 20, no. 1, pp. 67–77, 1973.

[21] J. H. Alexander and O. J. Myers, "Microstructure Properties and Strengthening Mechanisms of the AS4-3501-6 Polymeric Resin With Embedded Terfenol-D Particles," in *Proceedings of the ASME 2014 Conference on Smart Materials, Adaptive Structures and Intelligent Systems*, (Newport, Rhode Island, USA), American Society of Mechanical Engineers, Sept 2014.

[22] L. A. Dobrzański, A. E. Tomiczek, A. Szewczyk, K. Piotrowski, M. U. Gutowska, and J. Wieckowski, "Physical properties of magnetostrictive composite materials with the polyurethane matrix," *Archives of Materials Science and Engineering*, vol. 57, no. 1, pp. 21–27, 2012.

[23] J. Kaleta, D. Lewandowski, R. Mech, and P. Gąsior, "Magnetomechanical properties of Terfenol-D powder composites," *Solid State Phenomena*, vol. 154, pp. 35–40, 2009.

[24] R. Kellogg and A. Flatau, "Experimental investigation of terfenol-D's elastic modulus," *Journal of Intelligent Material Systems and Structures*, vol. 19, pp. 583–595, May 2008.

[25] ETREMA Products, "ETREMA Products Terfenol-D Magnetostrictive Smart Material," 2019.

[26] H. J. Götze, "Curie temperature," tech. rep., University of Oklahoma, Norman, Oklahoma, 2016.

[27] B. Williams, *Terfenol-D Carbon Fiber Reinforced Polymer (CFRP) Embedded Sensing for Early Damage Detection*. Master of science thesis, Clemson University, 2019.

[28] G. Engdahl, *Handbook of Giant Magnetostrictive Materials*. San Diego, California: London : Academic, 2000.

[29] M. Roos, "Deep Learning Neurons versus Biological Neurons," Mar 2019.

[30] A. M. Turing, "Computing Machinery and Intelligence," *Mind*, vol. 59, pp. 433–460, Oct 1950.

[31] M. Haenlein and A. Kaplan, "A brief history of artificial intelligence: On the past, present, and future of artificial intelligence," *California Management Review*, vol. 61, pp. 5–14, Aug 2019.

[32] L. Spector, "Evolution of artificial intelligence," *Artificial Intelligence*, vol. 170, pp. 1251–1253, Dec 2006.

[33] M. Kubat, *An Introduction to Machine Learning*. Cham, Switzerland: Springer Nature Switzerland AG, 2017.

[34] S. A. Bini, "Artificial Intelligence, Machine Learning, Deep Learning, and Cognitive Computing: What Do These Terms Mean and How Will They Impact Health Care?," *Journal of Arthroplasty*, vol. 33, pp. 2358–2361, Aug 2018.

[35] J. M. Helm, A. M. Swiergosz, H. S. Haeberle, J. M. Karnuta, J. L. Schaffer, V. E. Krebs, A. I. Spitzer, and P. N. Ramkumar, "Machine Learning and Artificial Intelligence: Definitions, Applications, and Future Directions," *Current Reviews in Musculoskeletal Medicine*, vol. 13, pp. 69–76, Feb 2020.

[36] Oliver Theobald, "Machine Learning For Absolute Beginners," Feb 2017.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[38] X. Goldberg, "Introduction to semi-supervised learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, pp. 1–116, Jan 2009.

[39] M. Sewak, *Deep Reinforcement Learning*. Singapore: Springer Singapore, 2019.

[40] K. Worden, C. R. Farrar, G. Manson, and G. Park, "The fundamental axioms of structural health monitoring," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 463, pp. 1639–1664, Jun 2007.

[41] C. R. Farrar and K. Worden, "An introduction to structural health monitoring," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 365, pp. 303–315, Feb 2007.

[42] B. Culshaw, "Structural health monitoring of civil engineering structures," *Progress in Structural Engineering and Materials*, vol. 1, pp. 308–315, Apr 1998.

[43] J. Lubliner and B. Moran, "Plasticity Theory," *Journal of Applied Mechanics*, vol. 59, pp. 245–246, Mar 1992.

[44] D. D. Le, J. C. Riddick, V. Weiss, B. R. Miller, and N. E. Bordick, ""Fatigue-free" platforms: Vision for army future rotorcraft," in *Annual Forum Proceedings - AHS International*, vol. 4, (Montreal, Canada), pp. 3035–3048, American Helicopter Society International, Inc., 2014.

[45] C. R. Farrar and K. Worden, *Structural Health Monitoring: A Machine Learning Perspective*. New York: Wiley, 2012.

[46] V. M. Karbhari, *Non-Destructive Evaluation (NDE) of Polymer Matrix Composites*, vol. 43. Elsevier Science & Technology, 2013.

[47] H. Sabbagh, "Nondestructive evaluation," *IEEE Potentials*, vol. 13, no. 5, pp. 35–38, 1995.

[48] C. R. Farrar, S. W. Doebling, and D. A. Nix, "Vibration-based structural damage identification," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 359, pp. 131–149, Jan 2001.

[49] A. Rytter, *Vibrational Based Inspection of Civil Engineering Structures*. Phd dissertation, Aalborg University, 1993.

[50] F. M. Ryan and R. C. Miller, "Phosphor combination and method, particularly adapted for use with explosives, for providing a distinctive information label," 1973.

[51] R. D. Hermansen and R. Sutherland, Thomas H. Predmore, "Method of bonding metals with a radio-opaque adhesive/sealant for void detection and product made," 1990.

[52] W. Clark, R. Sadhir, and W. Junker, "Tagged Adhesives for Improved Electromagnetic Inspection," *Materials Evaluation*, vol. 48, no. 1, pp. 60–64, 1990.

[53] A. V. Krishnamurthy, M. Anjanappa, Z. Wang, and X. Chen, "Sensing of delaminations in composite laminates using embedded magnetostrictive particle layers," *Journal of Intelligent Material Systems and Structures*, vol. 10, pp. 825–835, Oct 1999.

[54] Y. Wu and M. Anjanappa, "Modeling of embedded magnetostrictive particulate actuators," in *SPIE 2717, Smart Structures and Materials 1996: Smart Structures and Integrated Systems*, (San Diego, California), SPIE, May 1996.

[55] Y. Wu, *Magnetostrictive particulate actuator and its characterization for smart material applications*. Dissertation, University of Maryland Baltimore County, 1997.

[56] M. Anjanappa and Y. Wu, "Magnetostrictive particulate actuators: Configuration, modeling and characterization," *Smart Materials and Structures*, vol. 6, pp. 393–402, Aug 1997.

[57] J. D. Rudd, *Investigating Nondestructive Evaluation of Carbon Fiber Reinforced Polymer Beams using Embedded Terfenol-D Particle Sensors*. Phd dissertation, Mississippi State University, 2014.

[58] C. Coelho and A. Chattopadhyay, "Feature reduction for computationally efficient damage state classification using binary tree support vector machines," in *Proceedings of the ASME Conference on Smart Materials, Adaptive Structures and Intelligent Systems, SMASIS2008*, vol. 2, pp. 289–296, ASME, Oct 2008.

[59] W. Reynolds, D. Doyle, and B. Arritt, "Insight into active health monitoring methods using machine learning," in *ASME 2011 Conference on Smart Materials, Adaptive Structures and Intelligent Systems, SMASIS 2011*, vol. 2, pp. 535–543, ASME, Sept 2011.

[60] H. Salehi, R. Burgueño, S. Das, S. Biswas, and S. Chakrabartty, "Localized damage identification of plate-like structures with time-delayed binary data from a self-powered sensor network," in *ASME 2017 Conference on Smart Materials, Adaptive Structures and Intelligent Systems, SMASIS 2017*, vol. 2, ASME, Sept 2017.

[61] R. P. Finotti, A. A. Cury, and F. d. S. Barbosa, "An SHM approach using machine learning and statistical indicators extracted from raw dynamic measurements," *Latin American Journal of Solids and Structures*, vol. 16, no. 2, 2019.

[62] E. Figueiredo, G. Park, C. R. Farrar, K. Worden, and J. Figueiras, "Machine learning algorithms for damage detection under operational and environmental variability," *Structural Health Monitoring*, vol. 10, pp. 559–572, Nov 2011.

[63] M. Saffari, R. Sedaghati, and I. Stiharu, "Structural health monitoring of truss type structures using statistical approach," in *ASME 2014 Conference on Smart Materials, Adaptive Structures and Intelligent Systems, SMASIS 2014*, vol. 1, ASME, Sept 2014.

[64] A. Amaya, J. Alvarez-Montoya, and J. Sierra-Pérez, "Development of a structural health monitoring methodology in reinforced concrete structures using FBGs and pattern recognition techniques," in *ASME 2019 Conference on Smart Materials, Adaptive Structures and Intelligent Systems, SMASIS 2019*, ASME, Sept 2019.

[65] S. Li, P. Li, Y. Zhang, and X. Zhao, "Detection of component types and track damage for high-speed railway using region-based convolutional neural networks," in *ASME 2018 Conference on Smart Materials, Adaptive Structures and Intelligent Systems, SMASIS 2018*, vol. 2, ASME, Sept 2018.

[66] X. Zhao, S. Li, H. Su, L. Zhou, and K. J. Loh, "Image-based comprehensive maintenance and inspection method for bridges using deep learning," in *ASME 2018 Conference on Smart Materials, Adaptive Structures and Intelligent Systems, SMASIS 2018*, vol. 2, ASME, 2018.

[67] M. Calabrese, M. Cimmino, M. Manfrin, F. Fiume, D. Kapetis, M. Mengoni, S. Ceccacci, E. Frontoni, M. Paolanti, A. Carrotta, and G. Toscano, "An event based machine learning framework for predictive maintenance in industry 4.0," in *Proceedings of the ASME Design Engineering Technical Conference*, vol. 9, ASME, Aug 2019.

[68] G. X. Gu, L. Dimas, Z. Qin, and M. J. Buehler, "Optimization of Composite Fracture Properties: Method, Validation, and Applications," *Journal of Applied Mechanics, Transactions ASME*, vol. 83, pp. 71006–71007, Jul 2016.

[69] G. X. Gu, C. T. Chen, and M. J. Buehler, "De novo composite design based on machine learning algorithm," *Extreme Mechanics Letters*, vol. 18, pp. 19–28, Jan 2018.

[70] H. T. Fan and H. Wang, "Predicting the Open-Hole Tensile Strength of Composite Plates Based on Probabilistic Neural Network," *Applied Composite Materials*, vol. 21, pp. 827–840, Dec 2014.

[71] Z. Jiang, L. Gyurova, Z. Zhang, K. Friedrich, and A. K. Schlarb, "Neural network based prediction on mechanical and wear properties of short fibers reinforced polyamide composites," *Materials and Design*, vol. 29, no. 3, pp. 628–637, 2008.

[72] B. Wang, S. Zhong, T. L. Lee, K. S. Fancey, and J. Mi, "Non-destructive testing and evaluation of composite materials/structures: A state-of-the-art review," *Advances in Mechanical Engineering*, vol. 12, pp. 1–28, Apr 2020.

[73] A. O. Addin, S. M. Sapuan, E. Mahdi, and M. Othman, "Prediction and detection of failures in laminated composite materials using neural networks - A review," *Polymers and Polymer Composites*, vol. 14, pp. 433–442, May 2006.

[74] O. J. Myers and S. Banerjee, "Coupled damage precursor detection," in *ASME 2015 Conference on Smart Materials, Adaptive Structures and Intelligent Systems, SMASIS 2015*, vol. 2, American Society of Mechanical Engineers, Sept 2015.

[75] K. Makarian, S. Santhanam, and Z. N. Wing, "Coefficient of thermal expansion of particulate composites with ceramic inclusions," *Ceramics International*, vol. 42, pp. 17659–17665, Nov 2016.

[76] D. Sammons, W. P. Winfree, E. Burke, and S. Ji, "Segmenting delaminations in carbon fiber reinforced polymer composite CT using convolutional neural networks," *AIP Conference Proceedings*, vol. 1706, Feb 2016.

[77] J. Vitola, F. Pozo, D. A. Tibaduiza, and M. Anaya, "A sensor data fusion system based on k-nearest neighbor pattern classification for structural health monitoring applications," *Sensors (Switzerland)*, vol. 17, Feb 2017.

[78] L. Rodriguez-Cobo, J. Mirapeix, A. Cobo, and J. M. Lopez-Higuera, "Comparison of hierarchical temporal memories and artificial neural networks under noisy data," *Journal of Intelligent Material Systems and Structures*, vol. 26, pp. 1243–1250, Jul 2015.

[79] D. Webb, "Fiber Bragg Grating Sensors," in *Handbook of Optical Sensors* (A. Cusano, C. Antonello, and J. Albert, eds.), pp. 503–532, CRC Press, Oct 2014.

[80] FBGS Technologies, "FBG principle," 2014.

[81] D. Chakraborty, "Artificial neural network based delamination prediction in laminated composites," *Materials and Design*, vol. 26, no. 1, pp. 1–7, 2005.

[82] M. T. Valoor and K. Chandrashekhara, "A thick composite-beam model for delamination prediction by the use of neural networks," *Composites Science and Technology*, vol. 60, no. 9, pp. 1773–1779, 2000.

[83] Z. Zhang, O. K. Ihesiulor, K. Shankar, and T. Ray, "Comparison of inverse algorithms for delamination detection in composite laminates," in *ASME 2012 Conference on Smart Materials, Adaptive Structures and Intelligent Systems, SMASIS 2012*, vol. 1, pp. 577–585, ASME, Sept 2012.

[84] Z. Zhao, M. Yua, and S. Dong, "Damage location detection of the CFRP composite plate based on neural network regression," in *Proceedings of the 7th Asia-Pacific Workshop on Structural Health Monitoring, APWSHM 2018*, pp. 757–766, NDT.net, Nov 2018.

[85] *Prognostics of damage growth in composite materials using machine learning techniques*, IEEE, Mar 2017.

[86] M. Corbetta, C. Sbarufatti, A. Saxena, K. Goebel, and M. Giglio, "Model-based fatigue prognosis of fiber-reinforced laminates exhibiting concurrent damage mechanisms," in *6th European Conference on Structural Control*, (Sheffield; United Kingdom), pp. 11–13, NASA Ames Research Center, 2016.

[87] S. Dabetwar, S. Ekwaro-Osire, and J. P. Dias, "Damage classification of composites using machine learning," in *ASME International Mechanical Engineering Congress and Exposition, Proceedings (IMECE)*, vol. 13, ASME, Nov 2019.

[88] A. Saxena, K. Goebel, C. C. Larrosa, and F.-K. Chang, "CFRP Composites Data Set."

[89] C. C. Larrosa, K. Lonkar, S. Shankar, and F. K. Chang, "Damage classification in composite laminates: Matrix micro-cracking and delamination," in *Structural Health Monitoring 2011: Condition-Based Maintenance and Intelligent Structures - Proceedings of the 8th International Workshop on Structural Health Monitoring*, vol. 1, pp. 191–199, DEStech Publications Inc., Sept 2011.

[90] K. Farinholt, M. Desrosiers, M. Kim, F. Friedersdorf, S. Adams, and P. Beling, "Active sensing and damage classification for wave energy converter structural composites," in *ASME 2016 Conference on Smart Materials, Adaptive Structures and Intelligent Systems, SMASIS 2016*, vol. 1, ASME, Sept 2016.

[91] W. S. Na and J. Baek, "A review of the piezoelectric electromechanical impedance based structural health monitoring technique for engineering structures," *Sensors (Switzerland)*, vol. 18, p. 1307, Apr 2018.

[92] D. Tibaduiza, M. Á. Torres-Arredondo, J. Vitola, M. Anaya, and F. Pozo, "A Damage Classification Approach for Structural Health Monitoring Using Machine Learning," *Complexity*, vol. 2018, pp. 1–14, Dec 2018.

[93] C. Doyle and G. Fernando, "Detecting impact damage in a composite material with an optical fibre vibration sensor system," *Smart Materials and Structures*, vol. 7, pp. 543–549, Aug 1998.

[94] P. M. Pawar and S. N. Jung, "Support vector machine based online composite helicopter rotor blade damage detection system," *Journal of Intelligent Material Systems and Structures*, vol. 19, pp. 1217–1228, Oct 2008.

[95] K. Worden and G. Manson, "The application of machine learning to structural health monitoring," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 365, pp. 515–537, Feb 2007.

[96] S. Gupta, S. Krishnan, and V. Sundaresan, "Structural health monitoring of composite structures via machine learning of mechanoluminescence," in *ASME 2019 Conference on Smart Materials, Adaptive Structures and Intelligent Systems, SMASIS 2019*, (Louisville, Kentucky, USA), ASME, Sept 2019.

[97] D. C. Seo and J. J. Lee, "Damage detection of CFRP laminates using electrical resistance measurement and neural network," *Composite Structures*, vol. 47, no. 1-4, pp. 525–530, 1999.

[98] Hexcel, "Hexcel HexTow AS4 Carbon Fiber Data Sheet," 2018.

[99] A. Todoroki, "The effect of number of electrodes and diagnostic tool for monitoring the delamination of CFRP laminates by changes in electrical resistance," *Composites Science and Technology*, vol. 61, pp. 1871–1880, Oct 2001.

[100] V. Bheemreddy, K. Chandrashekhara, L. R. Dharani, and G. E. Hilmas, "Modeling of fiber pull-out in continuous fiber reinforced ceramic composites using finite element method and artificial neural networks," *Computational Materials Science*, vol. 79, pp. 663–673, Nov 2013.

[101] A. P. Vassilopoulos and T. Keller, *Fatigue of Fiber-reinforced Composites*. Engineering Materials and Processes, London: Springer London, 2011.

[102] T. Loutas, N. Eleftheroglou, and D. Zarouchas, "A data-driven probabilistic framework towards the in-situ prognostics of fatigue life of composites based on acoustic emission data," *Composite Structures*, vol. 161, pp. 522–529, Feb 2017.

[103] F. Aymerich and M. Serra, "Prediction of Fatigue Strength of Composite Laminates by Means of Neural Networks," *Key Engineering Materials*, vol. 144, pp. 231–240, Sept 1998.

[104] H. El Kadi and Y. Al-Assaf, "Prediction of the fatigue life of unidirectional glass fiber/epoxy composite laminae using different neural network paradigms," *Composite Structures*, vol. 55, no. 2, pp. 239–246, 2002.

[105] H. El Kadi and Y. Al-Assaf, "Energy-based fatigue life prediction of fiberglass/epoxy composites using modular neural networks," *Composite Structures*, vol. 57, no. 1-4, pp. 85–89, 2002.

[106] J. A. Lee, D. P. Almond, and B. Harris, "Use of neural networks for the prediction of fatigue lives of composite materials," *Composites Part A: Applied Science and Manufacturing*, vol. 30, no. 10, pp. 1159–1169, 1999.

[107] H. A. El Kadi and Y. Al-Assaf, "The use of neural networks in the prediction of the fatigue life of different composite materials," in *ICCM International Conferences on Composite Materials*, 2007.

[108] M. Al-Assadi, H. A. Kadi, and I. M. Deiab, "Using artificial neural networks to predict the fatigue life of different composite materials including the stress ratio effect," *Applied Composite Materials*, vol. 18, pp. 297–309, Aug 2011.

[109] C. T. Sun and J. L. Chen, "A Simple Flow Rule for Characterizing Nonlinear Behavior of Fiber Composites," *Journal of Composite Materials*, vol. 23, pp. 1009–1020, Oct 1989.

[110] J. L. Chen and C. T. Sun, "A Plastic Potential Function Suitable for Anisotropic Fiber Composites," *Journal of Composite Materials*, vol. 27, pp. 1379–1390, Dec 1993.

[111] M. Xie and D. F. Adams, "A plasticity model for unidirectional composite materials and its applications in modeling composites testing," *Composites Science and Technology*, vol. 54, no. 1, pp. 11–21, 1995.

[112] S. W. Tsai and E. M. Wu, "A General Theory of Strength for Anisotropic Materials," *Journal of Composite Materials*, vol. 5, pp. 58–80, Jan 1971.

[113] P. Labossière and N. Turkkan, "Failure Prediction of Fibre-Reinforced Materials with Neural Networks," *Journal of Reinforced Plastics and Composites*, vol. 12, pp. 1270–1280, Dec 1993.

[114] C. S. Lee, W. Hwang, H. C. Park, and K. S. Han, "Failure of carbon/epoxy composite tubes under combined axial and torsional loading 1. Experimental results and prediction of biaxial strength by the use of neural networks," *Composites Science and Technology*, vol. 59, no. 12, pp. 1779–1788, 1999.

[115] P. A. Lopes, H. M. Gomes, and A. M. Awruch, "Reliability analysis of laminated composite structures using finite elements and neural networks," *Composite Structures*, vol. 92, no. 7, pp. 1603–1613, 2010.

[116] A. Şerban, "Failure estimation of the composite laminates using machine learning techniques," *Steel and Composite Structures*, vol. 25, pp. 663–670, Dec 2017.

[117] F. H. Bhuiyan, L. Kotthoff, and R. S. Fertig, "A machine learning technique to predict biaxial failure envelope of unidirectional composite lamina," in *33rd Technical Conference of the American Society for Composites 2018*, vol. 3, (Seattle, Washington, USA), pp. 1451–1463, Sept 2018.

[118] J. Li, *Magnetostrictive particle tagging of epoxy and epoxy composites for self assessment of damage*. Phd dissertation, University of Illinois at Urbana-Champaign, 2004.

[119] N. Nersessian, *Characterization and Improvement of Magnetostrictive Composites*. Phd dissertation, University of California, Los Angeles, 2005.

[120] J. L. Butler, "Application Manual for the Design of ETREMA Terfenol-D Magnetostrictive Transducers," tech. rep., ETREMA Products, Incorporated, 1988.

[121] Jiro, "GRABIT," 2016.

[122] W. Ramberg and W. R. Osgood, "Description of stress-strain curves by three parameters," tech. rep., National Advisory Committee for Aeronautics, 1943.

[123] E. Du Trémolet de Lacheisserie, D. Gignoux, and M. Schlenker, eds., *Magnetism: materials and applications*. Springer Science + Business Media, Inc., 2005.

[124] Scikit-Learn Developers, "3.1. Cross-validation: evaluating estimator performance," 2011.

[125] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, l. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, A. Vijaykumar, A. P. Bardelli, A. Rothberg, A. Hilboll, A. Kloeckner, A. Scopatz, A. Lee, A. Rokem, C. N. Woods, C. Fulton, C. Masson, C. Häggström, C. Fitzgerald, D. A. Nicholson, D. R. Hagen, D. V. Pasechnik, E. Olivetti, E. Martin, E. Wieser, F. Silva, F. Lenders, F. Wilhelm, G. Young, G. A. Price, G. L. Ingold, G. E. Allen, G. R. Lee, H. Audren, I. Probst, J. P. Dietrich, J. Silterra, J. T. Webber, J. Slavič, J. Nothman, J. Buchner, J. Kulick, J. L. Schönberger, J. V. de Miranda Cardoso, J. Reimer, J. Harrington, J. L. C. Rodríguez, J. Nunez-Iglesias, J. Kuczynski, K. Tritz, M. Thoma, M. Newville, M. Kümmerer, M. Bolingbroke, M. Tartre, M. Pak, N. J. Smith, N. Nowaczyk, N. Shebanov, O. Pavlyk, P. A. Brodtkorb, P. Lee, R. T. McGibbon, R. Feldbauer, S. Lewis, S. Tygier, S. Sievert, S. Vigna, S. Peterson, S. More, T. Pudlik, T. Oshima, T. J. Pingel, T. P. Robitaille, T. Spura, T. R. Jones, T. Cera, T. Leslie, T. Zito, T. Krauss, U. Upadhyay, Y. O. Halchenko, and Y. Vázquez-Baeza, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020.

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

[127] X. Chen and M. Anjanappa, "Health monitoring of composites embedded with magnetostrictive thick film without disassembly," *Smart Materials and Structures*, vol. 15, pp. 20–32, Feb 2006.

[128] P. D. Soden, M. J. Hinton, and A. S. Kaddour, "Lamina properties, lay-up configurations and loading conditions for a range of fibre reinforced composite laminates," in *Failure Criteria in Fibre-Reinforced-Polymer Composites* (M. Hinton, A. S. Kaddour, and P. D. Soden, eds.), ch. Chapter 2, pp. 30–51, Elsevier, 2004.

[129] M. Haavisto, *Residual Stresses in Rotating Steel Cylinders with Composite Surface Layers*. Master of science thesis, Aalto University, 2015.

[130] B. M. Luccioni, "Constitutive model for fiber-reinforced composite laminates," *Journal of Applied Mechanics, Transactions ASME*, vol. 73, pp. 901–910, Nov 2006.

[131] Hexcel Corporation, "Hexcel 3501-6 Epoxy Matrix," 2016.

[132] G. P. McCombe, J. A. Etches, I. P. Bond, and P. H. Mellor, "Magnetostrictive actuation of fiber-reinforced polymer composites," *Journal of Intelligent Material Systems and Structures*, vol. 20, pp. 1249–1257, Jul 2009.

[133] Hexcel Composites, "HexTow AS4 Carbon Fibre," 2013.

[134] R. de Medeiros, M. E. Moreno, F. D. Marques, and V. Tita, "Effective properties evaluation for smart composite materials," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 34, no. Special Issue, pp. 362–370, 2012.

[135] E. P. Sideridis and J. C. Venetis, "Thermal expansion coefficient of particulate composites defined by the particle contiguity," *International Journal of Microstructure and Materials Properties*, vol. 9, no. 3-5, pp. 292–313, 2014.

[136] W. Voigt, "Ueber die Beziehung zwischen den beiden Elasticitätsconstanten isotroper Körper," *Annalen der Physik*, vol. 274, no. 12, pp. 573–587, 1889.

[137] M. A. Loja, J. I. Barbosa, and C. M. Mota Soares, "A study on the modeling of sandwich functionally graded particulate composites," *Composite Structures*, vol. 94, pp. 2209–2217, Jun 2012.

[138] M. Kursa, K. Kowalczyk-Gajewska, M. J. Lewandowski, and H. Petryk, "Elastic-plastic properties of metal matrix composites: Validation of mean-field approaches," *European Journal of Mechanics, A/Solids*, vol. 68, pp. 53–66, Mar 2018.

[139] A. Reuss, "Berechnung der Fließgrenze von Mischkristallen auf Grund der Plastizitätsbedingung für Einkristalle .," *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 9, no. 1, pp. 49–58, 1929.

[140] R. Hill, "A theory of the yielding and plastic flow of anisotropic metals," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 193, pp. 281–297, May 1948.

[141] A. Wiśniewska, S. Hernik, A. Liber-Kneć, and H. Egner, "Effective properties of composite material based on total strain energy equivalence," *Composites Part B: Engineering*, vol. 166, pp. 213–220, Jun 2019.

[142] Z. Hashin and S. Shtrikman, "A variational approach to the theory of the elastic behaviour of polycrystals," *Journal of the Mechanics and Physics of Solids*, vol. 10, pp. 343–352, Oct 1962.

[143] T. Mori and K. Tanaka, "Average stress in matrix and average elastic energy of materials with misfitting inclusions," *Acta Metallurgica*, vol. 21, no. 5, pp. 571–574, 1973.

[144] L. J. Walpole, "The determination of the elastic field of an ellipsoidal inclusion in an anisotronic medium," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 81, no. 2, pp. 283–289, 1977.

[145] T. Mura and P. C. Cheng, "The elastic field outside an ellipsoidal inclusion," *Journal of Applied Mechanics, Transactions ASME*, vol. 44, pp. 591–594, Oct 1977.

[146] J. D. Eshelby, "Elastic Inclusions and Inhomogeneities," in *Collected Works of J. D. Eshelby* (I. N. Sneddon and R. Hill, eds.), ch. Chapter 3, pp. 297–350, Amsterdam; New York: North-Holland Publishing Company;Interscience Publishing, Inc., 2007.

[147] M. W. Hyer, "Some Observations on the Cured Shape of Thin Unsymmetric Laminates," *Journal of Composite Materials*, vol. 15, pp. 175–194, Mar 1981.

[148] K. P. Mohanchandra, S. V. Prikhodko, K. P. Wetzlar, W. Y. Sun, P. Nordeen, and G. P. Carman, "Sputter deposited Terfenol-D thin films for multiferroic applications," *AIP Advances*, vol. 5, p. 097119, Sept 2015.

[149] M. K. Panduranga, T. Lee, A. Chavez, S. V. Prikhodko, and G. P. Carman, "Polycrystalline Terfenol-D thin films grown at CMOS compatible temperature," *AIP Advances*, vol. 8, p. 056404, May 2018.

[150] J. Rudd, D. Spayde, and O. Myers, "Experimental nondestructive testing using magnetostrictive particles embedded in carbon fiber reinforced polymer beams," in *ASME 2012 Conference on Smart Materials, Adaptive Structures and Intelligent Systems, SMASIS 2012*, vol. 1, (Stone Mountain, Georgia, USA), pp. 707–711, American Society of Mechanical Engineers, Sept 2012.

[151] E. Sideridis, V. N. Kytopoulos, E. Kyriazi, and G. Bourkas, "Determination of thermal expansion coefficient of particulate composites by the use of a triphase model," *Composites Science and Technology*, vol. 65, pp. 909–919, May 2005.

[152] S. Torquato, T. M. Truskett, and P. G. Debenedetti, "Is random close packing of spheres well defined?," *Physical Review Letters*, vol. 84, no. 10, pp. 2064–2067, 2000.

[153] F. Dullien, *Porous Media*. Academic Press, 2nd ed., 1991.

[154] G. Arthur and J. A. Coulson, "Physical properties of uranium dioxide-stainless steel cermets," *Journal of Nuclear Materials*, vol. 13, pp. 242–253, Jan 1964.

[155] G. Currie, D. Spayde, and O. Myers, "Two tiered analysis of a CFRP laminate imbedded with magnetostrictive particles," in *ASME 2010 Conference on Smart Materials, Adaptive Structures and Intelligent Systems, SMASIS 2010*, vol. 2, (Philadelphia, Pennsylvania, USA), pp. 685–691, American Society of Mechanical Engineers, Sept 2010.

[156] N. P. Patel, E. Sarraf, and M. H. Tsai, "The Curse of Dimensionality: Whi High Dimensional Data Can Be so Troublesome," 2019.

[157] N. Sharma, "Understanding the Mathematics behind Principal Component Analysis," 2018.

[158] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Communications of the ACM*, vol. 18, pp. 509–517, Sept 1975.

[159] S. M. Omohundro, "Five balltree construction algorithms," Tech. Rep. 1, International Computer Science Institute, Berkeley, California, 1989.

[160] S. Sharma, S. Sharma, and A. Athaiya, "Activation Functions in Neural Networks," 2020.

[161] S. C. Douglas and J. Yu, "Why RELU Units Sometimes Die: Analysis of Single-Unit Error Backpropagation in Neural Networks," in *Conference Record - Asilomar Conference on Signals, Systems and Computers*, vol. 2018-Octob, (Pacific Grove, California), pp. 864–868, IEEE, Oct 2019.

[162] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," 2016.

[163] G. W. Brassell and K. B. Wischmann, "Mechanical and thermal expansion properties of a participate filled polymer," *Journal of Materials Science*, vol. 9, pp. 307–314, Feb 1974.

[164] J. L. Cribb, "Shrinkage and thermal expansion of a two phase material," *Nature*, vol. 220, pp. 576–577, Nov 1968.

[165] E. H. Kerner, "The elastic and Thermo-elastic properties of composite media," *Proceedings of the Physical Society. Section B*, vol. 69, pp. 808–813, Aug 1956.

[166] T. Wang and T. K. Kwei, "Effect of Induced Thermal Stresses on Coefficients of Thermal Expansion and Densities of Filled Polymers," *J Polymer Science-Polymer Physics*, vol. 7, pp. 889–896, May 1969.

[167] R. R. Tummala and A. L. Friedberg, "Thermal expansion of composite materials," *Journal of Applied Physics*, vol. 41, pp. 5104–5107, Dec 1970.

[168] A. A. Fahmy and A. N. Ragai, "Thermal-expansion behavior of two-phase solids," *Journal of Applied Physics*, vol. 41, pp. 5108–5111, Dec 1970.

[169] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science and Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[170] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference* (S. van der Walt and J. Millman, eds.), pp. 56–61, 2010.

[171] J. P. Bernhardt, "lamipy," 2017.