TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Maschinenwesen Institut für Festkörpermechanik

Professur für Numerische und Experimentelle Festkörpermechanik

Master Thesis

Analysis of the Failure Behaviour of Carbon Fibre Reinforced Polymer Open Hole Tensile Test Specimens

(Analyse des Versagensverhaltens gelochter Carbon-faserverstärkter polymerer Zugprobekörper)

Submitted by:	Apurv Kulkarni
Matrikulation Number:	4815274
Degree programme:	Computational Modelling and Simulation
Field of Study:	Computational Engineering
Tutor:	Dipl.-Ing. Benjamin Schmidt
First Reviewer:	Prof. Dr.-Ing. habil. Markus Kästner
Second Reviewer:	Dr.-Ing. Marreddy Ambati
Date of Submission:	4 May 2021

TECHNISCHE UNIVERSITÄT DRESDEN

School of Engineering Sciences Institute of Solid Mechanics

Chair of Computational and Experimental Solid Mechanics

Task description – Research Project

Course of study	Computational modelling and simulation
Track	Computational Engineering
Name	Apurv Kulkarni

Topic: **Analyse des Versagensverhaltens gelochter Carbon-faserverstärkter polymerer Zugprobekörper**

Analysis of the failure behaviour of carbon fibre reinforced polymer open hole tensile test specimens

Objective:

Due to their outstanding mechanical properties paired with a high specific strength, carbon fibre-reinforced polymers (CFRP) became one of the most important materials in lightweight engineering. Composites like CFRP have a complex failure and fracture behaviour, since there often occurs a superposition of multiple damage phenomena. The wide range of application areas and hence the manifold of complex component geometries is also a challenge for the analysis due to the strong influence of notches and holes on the failure behaviour.
The aim of this work is the numerical analysis of the correlation between notch geometry and fracture behaviour of CFRP structures on the example tensile tests of specimens with elliptical holes with varying eccentricity. In a sensitivity analysis the influence of the hole geometry is compared to that of other geometry and material parameters.

Tasks:

- Literature review on the fracture behaviour of composites and its modelling
- Introduction to data analysis
- Introduction to the simulation software LS-Dyna
- Development of a data-analysis environment in python
- Analysis of the fracture behaviour and identification of parameters to characterise the failure of composites
- Discretisation of elliptical open hole tensile test specimens with varying eccentricity
- Sensitivity analysis to identify the influence of the kerf geometry and other geometry and material parameters on failure behaviour
- Thorough documentation of the achieved results

Tutor:	Benjamin Schmidt
First Reviewer:	Prof. Dr.-Ing. habil. Markus Kästner
Second Reviewer:	Dr.-Ing. Marreddy Ambati
Start:	01.12.2020
End:	04.05.2021

Acknowledgements

First and foremost, I am extremely grateful to my supervisor Mr Benjamin Schmidt for his invaluable advice, continuous support, and patience during my master's thesis. His immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. I would also like to thank Prof. Markus Kästner for providing me with an opportunity to work in the chair of Computational and Experimental Solid Mechanics. I would like to thank all the other professors, teaching staff, the members of the chair and TU Dresden for their direct and indirect support. It is their kind help and support that have made my study and life in Germany a wonderful time. Finally, I would like to express my gratitude to my parents and my friends. Without their tremendous understanding and encouragement in the past, it would be impossible for me to complete my study.

Abstract

Carbon fibre-reinforced polymers have become one of the most important material in lightweight engineering, due to their outstanding mechanical properties combined with high specific strength. Composites like Carbon Fibre Reinforced Polymer have complex failure and fracture phenomena. The notches and holes present in composite materials in structural components also have strong influence on the failure behavior. In this work, numerical analysis of the influence of material parameters on failure and fracture behavior of composite is studied. In the later part of the study, the influence of notch geometry on composites is presented. The experimental setup consist of CFRP open hole tensile specimen subjected to uniaxial tensile load. The computer experiments performed in this work, employ the methodology of Design of Experiments and sensitivity analysis, for setting up experiments and analysing the outcomes, respectively.

Contents

List of Figures IV

List of Tables V

List of Symbols VI

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Scope . . . 3

2 Theory of Composites 4
- 2.1 Constituent materials . . . 4
  - 2.1.1 Reinforcement materials . . . 4
  - 2.1.2 Matrix materials . . . 6
  - 2.1.3 Carbon fibre reinforced composites . . . 6
- 2.2 Mechanics of Composite Materials . . . 8
  - 2.2.1 Directional Material Behavior . . . 10
  - 2.2.2 Macromechanics of Composite Lamina . . . 12
  - 2.2.3 Micromechanics of Composite Lamina . . . 16

3 Failure in Composites 17
- 3.1 Damages in Composite Material . . . 17
- 3.2 Failure Models . . . 19
  - 3.2.1 Composite Material Modelling . . . 21
  - 3.2.2 Delamination Modelling . . . 24

4 Design of Composite Structures 26

5 Design of Experiments and Sensitivity Analysis 28
- 5.1 Design of Experiments . . . 28
- 5.2 Sensitivity Analysis . . . 31

6 Numerical Experiments 35
- 6.1 Workflow . . . 35
- 6.2 FEM Modelling . . . 37
- 6.3 Factor Selection . . . 38
- 6.4 Results and Discussion . . . 40

7 Conclusion **48**

References **50**

A Appendix **57**

A.1 Material Cards . . . 57
A.1.1 Composite Material Card . . . 57
A.1.2 Cohesive Element Material Card . . . 58
A.2 Experiment Parameters . . . 59
A.2.1 Design Variables . . . 59
A.2.2 Response Variables. . . . 60
A.3 Result Plots . . . 60
A.3.1 Bivariate Regression . . . 60
A.3.2 Multiple Regression . . . 66
A.4 Code Repository . . . 71
A.4.1 Step-0: Initial Setup and Input Variables . . . 71
A.4.2 Step-1: Database and LS-Dyna Card Creation . . . 76
A.4.3 Step-2: Creating Sbatch scripts for HPC Job Submission . . . 80
A.4.4 Step-3: Creating LS-PrePost Macros for Data Extraction . . . 83
A.4.5 Step-4: Data Extraction and Cleaning . . . 86
A.4.6 Step-5: Data Analysis . . . 89
A.4.7 Project Utilities . . . 94
A.4.8 LS-Prepost Macro Utilities . . . 105
A.4.9 Mesh Utilities . . . 112
A.4.10 LS-Prepost Sample Macro . . . 124

List of Figures

1.1 Strength versus mass density for various engineering materials as per Ashby (2005)[6]. . . . 2
1.2 OHT (a) Experimental setup [47] (b) FEM model. . . . 2

2.1 Constituent phases of composite material . . . 4
2.2 Classification of composites based on types of fibres. . . . 5
2.3 (a) CFRP Composite material [35] (b) CFRP structure.[10] . . . 7
2.4 Classification of carbon fibres based on mechanical properties.[31] . . . 7
2.5 Types of fabrication methods used in composite structure manufacturing.[56] 8
2.6 Mechanical properties of CFRP and other composites.[40] . . . 8
2.7 Different scales used in analysis of a composite material(adapted from [12]). 9
2.8 Ply configuration and their notation . . . 10
2.9 Material coordinate system in a ply. . . . 10
2.10 Planes of material symmetry in orthotropic material . . . 11
2.11 Materials under loading conditions (adapted from [12]). . . . 11
2.12 Stress at a point in continuum. . . . 13
2.13 Longitudinal tension. . . . 14
2.14 Transverse tension and compression. . . . 15
2.15 In-plane shear. . . . 15
2.16 Representation of RVE in a composite structure[48] . . . 16

3.1 Fibre-Matrix Debonding: (a) SEM image[71] (b) UD lamina under longitudinal tensile load. . . . 17
3.2 Fibre-Matrix Cracking: (a) SEM micrograph [17] (b) Laminate under longitudinal tensile load. . . . 18
3.3 Fibre fracture: (a) SEM micrograph of impact loading on composites[13] (b) SEM micrograph of bend test on composites[7] (c) Kinking[24]. . . . 18
3.4 Delamination: (a) Flexural failure of CFRP composite[79] (b) FEM analysis of composite under tensile loading. . . . 19
3.5 Various damages occurring in composite under longitudinal tension. . . . 20
3.6 (a) Longitudinal tensile failure (fibre failure mode) (b) Longitudinal compressive fracture (Compressive fibre mode) (c) Transverse fracture (Tensile matrix mode) (d) Transverse fracture (Compressive matrix mode). . . . 21
3.7 Definition of angles and stress for kinking in 3D space[52]. . . . 23
3.8 Traction components on fracture plane at angle ϕ[52]. . . . 23
3.9 (a) Non-linear in-plane shear behavior (b) Damage evolution law [37] . . . 24

3.10 (a) Decohesion element (b) Traction separation law for various modes [37] . . 25

5.1 General schematic of an experiment. 28
5.2 Design space in LHD: (a) low correlation (b) high correlation, between design variable sample points. 30
5.3 Different cases of: (a) PCC values (b) SRCC values. 33

6.1 Project workflow. 36
6.2 Specimen dimensions. 37
6.3 Different layers in the specimen. 37
6.4 Material axes. 38
6.5 Different mesh configurations used in convergence study. 38
6.6 (a) Stress concentration factor at the notch (b) Applied stress, for a 0° ply. . 39
6.7 Design variables in the specimen. 39
6.8 PCC heatmap of input design variables. 40
6.9 Heatmap plot of (a) PCC (b) SRCC, for input vs response variables. 41
6.10 Regression plot: XT vs response variables. 42
6.11 Regression plot: YT vs response variables. 43
6.12 Regression plot: (a) GI vs response variables (b) GII vs response variables . . 43
6.13 Regression plot: MANG vs response variables. 44
6.14 Regression plot: NOTCHR vs response variables. 45
6.15 Multiple linear regression coefficients. 46

A.1 Composite material card. 57
A.2 Cohesive element card. 58
A.3 Full heatmap plot of PCC values of design variables and response variables. . 61
A.4 Full heatmap plot of SRCC values of design variables and response variables. 61
A.5 Regression plots of design variables vs response variables. 62
A.6 Regression plots of design variables vs response variables. 63
A.7 Regression plots of design variables vs response variables. 64
A.8 Regression plots of design variables vs response variables. 65

List of Tables

2.1 Comparison between various properties of matrix used in composites[50]. . . . 6

5.1 A design matrix. 29

A.1 Composite material card. 58
A.2 Cohesive material card. 59
A.3 Design variables 60
A.4 Response variables. 60

List of Symbols

Mathematical Symbols	Meaning
θ	Angle, Fibre kinking angle
e	Unit vector, experiment run
$\boldsymbol{\varepsilon}$	Strain tensor
ε	Normal/Transverse strain
γ	Shear strain
$\boldsymbol{\sigma}$	Stress tensor
σ	Normal/Transverses stress
$\hat{\sigma}$	Degraded stress matrix
τ	Shear Stress
η	Shear coupling coefficient
E	Young's modulus
G	Shear modulus
ν	Poisson's ratio
$\mathbf{C}$	Elastic (Stiffness) material tensor
C_{ij}	Stiffness matrix coefficients
$\mathbf{S}$	Compliance material tensor
S_{ij}	Compliance matrix coefficients
$\mathbf{Q}$	Reduce stiffness matrix
Q_{ij}	Reduced stiffness matrix coefficients
X_{t}, X_{c}	Longitudinal tensile and compressive strength
Y_{t}, Y_{c}	Transverse tensile and compressive strength
S_{L}, S_{T}	Longitudinal and Transverse in-plane shear strength
ρ	Pearson's correlation coefficient
μ	Friction coefficient
ϕ	Fracture angle
Ψ	Fibre kinking plane angle
Γ	Fracture toughness
L	Characteristic length
δ	Relative displacement
I, II and III	Opening, In-plane and mixed shear mode
T, S	Peak traction in normal and tangential direction
XMU	Mixed-mode parameter
X	Design variables

Mathematical Symbols	Meaning
x	Design point
$\sum$	Summation
r_s	Spearman's ranked correlation coefficient
Subscripts:	
x, y, z	Global coordinate system
1, 2, 3	Material coordinate system in longitudinal, transverse and through-the thickness direction, respectively
a, b, c	Material coordinate system in longitudinal, transverse and through-the thickness direction, respectively
i, j	Row and column of a matrix
f	Fibre material
m	Matrix material
n	Normal direction, total number of design variables
p	total number of experimental runs
t, c	Tension and compression respectively
Superscripts:	
f,F	Failure point
0	Damage initiation point

Abbreviations	Meaning
API	Application Package Interface
CDM	Continuum Damage Mechanics
CFRP	Carbon Fibre Reinforced Polymer
CLT	Classical Laminate Theory
CMC	Ceramic Matrix Composites
DOE	Design of Experiments
FAST	Fourier Amplitude Sensitivity Test
FEA/M	Finite Element Analysis/Method
FRP	Fibre Reinforced Plastic
GFRP	Graphite Fibre Reinforced Polymer
HP-Grade	High Performance grade
HPC	High Performance Computing
LHD	Latin Hypercube Design
MMC	Metal Matrix Composites

Abbreviations	Meaning
MOAT	Morris-One-at-a-time
OFAT	One Factor At a Time
OHT	Open Hole Tension
PAN	Polyacrylonitril
PCC	Pearson's Correlation Coefficient
PMC	Polymer Matrix Composites
RVE	Representative Volume Element
RRIM	Reinforced Reaction Injection Molding
SA	Sensitivity Analysis
SEM	Scanning Electron Microscope
SRCC	Spearman's Ranked Correlation Coefficient
TSLC	Traction Separation Law Curve
UD	Unidirectional lamina
VCC	Virtual crack closure

1 Introduction

1.1 Background

Fibre-reinforced plastic (FRP) composites are known to have higher strength per unit weight than conventional materials like metals. It was first demonstrated by Griffith [22] that many materials like glass are much stronger and stiffer in fibre form when compared to their bulk form. There are a wide variety of fibres available for use of composites. The first fibres were synthetically made from glass in the 1930s. In later years fibres with high specific stiffness and specific strength were developed. They are called advanced fibres and composites made from them are called advanced composites or high-performance composites. Boron, Carbon, Silicon carbide, Aramid (Kevlar) and sapphire are some examples of materials used in such composites. Glass fibres because of their comparatively low specific strength and specific stiffness not considered advanced fibres. Among many available advanced fibres, carbon fibre has gained popularity as the choice of reinforcement in various high-performance applications like aerospace structures. Carbon fibres typically have a carbon content between 80%-95%. Carbon fibre reinforcement polymer (CFRP) composites are formed by embedding these carbon fibres in a matrix. Matrix materials like thermosetting epoxy resin polymer provide a cohesive strength to protect and hold the fibres together and provide some toughness, while carbon fibres mainly contribute to the strength and stiffness of the CFRP composites.

CFRP composites have excellent structural properties, like high stiffness and high strength to density ratio (Figure 1.1), low thermal expansion coefficient and good vibration damping characteristics when compared with other materials. Due to this these composites are widely used in structural components. These are used to create aerospace components like wings, fuselage, rotor blades, airframes etc. CFRP composites are used in constructing monocoque chassis of high-end sports car components and Formula 1 racing cars. The use of blades made from CFRP composites, in General Electric's (GE Aviation) GE9X jet engines helped in creating efficient and lighter engines allowing airlines to save fuels [72]. CFRP composites are also used in sports equipment like tennis, badminton racquets, hockey sticks, fishing rods and bicycle frames.

Load-bearing structures are often constructed from composites (like CFRP) using joints like welding, adhesive joints and mechanical fastening. Using joints instead of uni-body helps easy replacement of parts, good accessibility to components, facilitating efficient maintenance. Among all kinds of joints, mechanically fastened joints are frequently used because of their lower cost and easy assembly. Such joints result in open holes and cuts which acts as the sources of stress concentrations causing premature failure of the composite. It is important to understand the underlying failure processes occurring in composite to improve

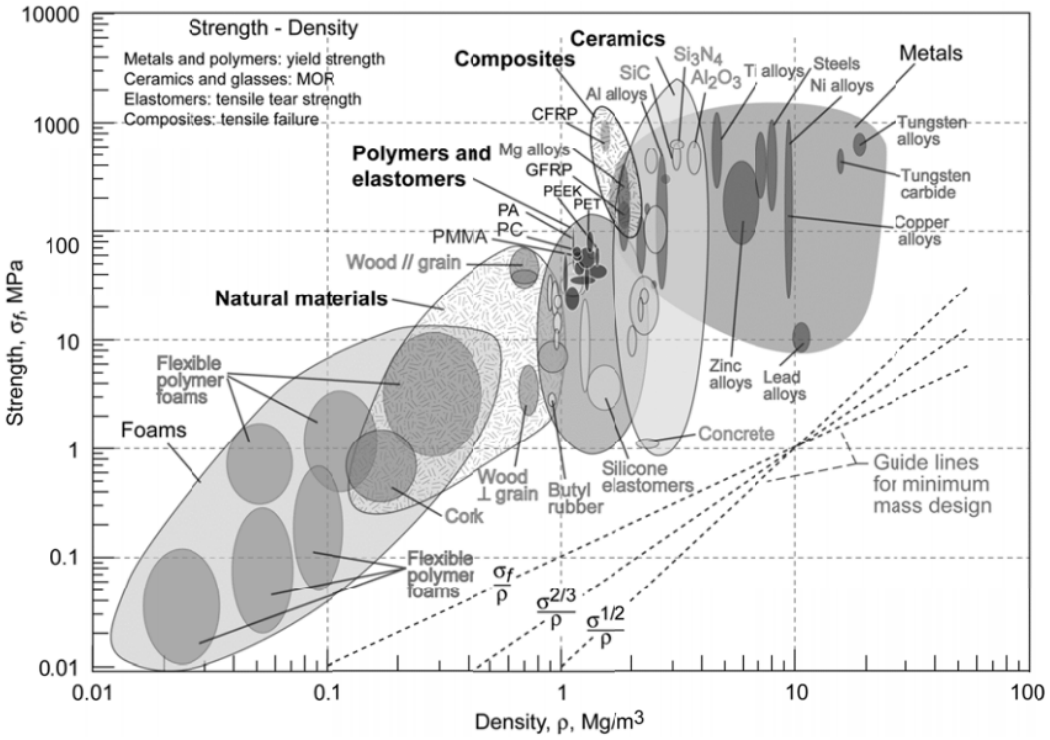


Figure 1.1: Strength versus mass density for various engineering materials as per Ashby (2005)[6].

their performance and reliability. In the recent decade, several studies have been conducted to understand the complex phenomenon of failure in composite structures, especially in the case of open-hole composite specimens of finite width under uniaxial tension [23].

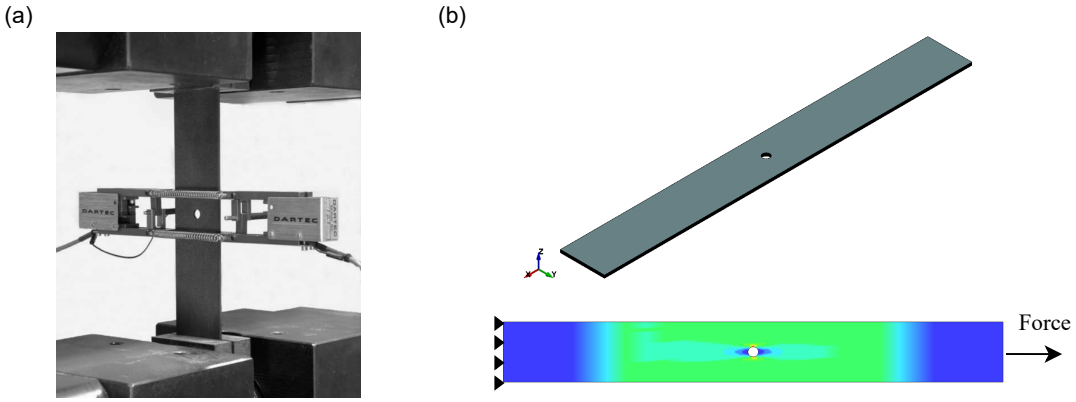


Figure 1.2: OHT (a) Experimental setup [47] (b) FEM model.

Modelling and predicting the response of composite laminates accurately in an open hole tensile/tension (OHT) test (Figure 1.2) is still an area which many researchers find challenging. During OHT test there are several damage mechanisms taking place like matrix cracking, fibre-matrix debonding, delamination and fibre breakage [9, 23, 77]. There have

been several failure models and theories that have been developed. This work uses one of such material model proposed by Pinho et al.[51, 52] to asses various parameters in an OHT testing.

1.2 Scope

The objective of this work is to study various material parameters used in Pinho [51, 52] material model implemented in an explicit FE code in LS-Dyna [36]. This material model distinguishes between matrix failure, fibre kinking and fibre tensile failure. In addition, effect of different elliptical notch geometries is also studied for OHT specimen under tensile load. The computer experiments are created using Design of Experiments (DOE) methodology and later are studied using sensitivity analysis measures. The main goals are to

- understand basics of failure behavior of CFRP composites,
- create DOE workflow and tools in Python [54],
- create simulation experiments using different material parameters and notch geometry using concepts of DOE and perform analysis in LS-Dyna software suit,
- collect output results of the simulation experiment and perform sensitivity analysis using Python and its libraries,
- assess the results and draw conclusion based on sensitivity analysis.

The report is organized into the following seven chapters. In Chapter 2 various aspects of composite materials like constituent materials and mechanics are discussed. Chapter 3 begins with the discussion on failure mechanisms in the composite. The material model used in the study is also described in this chapter. Chapter 4 presents various findings from the literature concerning the study. Chapter 5 throws light upon the DOE methodology and sensitivity analysis in brief. The main work of this study is presented in Chapter 6. Section 6.1, Section 6.2 and Section 6.3 explains the experimental setup, while the results are presented in Section 6.4. The report ends with the Chapter 7 concluding outcomes of the study.

2 Theory of Composites

In this chapter, basic theory about various aspects of composite materials is presented. This would help in understanding the work better. Section 2.1 covers the composition of composite materials. Section 2.2 presents the mechanics of composite.

2.1 Constituent materials

Composites are heterogeneous material systems formed by the combination of two or more materials phases. One of the material phases is called reinforcement, while the other is called matrix, with the former being stronger and stiffer than the latter. The reinforcement material is present discontinuously throughout the material and is embedded in the continuous matrix material as shown in Figure 2.1. These constituent materials with different physical and material properties remain in a separate and distinctive state without losing their individual characteristics. The finished composite material has improved mechanical performance and properties to those of constituent materials. The overall property of a composite depends upon properties, distribution and volume fraction of these constituent phases, and geometry of the composite structure itself. Various reinforcement and matrix materials are discussed in following subsections.

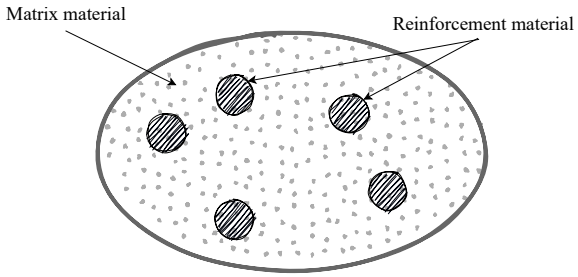


Figure 2.1: Constituent phases of composite material

2.1.1 Reinforcement materials

The superior performance of composites lies in the high specific strength and high specific modulus controlled by the type of reinforcement material used in it. The composite material comprises reinforcement materials in the form of fibre, particulate or both mixed, depending upon application and requirements. Fibre reinforcement materials, characterized by a high length to cross-sectional ratio, are used for most of the high performance and high strength applications. Particle reinforcement materials are used in construction material like concrete, sand and gravel. Reinforcement materials can be further classified based on their type, layup and size as shown in Figure 2.2. More information about this can be found in [34]. In general, composites with continuous fibres have better load transfer capability than those composites

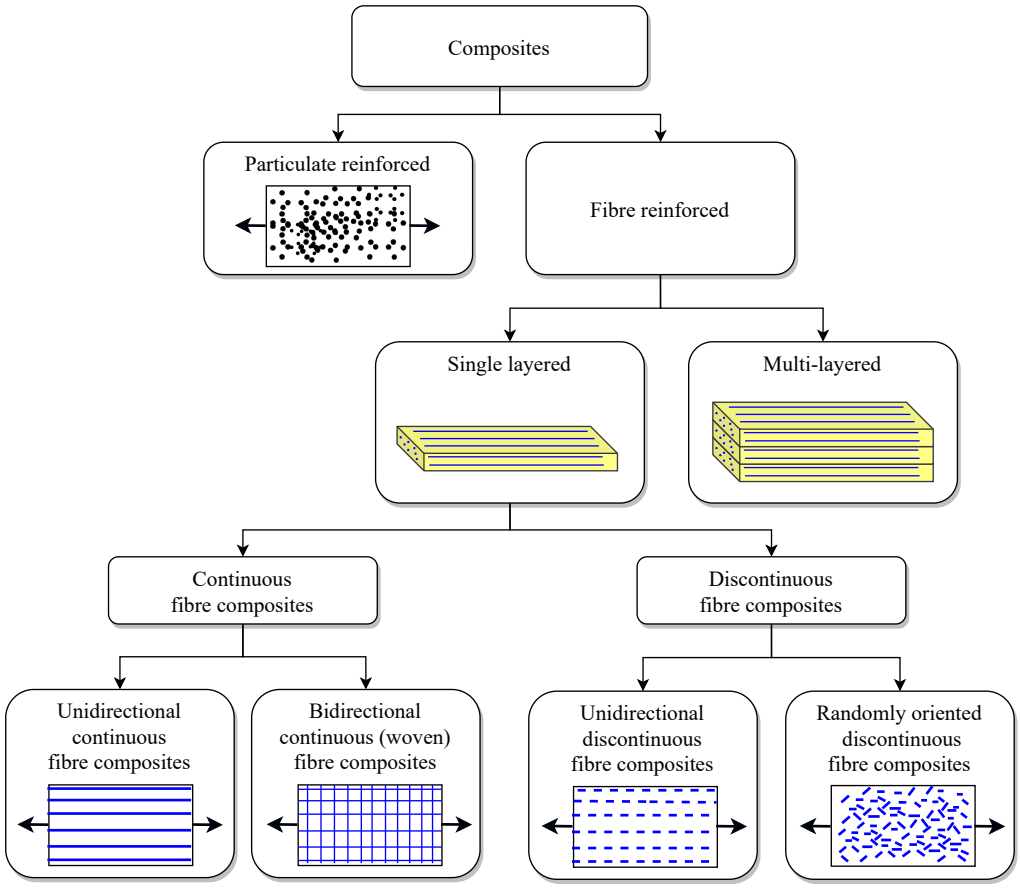


Figure 2.2: Classification of composites based on types of fibres.

with short or discontinuous fibres. Fibres can be made from materials like Glass, Carbon, Aramid and Boron. These fibres are described as follows,

- **Glass fibres**
  These fibres consists of tetrahedral units of $(SiO_4)_n$ as a structural backbone[34]. They offer good strength at a relatively low cost but have poor abrasion resistance and adhesion to the matrix. Glass fibres are most widely used fibres in the industry.

- **Carbon/Graphite fibres**
  Manufactured from polyacrylonitrile polymer fibre (PAN) and other materials like rayon and petroleum pitch, carbon/graphite fibres Offers high strength and stiffness. These are the most widely used advanced fibre in aerospace industries and military applications where high performance is required.

- **Aramid fibres**
  Also known as Kevlar, aramid fibres are manufactured from aromatic polyamides. It offers high tensile strength and modulus, resistance to chemicals and good temperature stability.

- **Boron fibres**
  These fibres are made from chemical vapor decomposition of a boron tri-chloride gas. They offer low density high tensile strength and have relatively high fabrication cost.

2.1.2 Matrix materials

Matrix material in composites is used to bind the fibres together. It helps in transmitting loads to the fibres. These have low modulus and strength as compared to fibre materials. Depending upon the type of matrix, composites are again classified as Polymer Matrix Composites (PMCs), Metal Matrix Composites (MMCs) and Ceramic Matrix Composites (CMCs). Comparison between properties of these matrices is presented in Table 2.1. Composites with polymer as a matrix are most extensively used in the industry in high-performance applications, while other matrices are generally used for high-temperature applications. Polymer matrices are classified into Thermoset and Thermoplastic polymers. Composites with thermoset resins as the matrix material are made by adding curing agent, drenching reinforced materials followed by curing process. Epoxies, polyamides, unsaturated polyesters are some of the commonly used thermoset polymers. Among these, Epoxy resins are the most predominantly used thermoset polymers exhibiting low molecular weight, good mechanical and thermal properties. Unlike thermoset resins, thermoplastic resins can be reformed and reshaped by reheating them. Polycarbonate, nylon, polystyrene are some of the examples of thermoplastic resins.

Property	Ceramic Matrix	Metal matrix	Polymer Matrix
Hardness	High	Low	Low
Elastic modulus	High	High	Low
High temperature strength	High	Low	Low
Thermal expansion	Low	High	High
Ductility	Low	High	High
Corrosion resistance	High	Low	Low
Resistance to wear	High	Low	Low
Electrical conductivity	High and Low	High	Low
Density	Low	High	Low
Thermal conductivity	High and Low	High	Low

Table 2.1: Comparison between various properties of matrix used in composites[50].

2.1.3 Carbon fibre reinforced composites

CFRP composite materials are composed of carbon fibres and polymer as the matrix material. Figure 2.3 shows the general structure of CFRP.

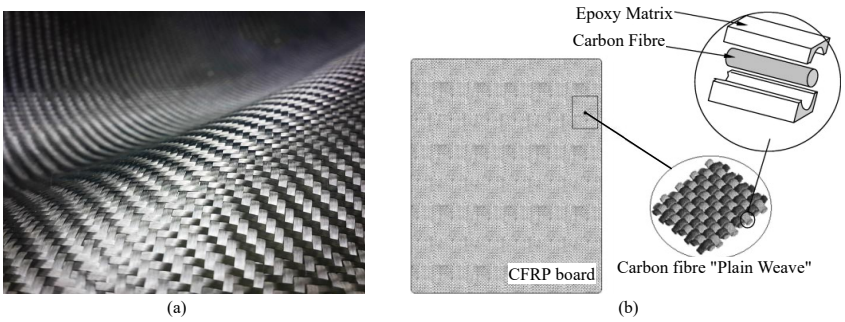


Figure 2.3: (a) CFRP Composite material [35] (b) CFRP structure.[10]

Carbon fibres have over 90% carbon contents by weight and are fabricated from a series of operation of heating and tension on pitch or polyacrylonitrile (PAN) materials. PAN type CFRPs have better mechanical properties than pitch type CFRPs but are costlier than pitch type CFRPs[18]. Carbon fibres present in commercially available CFRPs can be categorized based on their mechanical properties. Figure 2.4 shows various types of carbon fibres used in the industries.

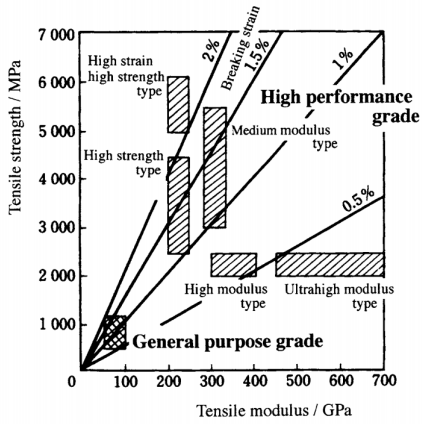


Figure 2.4: Classification of carbon fibres based on mechanical properties.[31]

High-performance grade (HP-Grade) carbon fibres can have higher strength and modulus when compared to traditional materials like steel with the advantage of providing CFRP material with lower density. Epoxy resin is usually used as matrix materials in CFRPs. CFRPs can be fabricated by various moulding methods shown in Figure 2.5.

As discussed previously, CFRP offers excellent mechanical (high specific strength and modulus) and chemical properties (abrasion resistance, high working temperature) when compared with conventional materials. Mechanical properties of carbon fibre composites along with other composites are shown in Figure 2.6.

Having discussed the composition of Composite materials and CFRP, the next section presents theory on mechanics of composite in brief.

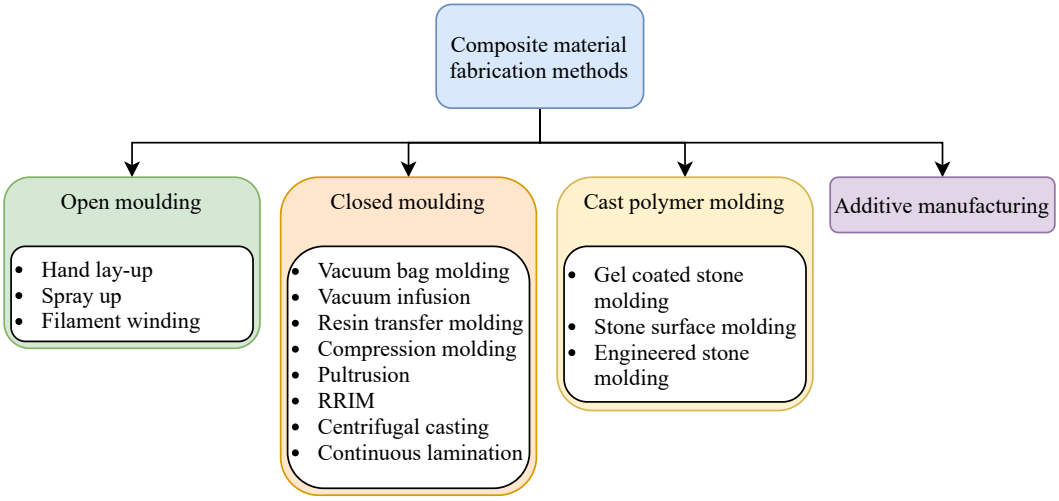


Figure 2.5: Types of fabrication methods used in composite structure manufacturing.[56]

Mechanical Properties of Carbon Fibre Composite Materials, Fibre / Epoxy resin (120°C Cure)

Fibres @ 0° (UD), 0/90° (fabric) to loading axis, Dry, Room Temperature, Vf = 60% (UD), 50% (fabric)

	Symbol	Units	Std CF Fabric	HMCF Fabric	E glass Fabric	Kevlar Fabric	Std CF UD	HMCF UD	M55** UD	E glass UD	Kevlar UD	Boron UD	Steel S97	Al. L65	Tit. dtd 5173
Young's Modulus 0°	E1	GPa	70	85	25	30	135	175	300	40	75	200	207	72	110
Young's Modulus 90°	E2	GPa	70	85	25	30	10	8	12	8	6	15	207	72	110
In-plane Shear Modulus	G12	GPa	5	5	4	5	5	5	5	4	2	5	80	25	
Major Poisson's Ratio	v12		0.10	0.10	0.20	0.20	0.30	0.30	0.30	0.25	0.34	0.23			
Ult. Tensile Strength 0°	Xt	MPa	600	350	440	480	1500	1000	1600	1000	1300	1400	990	460	
Ult. Comp. Strength 0°	Xc	MPa	570	150	425	190	1200	850	1300	600	280	2800			
Ult. Tensile Strength 90°	Yt	MPa	600	350	440	480	50	40	50	30	30	90			
Ult. Comp. Strength 90°	Yc	MPa	570	150	425	190	250	200	250	110	140	280			
Ult. In-plane Shear Stren.	S	MPa	90	35	40	50	70	60	75	40	60	140			
Ult. Tensile Strain 0°	ext	%	0.85	0.40	1.75	1.60	1.05	0.55		2.50	1.70	0.70			
Ult. Comp. Strain 0°	exc	%	0.80	0.15	1.70	0.60	0.85	0.45		1.50	0.35	1.40			
Ult. Tensile Strain 90°	eyt	%	0.85	0.40	1.75	1.60	0.50	0.50		0.35	0.50	0.60			
Ult. Comp. Strain 90°	eyc	%	0.80	0.15	1.70	0.60	2.50	2.50		1.35	2.30	1.85			
Ult. In-plane shear strain	es	%	1.80	0.70	1.00	1.00	1.40	1.20		1.00	3.00	2.80			
Thermal Exp. Co-ef. 0°	Alpha1	Strain/K	2.10	1.10	11.60	7.40	-0.30	-0.30	-0.30	6.00	4.00	18.00			
Thermal Exp. Co-ef. 90°	Alpha2	Strain/K	2.10	1.10	11.60	7.40	28.00	25.00	28.00	35.00	40.00	40.00			
Moisture Exp. Co-ef 0°	Beta1	Strain/K	0.03	0.03	0.07	0.07	0.01	0.01		0.01	0.04	0.01			
Moisture Exp. Co-ef 90°	Beta2	Strain/K	0.03	0.03	0.07	0.07	0.30	0.30		0.30	0.30	0.30			
Density		g/cc	1.60	1.60	1.90	1.40	1.60	1.60	1.65	1.90	1.40	2.00			

*** Calculated figures*

Figure 2.6: Mechanical properties of CFRP and other composites.[40]

2.2 Mechanics of Composite Materials

Analysis of composite material is different than that of of conventional materials like metal or rubber as their composition is unique. It needs to be understood on different levels as shown in Figure 2.7.

When composite materials are looked at the most basic level, it is a mixture of matrix and fibres/particulate. In the micromechanical analysis the interactions between these constituent materials are studied at the microscopic level. At this level, the material is

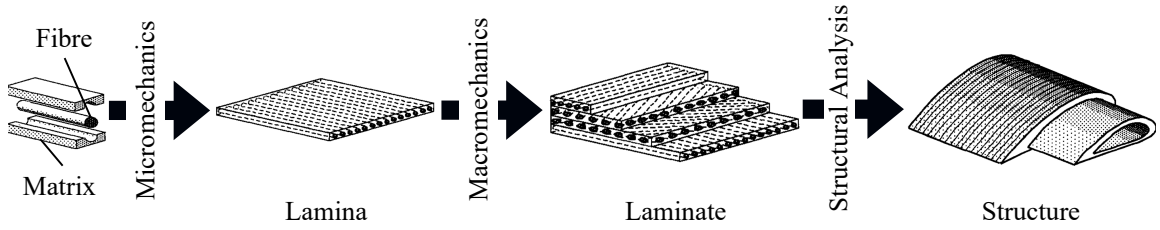


Figure 2.7: Different scales used in analysis of a composite material(adapted from [12]).

nonhomogeneous. Because of this, there is large variation in material properties. Analysis at this level helps to understand different failure mechanisms in fibres and matrix, different material properties like fracture toughness, fatigue life, strength, state of deformation and stress in constituent materials. Because of computational limitation, it is very difficult to compute the response of large structures using micromechanical analysis. This further forms the basis for macromechanical analysis, as it allows for average behavior at this level as a function of constituent properties and local conditions. In macromechanical analysis or Macromechanics, the analysis is applied on individual lamina. A lamina (or ply or layer), is a layer (curved or plane) of composite comprising unidirectional or woven fibres embedded in matrix. When unidirectional fibres are used, the lamina is also called as unidirectional (UD) lamina. Analysis at this level assumes that the material is homogeneous and that the properties of lamina are the averaged properties of its constituents. Failure criteria at this level of analysis can be expressed in terms of average stresses and overall lamina strengths.

This analysis is further extended to laminates. A laminate is a stack of two or more UD plies glued together. The plies can be present in the same or different orientations and can be constructed using the same or different materials. Orientation of a ply is given as the angle between the reference axis (usually x-axis) of a fixed coordinates system and the fibre orientation/warp direction, measured in the counterclockwise direction. Composite laminates are designated using orientation, number, type and stacking sequence of plies. Figure 2.8 shows naming of laminates based on their configuration. The subscript 's' denotes symmetry.

The macromechanics of whole laminate can be developed from macromechanics of single lamina by defining the overall behaviour and property as the function of individual lamina properties. One of the most widely used theories known as Classical Laminate Theory (CLT), predicts laminate behaviour as a function of properties, thickness, orientation and arrangements of the individual lamina.

Eventually the analysis is performed at the component level where components or structures are made from composites. At this level behaviour of component/structure is predicted

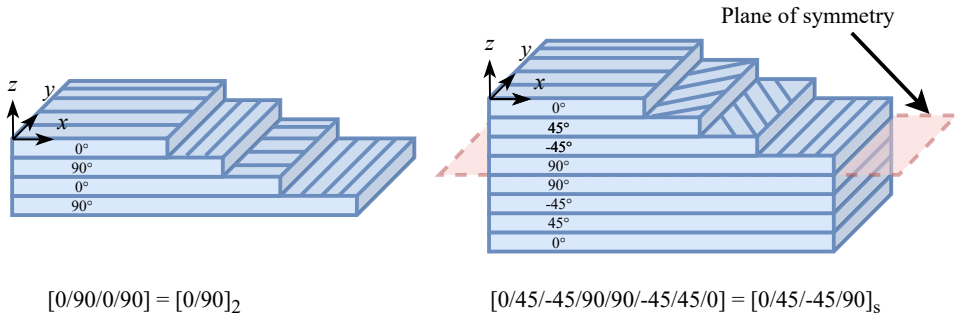


Figure 2.8: Ply configuration and their notation

in terms of performance (stress, strain), dynamics stability and failure using numerical methods like Finite Element Analysis (FEA).

2.2.1 Directional Material Behavior

Many material properties like stiffness, strength, thermal expansion, conductivity and permeability depend upon material direction. Based on this characteristic, materials can be categorized into isotropic and anisotropic materials. All the properties of isotropic materials are same in all directions. Anisotropic materials have different material properties in all directions. The material property, in this case, is dependent on material orientation with respect to the principle material coordinate system as shown in Figure 2.9.

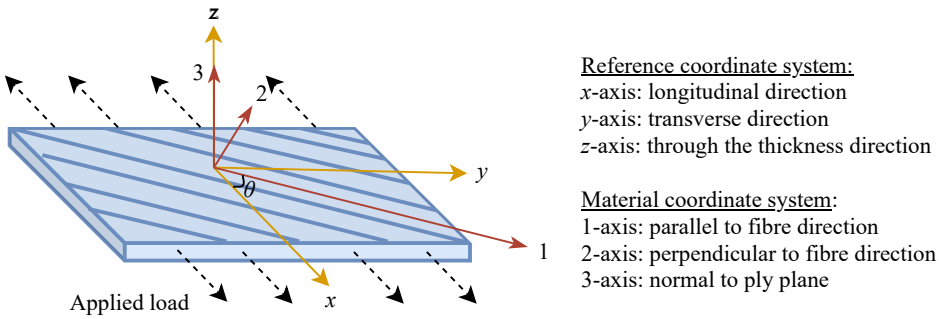


Figure 2.9: Material coordinate system in a ply.

In the figure, material coordinates system $1-2-3$ is defined with respect to reference coordinate system $x-y-z$ where θ is the angle between the $x-$axis and $1-$axis denoting material orientation. A material may have a plane of symmetry across which, in the direction of the symmetry plane, material properties are the same. A material might have zero to an infinite number of symmetry planes. Isotropic materials have infinite planes of material symmetry, while anisotropic materials have none. Orthotropic materials are a subclass of anisotropic materials with three mutually perpendicular planes of material symmetry as

shown in Figure 2.10. Hence they have unique material properties in these three directions (e_1, e_2, e_3).

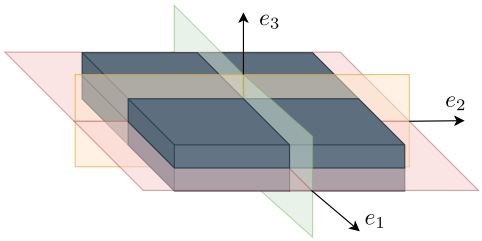


Figure 2.10: Planes of material symmetry in orthotropic material

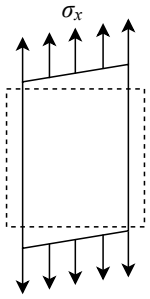
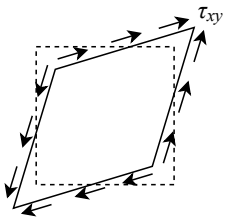
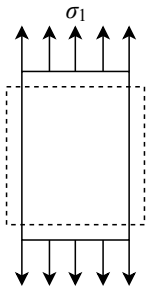
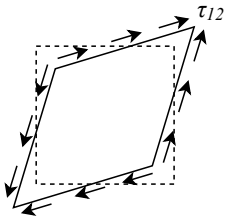
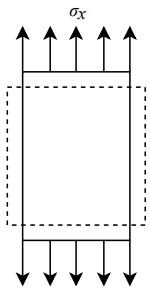
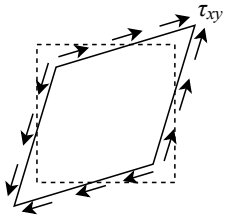
Normal Stress	Shear Stress
Anisotropic Material (or Orthotropic material loaded along non principle axis)	
 $\varepsilon_x = \frac{\sigma_x}{E_x}$ $\varepsilon_y = -\frac{\nu_{xy}\sigma_x}{E_x}$ $\gamma_{xy} = \eta_{xs}\ \varepsilon_x$	 $\gamma_{xy} = \frac{\tau_{xy}}{G_{xy}}$ $\varepsilon_x = \eta_{sx}\gamma_{xy}$ $\varepsilon_y = \eta_{sy}\gamma_{xy}$
Orthotropic Material loaded along principle material axis	
 $\varepsilon_1 = \frac{\sigma_1}{E_1}$ $\varepsilon_2 = -\frac{\nu_{12}\sigma_1}{E_1}$ $\gamma_{12} = 0$	 $\gamma_{12} = \frac{\tau_{12}}{G_{12}}$ $\varepsilon_1 = \varepsilon_2 = 0$
Isotropic Material	
 $\varepsilon_x = \frac{\sigma_x}{E}$ $\varepsilon_y = -(\frac{\nu\sigma_x}{E})$ $\gamma_{xy} = 0$	 $\gamma_{xy} = \frac{\tau_{xy}}{G}$ $= (2\tau_{xy}\frac{1+\nu}{E})$ $\varepsilon_x = \varepsilon_y = 0$

Figure 2.11: Materials under loading conditions (adapted from [12]).

Figure 2.11 presents an overview of different material types under various loading conditions. On application of loading in any direction, an anisotropic material produces axial, transverse and shear deformations. Similarly, shear loading produces axial, transverse deformations along with shear deformations. Shear and axial deformations are related using shear coupling coefficients (η_{xs}, η_{sx} and η_{sy}). Orthotropic materials under loading along the principle material axis produce pure axial and transverse deformations. Similarly, with pure shear loading, pure shear deformations are produced. When loaded along a non-principle material axis, orthotropic materials behave like anisotropic materials. Anisotropic materials under uniaxial loading produce similar pure axial and transverse deformations. Under pure shear load, isotropic material produces pure shear deformations. The modulus of elasticity, in this case, is the same in all directions. Composite materials are considered as anisotropic in nature. Hence, their behaviour is the function of material orientation and loading direction. Even though composite material show high anisotropic characteristics at microscopic scale, they can be modelled as an orthotropic material at macroscopic scale under assumptions. In this work, cross-ply laminate exhibiting orthotropic material property is used for performing analysis.

With a brief introduction of analysis types and material types used in composite materials, the next subsection presents the constitutive equations used in the macromechanical analysis of composite materials.

2.2.2 Macromechanics of Composite Lamina

In this section macromechanics of lamina is discussed. The state of stress at any point in general continuum is represented using nine components of a stress tensor, $\boldsymbol{\sigma}$, as shown in Figure 2.12. The state of deformation is represented by nine components of strain tensor, $\boldsymbol{\varepsilon}$. For a linear elastic solid body, these stresses and strains are related using Hook's law [57] given by,

$$\boldsymbol{\sigma} = \mathbf{C} : \boldsymbol{\varepsilon}, \tag{2.1}$$

where, $\mathbf{C}$ is fourth order material tensor (stiffness matrix) which contains eighty one terms. Equation 2.1 can also be written in the Voigt/Engineering notation using subscripts as follows,

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl}, \quad i,j,k,l \in \{1,2,3\}. \tag{2.2}$$

After applying symmetry conditions on material tensor, the number of terms in the material tensor reduces drastically to thirty six. For anisotropic material, this reduced constitutive equation is given by,

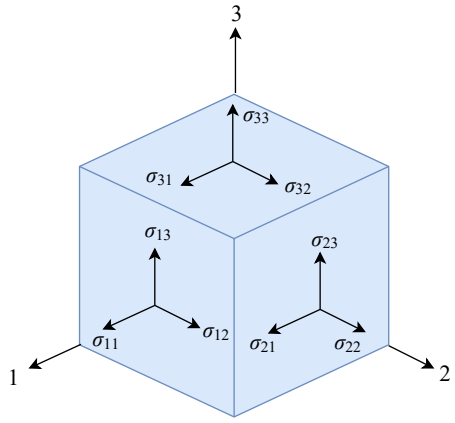


Figure 2.12: Stress at a point in continuum.

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{31} \\ \sigma_{12} \end{Bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} \\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} \end{bmatrix} \begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{23} \\ 2\varepsilon_{31} \\ 2\varepsilon_{12} \end{Bmatrix}. \tag{2.3}$$

As discussed earlier, orthotropic materials have three mutually perpendicular axes of symmetry. Because of this property, the number of coefficients in material tensor reduces further to nine as follows,

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \tau_{23} \\ \tau_{31} \\ \tau_{12} \end{Bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{21} & C_{22} & C_{23} & 0 & 0 & 0 \\ C_{31} & C_{32} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{66} \end{bmatrix} \begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \gamma_{23} \\ \gamma_{31} \\ \gamma_{12} \end{Bmatrix}, \tag{2.4}$$

where, shear stresses are used as,

$$\tau_{23} = \sigma_{23}, \quad \tau_{31} = \sigma_{31}, \quad \tau_{12} = \sigma_{12}, \tag{2.5}$$

and strains are used as,

$$\gamma_{23} = 2\varepsilon_{23}, \quad \gamma_{31} = 2\varepsilon_{31}, \quad \gamma_{12} = 2\varepsilon_{12}. \tag{2.6}$$

The inverse of the Equation 2.4 can be written as,

$$\begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \gamma_{23} \\ \gamma_{31} \\ \gamma_{12} \end{Bmatrix} = \begin{bmatrix} S_{11} & S_{12} & S_{13} & 0 & 0 & 0 \\ S_{21} & S_{22} & S_{23} & 0 & 0 & 0 \\ S_{31} & S_{32} & S_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & S_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & S_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & S_{66} \end{bmatrix} \begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \tau_{23} \\ \tau_{31} \\ \tau_{12} \end{Bmatrix}, \tag{2.7}$$

where S_{ij} are compliance coefficients of compliance matrix $\mathbf{S}$. The Equation 2.7 can be represented in terms of engineering constants like Young's modulus (E), shear modulus (G) and Poisson's ratio (ν) as follows,

$$\begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \gamma_{23} \\ \gamma_{31} \\ \gamma_{12} \end{Bmatrix} = \begin{bmatrix} \frac{1}{E_{11}} & -\frac{\nu_{21}}{E_{22}} & -\frac{\nu_{31}}{E_{33}} & 0 & 0 & 0 \\ -\frac{\nu_{12}}{E_{11}} & \frac{1}{E_{22}} & -\frac{\nu_{32}}{E_{33}} & 0 & 0 & 0 \\ -\frac{\nu_{13}}{E_{11}} & -\frac{\nu_{23}}{E_{22}} & \frac{1}{E_{33}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{23}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{13}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{12}} \end{bmatrix} \begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \tau_{23} \\ \tau_{31} \\ \tau_{12} \end{Bmatrix}. \tag{2.8}$$

For different loading conditions on an orthotropic material, Equation 2.8 can be simplified as follows,

- **Longitudinal tension/compression**
  Loading is applied along fibre direction/principle material axis as shown in Figure 2.13. Stress and strains in terms of material properties are related as follows,

$$\varepsilon_{11} = \frac{\sigma_{11}}{E_{11}} = S_{11}E_{11}, \quad \varepsilon_{22} = -\frac{\nu_{12}}{E_{11}}\sigma_{11} = S_{22}E_{22}, \quad \gamma_{12} = 0. \tag{2.9}$$

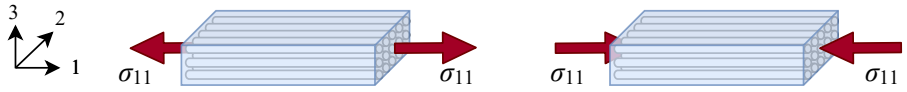


Figure 2.13: Longitudinal tension.

  The longitudinal tensile strength in this case is denoted by X_t, while longitudinal compressive strength is denoted by X_c.

- **Transverse tension/compression (in-plane)**
  Loading is applied perpendicular to fibre direction/principle material axis as shown in Figure 2.14. Stress and strains in terms of material properties are related as follows,

$$\varepsilon_{11} = -\frac{\nu_{21}}{E_{22}}\sigma_{22} = S_{12}\sigma_{22}, \quad \varepsilon_{22} = \frac{\sigma_{22}}{E_{22}} = S_{22}\sigma_{22}, \quad \gamma_{12} = 0. \tag{2.10}$$

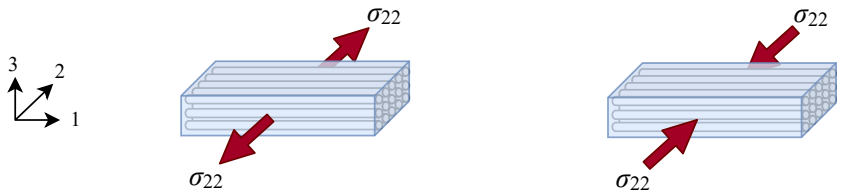


Figure 2.14: Transverse tension and compression.

The transverse tensile strength in this case is denoted by Y_t, while transverse compressive strength is denoted by Y_c.

- **In-plane shear load**
  In this case shear load is applied in plane 1-2 as shown in Figure 2.15. Stress and strains in terms of material properties are related by,

$$\varepsilon_{11} = 0, \quad \varepsilon_{22} = 0, \quad \gamma_{12} = \frac{\tau_{12}}{G_{12}} = S_{66}\tau_{12}. \tag{2.11}$$

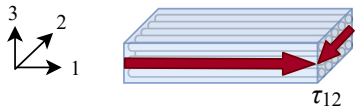


Figure 2.15: In-plane shear.

Longitudinal and transverse in-plane shear strength in this case is denoted by S_L and S_T.

It can be be observed that there are in total six strength parameters (X_t, X_c, Y_t, Y_c, S_L and S_T) which are used to assess failure in a UD lamina. Stress values exceeding these would indicate a failure in the lamina. Based on these values, some failure theories are developed which are discussed in Section 2.2. For simplicity, the derivations of all the equations are not discussed here. More details about this topic can be found in [12] and [57]. Failure in composite structure is a multi-scale process that comprises many local damage mechanisms simultaneously occurring. Although macromechanical analysis is beneficial for large structures, it poses difficulty in modelling single cracks and microscopic damage accumulation. Micromechanical analysis helps in providing deeper insights into such microscopic development of failure in the lamina. Such analysis helps in accurately understanding how the underlying behaviour at the microscopic level affects the ultimate macroscopic properties of the lamina. The next subsection briefly discusses the micromechanics of lamina.

2.2.3 Micromechanics of Composite Lamina

As discussed in the beginning of Section 2.2, micromechanics consists of studying interactions between constituents materials of a composite material on a microscopic scale. Due to computational limitation, it is impractical to do such analysis for large structures. Despite this limitation, micromechanics of lamina helps in understanding failure mechanisms, material properties theoretically, which lays ground work for developing failure models. Hence developing analytical models that would help in calculating these values is also one of the motivation to do the micromechanical analysis of lamina. Micromechanical analysis begins with selection of a representative volume element (RVE) as shown in Figure 2.16.

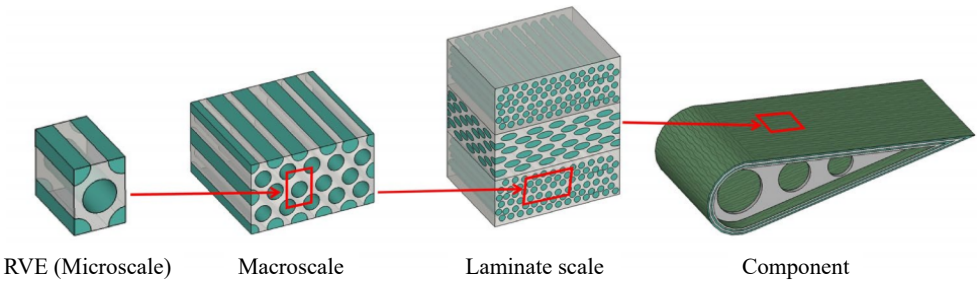


Figure 2.16: Representation of RVE in a composite structure[48]

RVE is a sub-volume of composite material. The composite is considered as an assembly of these periodic RVEs. The statistical characters like volume fractions for the whole composite component are considered same as that of RVE. Micromechanics aims to obtain homogenized material properties at this smallest scale. Several analytical approaches have long existed in this regard[26, 43, 58, 74]. Numerical techniques mostly relay on finite element discretization methods and is widely implemented in commercial finite element codes[48]. In FEA, the matrix and fibre phases in the RVE are discretized into number of finite elements. The discretized RVE is the subjected to uniform stress or displacements using periodic boundary conditions. The stresses in the elements are averaged over volume to calculate average strains. Effective material properties of composite is then calculated from the analysis of these RVEs. For the simplicity purposes the discussion of micromechanical analysis is limited upto this point. For more information [12, 48] can be referred.

Chapter 2 discussed the basics of constituent materials of composite materials and its mechanics. Multi-scale modelling of composite materials are reviewed in brief. The next chapter presents various failures occurring in composite materials. Failure criteria, based on the concepts discussed in this chapter, are presented in brief.

3 Failure in Composites

This chapter builds upon various concepts of composite material and its mechanics discussed in Chapter 2. In this chapter, first various damages occurring in the composite are presented, followed by an overview of various failure theories. Failure model used in this work is discussed in details, in the in the later part of the chapter.

3.1 Damages in Composite Material

Damages in composite material occur at various scales ranging from micro-structure to laminate. Composite materials have many complex damage mechanisms when compared to traditional materials like metals. In subsequent subsections, a few of these are discussed generically.

- **Fibre-Matrix Debonding**
  Debonding is intralaminar damage, which occurs due to high shear stresses at the fibre-matrix interface. Figure 3.1(a) shows the SEM image showing fibre matrix debonding. The high stresses are developed because of high strains in a matrix material which are present between two fibres having Young's modulus greater than the matrix. In a UD lamina under longitudinal tensile load, fibre fracture can act as a precursor to debonding (Figure 3.1(b)). This can also occur due to manufacturing defects like poor adhesion between fibre and matrix. This damage can coalesce into a matrix crack leading to the overall failure of the whole composite.

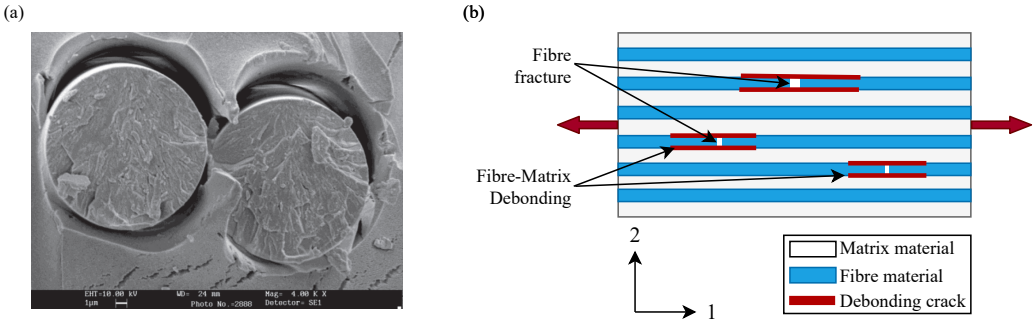


Figure 3.1: Fibre-Matrix Debonding: (a) SEM image[71] (b) UD lamina under longitudinal tensile load.

- **Matrix Cracking**
  This is one of the most serious types of damage found in composite structures. It causes total failure of the composite. Fibre-Matrix debonding acts as a precursor of this damage. It is usually found in 90° ply (or off-axis plies) when longitudinal tensile

or compressive load is applied on composite with the brittle matrix. During transverse loading, because of the high difference in stiffness of fibres and matrix, substantial strain magnification occurs in the matrix[2]. This results in transverse matrix cracking. This damage causes an overall stiffness reduction of the composite. Figure 3.2 shows matrix cracking under various load conditions.

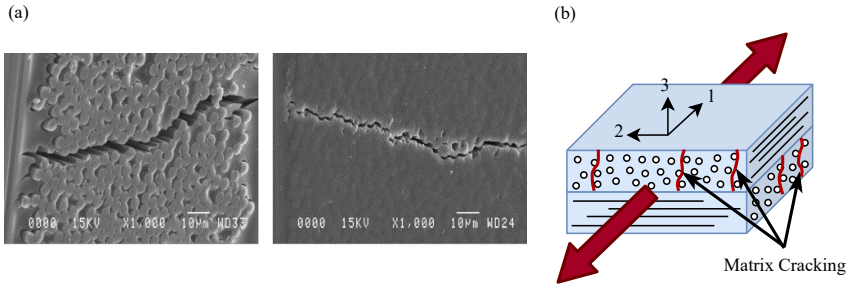


Figure 3.2: Fibre-Matrix Cracking: (a) SEM micrograph [17] (b) Laminate under longitudinal tensile load.

- **Fibre Fracture**

  Defects in fibres occurred during the manufacturing of fibre (e.g. scratches) can lead to fibres with different failure strain rate bundled together. These fibres under loading fracture at different strains. These isolated fibre fractures can cause failure in adjacent fibres, leading to total fibre fracture and composite failure. Other damage mechanisms like matrix cracking can also lead to localized stresses in fibres leading to fibre fracture. This failure is usually found in fibres that are parallel to the applied load[41]. Figure 3.3 shows SEM images of fibre fracture of impacted composite Figure 3.3(a) and composite under bend test Figure 3.3(b). Fibre fracture can also be induced during kinking. Kinking is the localized shear deformation occurring in the matrix during compressive loads. In this kind of damage, fibres are rotated by a large amount and the matrix undergoes large shear deformation as shown in Figure 3.3(c).

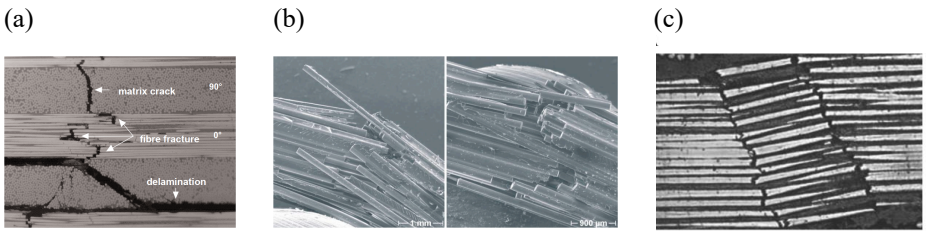


Figure 3.3: Fibre fracture: (a) SEM micrograph of impact loading on composites[13] (b) SEM micrograph of bend test on composites[7] (c) Kinking[24].

- **Delaminations**

  In this kind of damage, plies get separated from each other, leading to degradation in

mechanical properties of the overall composite structure. Delaminations are caused by interlaminar stresses at the free edges. These stresses are the consequence of different moduli of constituent material and different material orientation between lamina. These stresses can be shear or normal in nature. Matrix cracking usually acts as a precursor for delaminations. Manufacturing defects like improper curing can also cause such failure. Figure 3.4 illustrates delamination damage occurring in composites.

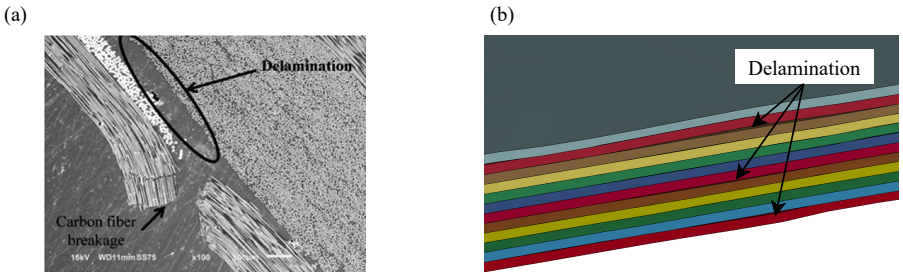


Figure 3.4: Delamination: (a) Flexural failure of CFRP composite[79] (b) FEM analysis of composite under tensile loading.

Damages discussed above can occur in various manner depending upon the type of loading conditions, ply orientation and material properties. In OHT specimen under tensile loading, all kinds of damages like micro-cracks, delamination, fibre breakage, first initiate from notch/hole due to high-stress concentrations. On increasing loads further, the delamination propagates between plies and becomes splitting bands near the edge. The micro-cracks generated near the notch also travel across plies, initiating intralaminar matrix cracking. These cracks also induce debonding between fibres and matrix, travelling further across plies. Because of debonding, the stresses are not transferred towards low-stress regions. This results in fibres undergoing maximum stresses, which in turn leads to fibre fracture and fibre pull-outs. Near the ultimate strength, the plies with fibres parallel to the loading direction, bear the most of the loads. This eventually leads to fibre fracture in such plies resulting in complete structural failure. Figure 3.5 shows various damages occurring in cross-ply laminate under longitudinal tensile loading.

Having discussed different failures in composite, the next section presents various failure models and theories that are used to numerically model and predict these failures in analysis like FEM.

3.2 Failure Models

As discussed in Section 3.1, composites have very complex failure and damage mechanisms. Because of this reason, it is still very challenging to accurately predict the failure behavior. Development of failure criteria for composites has been going on for over 30 years and is still

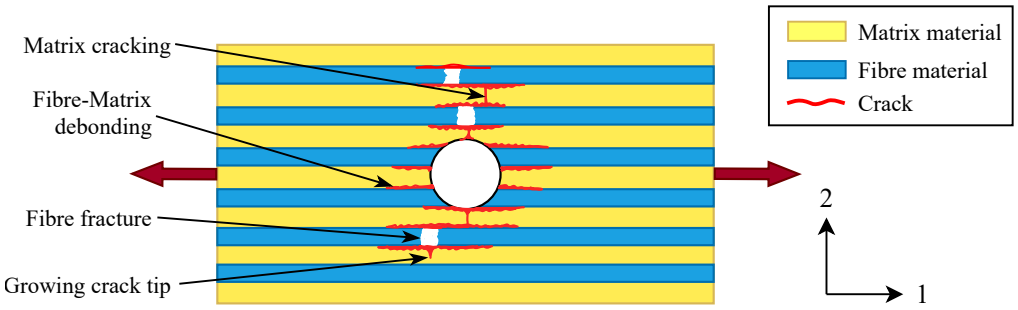


Figure 3.5: Various damages occurring in composite under longitudinal tension.

an actively ongoing field of research. In this section, various failure theories and models are presented in brief.

Most of the failure criteria are derived from the theory of strength of materials and use failure envelopes of composites to predict failure. Some known theories are maximum stress/strain, Hoffman[29], Tsai-Wu[73], Tsai-Hill[28] and Hashin[25]. Among these, some theories like Tsai-Hill, Hoffman and Tsai-Wu don't distinguish between different failure modes (fibre/matrix tensile/compression failure, matrix shear failure) while some theories like maximum stress/strain, Puck[53] can specify failure mode depending upon loading conditions. Hoffman failure criterion considers the difference between tensile and compressive strength. failure criteria from Chang-Chang[11] considers matrix cracking, fibre breakage and fibre matrix shearing for predicting failure. The failure criterion proposed by Puck differentiates between fibre failure and inter-fibre fracture. It also takes tensile and compressive fibre failure modes into account. Using Puck's criteria, fracture angle can also be predicted. The finite fracture mechanics method proposed by Camanho et al.[9] uses stress and energy-based criteria to predict the failure in OHT specimen composite. Pinho et al.[51, 52] proposed physically-based failure criteria for three-dimensional model. This model distinguishes between fibre and matrix tensile/compressive failure modes.

Delamination is a special mode of composite failure that depends upon ply layup and material orientation. Numerous research has been done to predict this failure accurately by using the different combination of failure modes. Failure models of delamination are categorized into delamination initiation and delamination propagation. The former is usually based on point or average stress failure criteria (Whitney et al.[75]). The failure models for delamination propagation are generally based on the fracture mechanics approach, which evaluates the energy release rate [5, 8, 78]. In Sun et al.[70], modes I and II were investigated to energy release rates for interfacial cracks. In Benzeggagh et al.[5] failure model for mode I, II and III are proposed and is expressed in terms of total critical strain energy release rate and the total fracture resistance. The virtual crack closure (VCC) technique is another approach

proposed by Rybicki et al.[60]. It is based on an assumption that the energy released during crack extension is equal to the work required to close it to its original length.

In this work, composites materials for FEM analysis are modelled with a failure model proposed by Pinho et al.[51, 52]. The delamination is modelled using traction separation law proposed by Benzeggagh-Kenane [5]. Henceforth the discussion is limited to these models only. Both models are implemented in explicit FEM analysis software LS-Dyna [36].

3.2.1 Composite Material Modelling

Pinho et al. [51, 52] proposed a model which considers a physical model covering several failure modes along with non-linear matrix shear behaviour occurring in FRP ply. Failure modes examined in the model are analyzed under loading conditions shown in Figure 3.6.

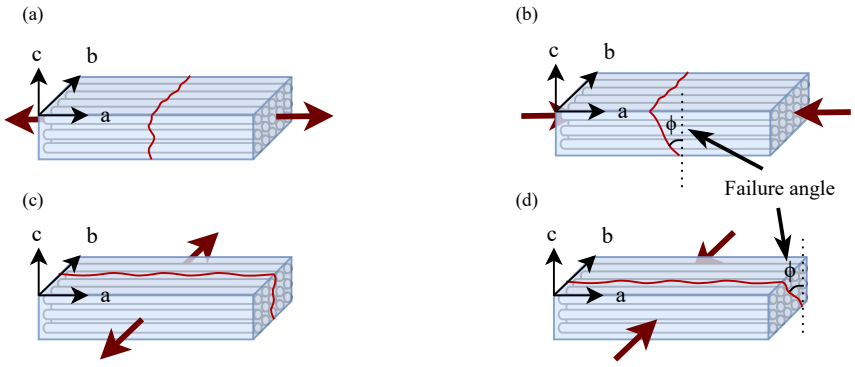


Figure 3.6: (a) Longitudinal tensile failure (fibre failure mode) (b) Longitudinal compressive fracture (Compressive fibre mode) (c) Transverse fracture (Tensile matrix mode) (d) Transverse fracture (Compressive matrix mode).

This model is implemented in *MAT_261 material card in LS-Dyna (Subsection A.1.1). Henceforth, throughout the discussion of the model, 'a' is considered as fibre direction, 'b' is considered as in-plane transverse direction, and 'c' is considered as through-the-thickness direction. In this model, various failure criteria for different failure modes are proposed as follows,

- **Fibre tensile failure**

  In this kind of failure, longitudinal (along fibre) stresses are found to be the most influential. Hence maximum stress criteria is used to model such failure. The yield condition is given by,

  $$f_{\mathrm{ft}} = \frac{\sigma_{\mathrm{a}}}{X_{\mathrm{t}}} = 1, \tag{3.1}$$

  where subscript 'ft' represents fibre tensile, σ_{a} is the longitudinal stress and X_{t} is the longitudinal tensile strength (discussed in Subsection 2.2.2).

- **Fibre compression failure (Kinking)**

  A new model for three-dimensional kinking is proposed, as a generalization of the two-dimensional kinking model. It assumes initial fibre misalignment, θ (Figure 3.7) and non-linear shear behaviour of composites. Because of the three-dimensional nature of fibre-kinking, the kinking plane is transformed from two dimensions to three dimensions at an angle, Ψ as shown in Figure 3.7. The value of Ψ depends upon assumed conditions. For 2D kinking, where movement in c-axis is constrained, the Ψ will be zero. The failure criterion is given by,

$$f_{\text{kink}} = \begin{cases} \left(\frac{\tau_{\text{T}}}{S_{\text{T}}-\mu_{\text{T}}\sigma_{\text{n}}}\right)^2 + \left(\frac{\tau_{\text{L}}}{S_{\text{L}}-\mu_{\text{L}}\sigma_{\text{n}}}\right)^2 = 1 & \text{if } \sigma_{\text{b}^{\text{m}}} \leq 0, \\ \left(\frac{\sigma_{\text{n}}}{Y_{\text{t}}}\right)^2 + \left(\frac{\tau_{\text{T}}}{S_{\text{T}}}\right)^2 + \left(\frac{\tau_{\text{L}}}{S_{\text{L}}}\right)^2 = 1 & \text{if } \sigma_{\text{b}^{\text{m}}} > 0, \end{cases} \tag{3.2}$$

  with,

$$\begin{aligned} S_{\text{T}} &= \frac{Y_c}{2\tan(\phi_0)}, \\ \mu_{\text{T}} &= -\frac{1}{\tan(2\phi_0)}, \\ \mu_{\text{L}} &= S_{\text{L}}\frac{\mu_{\text{T}}}{S_{\text{T}}}, \\ \sigma_{\text{n}} &= \frac{\sigma_{\text{b}^{\text{m}}} + \sigma_{\text{c}^{\Psi}}}{2} + \frac{\sigma_{\text{b}^{\text{m}}} - \sigma_{\text{c}^{\Psi}}}{2}\cos(2\phi) + \tau_{\text{b}^{\text{m}}\text{c}^{\Psi}}\sin(2\phi), \\ \tau_{\text{T}} &= -\frac{\sigma_{\text{b}^{\text{m}}} - \sigma_{\text{c}^{\Psi}}}{2}\sin(2\phi) + \tau_{\text{b}^{\text{m}}\text{c}^{\Psi}}\cos(2\phi), \\ \tau_{\text{L}} &= \tau_{\text{a}^{\text{m}}\text{b}^{\text{m}}}\cos(\phi) + \tau_{\text{c}^{\Psi}\text{a}^{\text{m}}}\sin(\phi), \end{aligned} \tag{3.3}$$

  where, subscripts L and T denotes longitudinal and transverse direction in the fracture plane ϕ. S is the shear strength. μ denotes the friction coefficient. τ and σ denotes shear and normal stresses in the fracture plane. Y_{t} and Y_{c} denotes in-plane tensile and compressive strength in transverse direction, respectively (Subsection 2.2.2).

- **Matrix tensile failure (Transverse direction)**

  The failure criterion is given by quadratic relation between shear stresses and normal stresses in fracture plane as follows,

$$f_{\text{mt}} = \left(\frac{\sigma_{\text{n}}}{Y_{\text{t}}}\right)^2 + \left(\frac{\tau_{\text{T}}}{S_{\text{T}}}\right)^2 + \left(\frac{\tau_{\text{L}}}{S_{\text{L}}}\right)^2 = 1 \quad \text{if} \quad \sigma_{\text{n}} \geq 0, \tag{3.4}$$

  with,

$$\begin{aligned} \sigma_{\text{n}} &= \frac{\sigma_{\text{b}} + \sigma_{\text{c}}}{2} + \frac{\sigma_{\text{b}} - \sigma_{\text{c}}}{2}\cos(2\phi) + \tau_{\text{bc}}\sin(2\phi), \\ \tau_{\text{T}} &= -\frac{\sigma_{\text{b}} - \sigma_{\text{c}}}{2}\sin(2\phi) + \tau_{\text{bc}}\cos(2\phi), \\ \tau_{\text{L}} &= \tau_{\text{ab}}\cos(\phi) + \tau_{\text{ca}}\sin(\phi), \end{aligned} \tag{3.5}$$

  where τ, σ are traction components as shown in Figure 3.8.

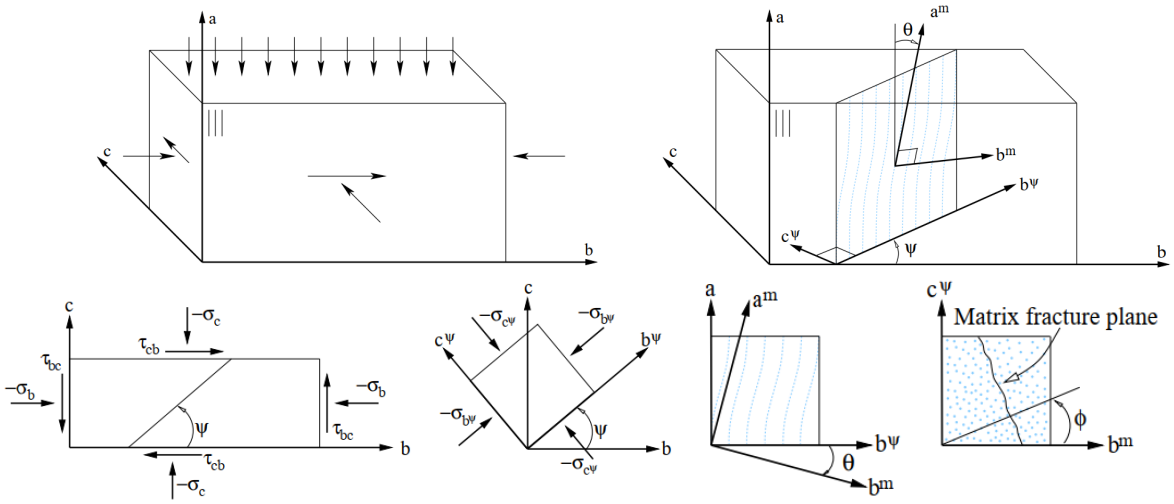


Figure 3.7: Definition of angles and stress for kinking in 3D space[52].

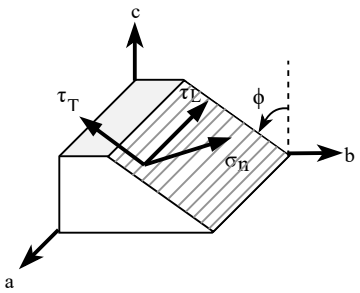


Figure 3.8: Traction components on fracture plane at angle ϕ[52].

- **Matrix compressive/shear failure (Transverse direction)**
  The matrix compressive/shear failure in transverse direction is modelled as follows,

$$f_{\text{mc}} = \left(\frac{\tau_{\text{T}}}{S_{\text{T}} - \mu_{\text{T}}\sigma_{\text{n}}}\right)^2 + \left(\frac{\tau_{\text{L}}}{S_{\text{L}} - \mu_{\text{L}}\sigma_{\text{n}}}\right)^2 = 1 \text{ if } \sigma_{\text{n}} < 0. \tag{3.6}$$

  The non-linear in-plane shear behavior of laminated composites is modelled by coupling 1D-elastoplastic formulation to a linear damage behavior. They are coupled at the point where maximum allowed shear stress is reached as shown in the figure Figure 3.9(a).

The composite laminate model operates as an orthotropic elastic material as long as stresses in the material stay within the failure surface. Once the failure criteria are satisfied, undamaged stresses are reduced linearly by a factor of $(1 - d)$ where d is a damage variable. The damage variable d is defined for each failure mechanisms as d_{a} for fibre tensile failure, d_{kink} for fibre compressive failure (kinking), d_{mat} for matrix transverse tensile and d_{mac} for matrix transverse compressive failure. These damage variables follow a linear damage evolution law based on fracture toughness Γ and a characteristic internal element length

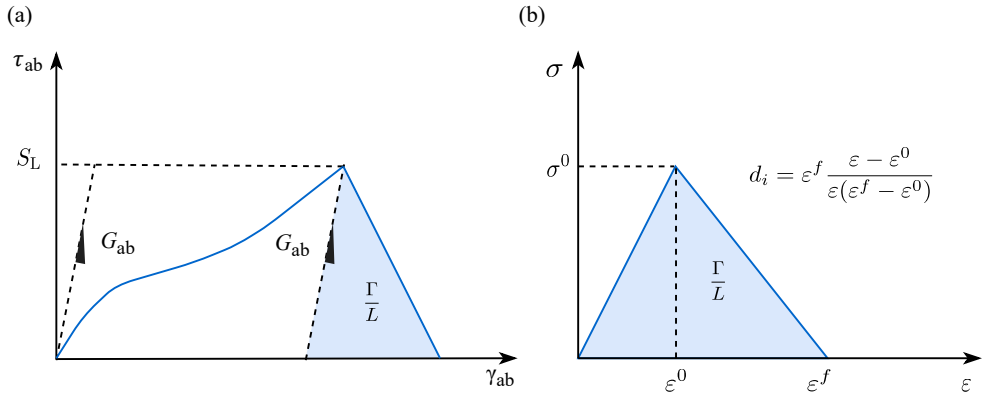


Figure 3.9: (a) Non-linear in-plane shear behavior (b) Damage evolution law [37] .

L, as shown in Figure 3.9(b). Fracture toughness is a mechanical property that measures material's resistance to fracture [14]. This parameter characterizes the stress field intensity in the material near the crack tip during crack growth. At every time step, the damage variable degrades the stiffness matrix depicting material property degradation. The new degraded stiffness matrix is calculated as follows,

$$\begin{aligned}
\hat{\sigma} &= (1 - d_a)[\sigma_a, \sigma_b, \sigma_{ab}, \sigma_{bc}, \sigma_{ca}], \\
\hat{\sigma} &= (1 - d_{kink})[\sigma_a, \sigma_b, \sigma_{ab}, \sigma_{bc}, \sigma_{ca}], \\
\hat{\sigma} &= (1 - d_{mat})[\sigma_b, \sigma_{ab}, \sigma_{bc}, \sigma_{ca}], \\
\hat{\sigma} &= (1 - d_{mac})[\sigma_b, \sigma_{ab}, \sigma_{bc}, \sigma_{ca}].
\end{aligned} \tag{3.7}$$

When the damage variable of an element on all its through-the-thickness integration points, associated with corresponding failure mode reaches the maximum value ($d_{max} = 1$), the material is assumed to be failed and the element is then deleted. In the next subsection, the delamination model used in the study is discussed.

3.2.2 Delamination Modelling

In LS-Dyna delamination is modelled using *MAT_186 (*MAT_COHESIVE_GENERAL) material card. Delamination is modelled using zero-thickness decohesion elements with eight nodes as shown in Figure 3.10(a). δ denotes relative displacements across ply interface. $\delta_I = \delta_3$ is the relative displacement in the normal direction (3), $\delta_{II} = \sqrt{\delta_1^2 + \delta_2^2}$ is the relative displacement in the tangential direction (in 1-2 plane) and $\delta_{III} = \sqrt{\delta_I^2 + \delta_{II}^2}$ is the mixed mode relative displacement. A cohesive material is said to be failed when the relative displacement exceeds maximum separation (failure separation distance), δ^F. The maximum separation is defined using the traction separation law curve (TSLC) as shown in Figure 3.10(b). In

this work failure criterion proposed by Benzeggagh and Kenane[5] is used for defining the maximum separation distance. For mode I, mode II and mode III, it is given as follows,

$$\text{Mode I (normal mode)} : \qquad \delta_{\mathrm{I}}^{F} = \frac{G_{\mathrm{I}}^{C}}{A_{\mathrm{TSLC}} T}, \tag{3.8}$$

$$\text{Mode II (tangential mode)} : \delta_{\mathrm{II}}^{F} = \frac{G_{\mathrm{II}}^{C}}{A_{\mathrm{TSLC2}} S}, \tag{3.9}$$

Mode III (mixed mode) :

$$\delta_{\mathrm{III}}^{F} = \frac{1+\beta^2}{A_{\mathrm{TSLC}}\mathrm{T} + A_{\mathrm{TSLC2}} S\beta^2} \left[G_{\mathrm{I}}^{c} + (G_{\mathrm{II}}^{c} - G_{\mathrm{I}}^{c}) \left(\frac{A_{\mathrm{TSLC2}} S\beta^2}{A_{\mathrm{TSLC}} T + A_{\mathrm{TSLC2}} S\beta^2} \right)^{\mathrm{XMU}} \right], \tag{3.10}$$

where $\beta = \delta_{\mathrm{II}}/\delta_{\mathrm{I}}$ is the mode mixity, A is the area under TSLC2 curve. T and S denotes peak traction in normal and tangential direction, respectively. G_{I} and G_{II} are the fracture toughness (or energy release rate) for mode I and II, respectively. XMU is the parameter used in Benzeggagh-Kenane failure criterion. Higher the value of XMU is, higher the fracture toughness in mixed mode will be. In this model, irreversible conditions are enforced with loading path coming from the origin and unloading paths path pointing to the origin.

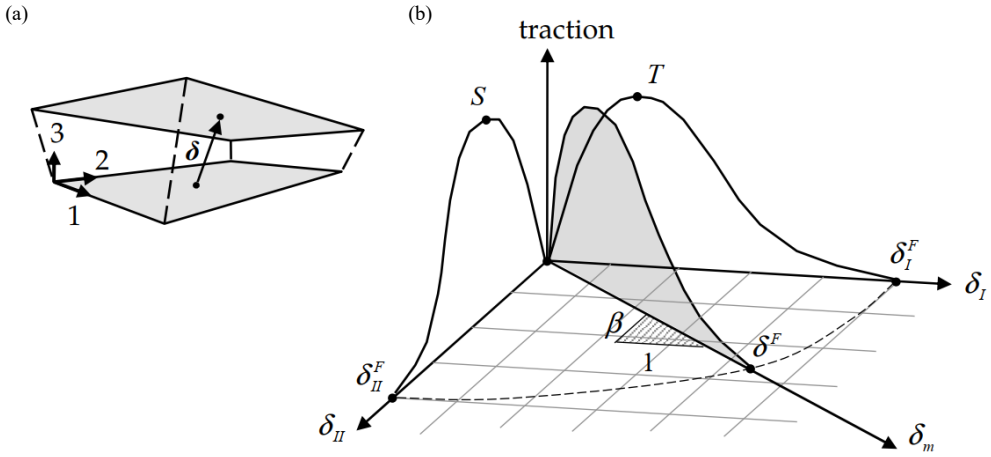


Figure 3.10: (a) Decohesion element (b) Traction separation law for various modes [37]

This chapter reviewed different failure models and criteria in brief. Two failure models Pinho et al.[51, 52] and Benzeggagh-Kenane[5] are discussed in details. Based on these theories, different virtual testings are performed to assess the composite structures.

4 Design of Composite Structures

Composite material selection is an important part of designing any composite structure or component. The function of the component decides which kind of material properties are required by the material. Some components require high load-bearing capability or impact resistance (when used in the cockpit) and some components with holes require the capability to withstand high stress concentrations. One advantage of composite materials is that their properties can be adjusted by selecting appropriate constituent materials. Since many combinations of reinforcement and matrix material are possible, their properties are highly customizable. The different combinations of ply orientations add to the complexity of selecting material with required properties. Understanding how the material properties like density, Young's modulus and Toughness affects the whole component, could help in narrowing down the selection of appropriate composite material. Several studies have has been conducted in the past to assess the effect of material properties and geometry. This chapter reviews such studies and experiments in brief.

In Ercin et al.[15], an OHT specimen with a different ply orientation was tested using various failure criteria and later was compared with experimental results. Two specimens with orientation, OHT1=$[90/+45/0/-45]_{3s}$ (minimum angle difference) and OHT2 = $[90_2 0_2/45_2/45_2/\ 90/0/45/45]_s$ (maximum angle difference), were used. It was found that the specimen with the maximum difference in orientation angle between consecutive plies (OHT2) showed more pronounced damage mechanisms like debonding and delamination. It was also found that as the size of the notch increased, the strength of the specimen decreased. In Moure et al.[44] influence of Weibull modulus on OHT specimen subjected to in-plane loads is analyzed. Weibull modulus from the Weibull distribution is used to predict fibre failure of the laminate. Increasing the value of the Weibull modulus resulted in reduced failure strength of laminate and an increase in stress concentration. Zhang et al.[80] studied an open hole specimen under longitudinal load using a three-dimensional progressive damage analysis model. Fibre kinking and shear non-linearity were considered in this model. Matrix damage was observed first in tensile testing while in compression testing, micro-buckling was observed first. From the observations, it was suggested that damages in 0° plies in composites can be a good indication of the extent of damages occurring in the composite structure. In Xiao et al.[76] it was found that open hole composite laminates under tensile loads, with weak interfacial strength, suffered massive delamination and splitting bonds. This helped in releasing the stress concentration energy, thus increasing overall strength. Specimen with higher interfacial strength had lesser delamination and suffered brittle fracture at lower loads. Hallett et al.[23] presented virtual testing of different ply configurations using FEM. It was shown how parameters like thickness scaling, in-plane dimensions, stacking sequence and

specimen width to hole diameter ratio affects the tensile strength of the OHT specimen. Among all the parameters, ply block thickness was found to be the the most influential parameter. Increasing its value increased intra-ply splitting and inter-ply delamination. Increasing the width to hole diameter ratio increased the strength of the specimen. It was observed that an increase in this ratio can increase the difficulty for damage to propagate from the notched edge towards the free edge (due to increased distance between notch edge and free edge). Andrew et al.[3] presented a critical review on the influence of key parameters- impact velocity, material, geometry, event and environment-related conditions on impact loading of composites. The fabrication processes also affects the properties of a composite structure. Damages are induced while cutting and joining processes. Schmidt et al[63] investigated effect of laser cutting process on an OHT specimen. Fibre laser cut specimen showed higher tensile strength at medium heat affected zones, compared to milled specimen. Abish et al.[1] demonstrated effect of cryogenic assisted drilling of CFRP specimen. It was observed that, under high feed rate and cutting velocity; delamination factor, surface roughness and acoustic emissions are reduced with such setup. Hui et al.[30] studied the effect of residual stresses induced in autoclave curing process using multi-scale modelling strategy.

From the discussion, it can be understood how varying material and geometrical properties can affect the performance of the components. Virtual testing is gaining popularity as a cheaper alternative to physical testing, as more and more efficient failure models are being developed. Although they are more convenient, but it is very important that the results from virtual testing conforms with the results of physical testing. As discussed previously, this work focuses on studying effect of material and geometrical properties using physical testing. In the next chapter aspects of designing computer experiments using design of experiment approaches are discussed.

5 Design of Experiments and Sensitivity Analysis

The importance of conducting virtual and physical testing was described in the previous chapter. This chapter discusses methodologies to assess such parameters, using the concept of Design of Experiment (DOE). The first part of this chapter presents different aspects of designing computer experiments. The latter part presents a brief introduction to sensitivity analysis. Sensitivity analysis techniques like correlation coefficients and regression are discussed in this section.

5.1 Design of Experiments

The design of experiment aims to plan and conduct, physical or computer tests, to analyze and interpret systems and processes. The effect of the parameters (or factors) or group of parameters on the system is evaluated as shown in Figure 5.1.

Experiments can be designed under controlled condition or uncontrolled conditions. Controlled experiments comprise factors that can be controlled like material properties. The uncontrolled experiments comprise factors that can't be controlled, such as environmental, ethical factors. The factors that are provided as an input to any model are called as input factors or design variables. They are denoted by X_i where $i \in \{1, 2, 3, ..., n\}$. Design variables are usually varied during an experiment to study its effect on output parameters of the model. The output parameters of the model are also called as response variables. The space formed by the upper and lower bound of each design variable is known as design space. An n-dimensional hypercube design space is denoted by $\mathcal{X}^n$. A specific instance of design

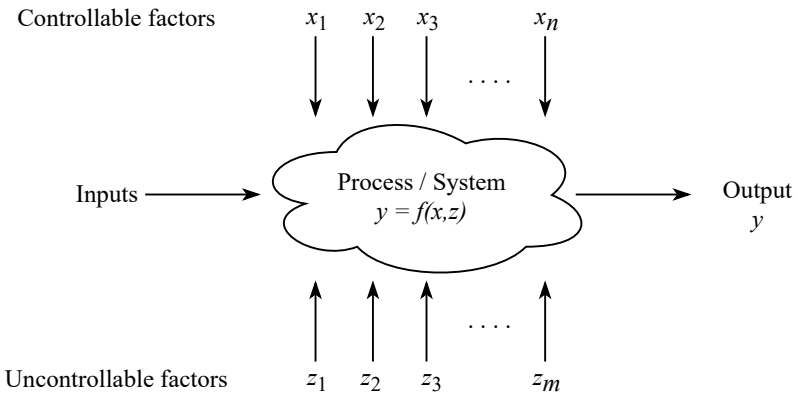


Figure 5.1: General schematic of an experiment.

variable is called a sample or design point. It is denoted by x_{ij} for j^{th} sample. DOE aims to select a set of samples (e_j) in design space in such a way that the maximum information can be gained from a few samples by performing certain p number of experiments. This complete set of runs or experiments are called a design matrix, as shown in Table 5.1.

Experiment	Design Variables					
	X_1	X_2	..	..	..	X_n
e_1	x_{11}	x_{21}	..	..	..	x_{n1}
e_2	x_{12}	x_{22}	..	..	..	x_{n2}
..	..	..	..	..	..	..
e_p	x_{1p}	x_{2p}	..	..	..	x_{np}

Table 5.1: A design matrix.

The design of experiments comprises three key steps. The first step consists of deciding the number of design variables to be used in the experiment. Having some knowledge in the subject area (for example in fracture mechanics) helps in avoiding the selection of unnecessary factors that won't affect the outcome. The second step consists of deciding the number of experiments to be run. With physical testing, because of high costs, a limited number of experiments can be performed. Although computer experiments are more cost-effective than physical experiments, these are computationally very expensive, which could take days to complete a single run. Hence, one should be very economical and careful, while deciding the number of computer experiments as well. There are different methods using which the experiment can be designed. Some of these are as follows,

- One Factor At a Time (OFAT),
- Full factorial designs,
- Taguchi's design,
- Plackett-Burman design,
- Central composite design,
- Box-Behnken design,
- Doehlert design,
- Monte Carlo sampling method,
- Space-filling designs-
  - Latin hypercube design,
  - Sphere packing design.

In this work Latin Hypercube Design (LHD) is used to design experiments. Hence, for simplicity purposes, the discussion will be limited to this method only. More details about other methods can be found in [42]. LHD was proposed by McKay et al.[39]. It is a stratified stochastic sampling method where design space is subdivided into ranges (strata/bins) of equal probability and a sample is chosen at random in each range (bin)[21]. For n design

variables the design space is divided into number of required experiments to be performed (p) resulting in n^p bins. Required samples are then generated by selecting a sample point from each bin in such a way that that there is only one sample point in each axis-aligned hyper-plane of design space. In other words, there will be only one sample in each bin if one-dimensional projections of p samples and bins are taken. Figure 5.2 shows the design space generated for two design variables and four samples. It can be observed that each column and row contains only one sample.

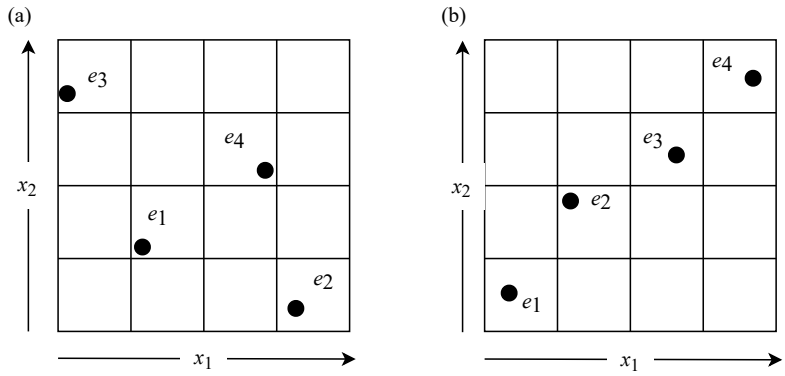


Figure 5.2: Design space in LHD: (a) low correlation (b) high correlation, between design variable sample points.

In LHD, the required number of experiments is independent of the number of design variables. Because of this advantage the sample size (or the total number of runs), is significantly reduced when compared with other methods like full factorial. For example, an experiment with 10 design variables will require $2^{\mathrm{n}} = 2^{10} = 1024$ experimental runs for full factorial design method. In this method the number of input factors greatly affects the computational effort required for performing the whole experiment. But in case of LHD method, the same experiment can be performed in any number of experimental runs irrespective of number of design variables. This significantly reduces the time to perform the whole computer experiment where each run takes a lot of time. It is natural that the higher number of experiments would make the analysis more robust. From Figure 5.2, it can be observed that, more than one design matrix can be generated with LHD method with same number of design variables and sample points. The sample points can have high correlations between them or can have lower correlation between them. The phenomenon of high correlation between design variable sample points is called as collinearity (or multicollinearity)[62]. In such cases it becomes difficult to distinguish the effect of design variables on the response. Hence, design with minimum correlation between design variable sample points are preferred which helps in truly understanding the effect of the individual parameters. The optimal design matrix can be achieved by several methods like entropy [67], integrated mean squared error [61], maximin or minimax distance [33] or minimizing the maximum correlation between sample

points. In this work, optimum LHD design matrix is obtained by minimizing the maximum correlation between samples. The python library pyDOE [68] repeatedly creates several LHD matrices and selects a design matrix with the minimum of maximum correlation observed between sample points of design variables.

The last step in design of experiment is to run the computer simulations using optimal design matrix generated by LHD. The appropriate response is then extracted and read from the experiment. Sensitivity analysis is performed on the response generated from such experiments. In next section the topic of sensitivity analysis is discussed in brief.

5.2 Sensitivity Analysis

Sensitivity analysis (SA) is the procedure to find the dependency of the model output from the given model input. It helps in understanding the influence of each input design variable over model output. In SA, the simulation model is considered a black box. Only the simulation inputs and outputs are considered for the analysis ignoring all the internal variables and procedures present in the computer model[32]. It helps in identifying essential inputs without having deep knowledge of the computer model. There are many techniques to perform sensitivity analysis of a DOE. Some of them are as follows[19, 42, 55],

- Morris-One-at-a-time (MOAT),
- Regression and correlation analysis,
- Derivative based local and global sensitive measures,
- Sobol method,
- Fourier Amplitude Sensitivity Test (FAST),
- Visual methods-
  - Scatter plot,
  - Cobweb plot,
  - Contour plot,
  - Bar Plot,
  - Heatmap/Matrix plot.

Visual methods are used to present the output data in a more intuitive way. In this work, regression and correlation analysis is used for conducting sensitivity analysis. To visualize results, scatter plots and heatmaps are used.

Regression and Correlation analysis are extensively used approaches in sensitivity analysis. These methods are relatively simple and are used to measure strength association between input and output factors. Regression analysis is the method of associating input and output factors using some algebraic expression[27]. In this analysis usually linear models are used which establish a linear relationship between factors as follows,

$$\hat{y}_j = f(x_j) = c_j + k_j x_j, \tag{5.1}$$

where $\hat{y}_j$ is the predicted output from the fitted curve, y is the actual output of the model, x_j is the input factor provided to the model, of the j^{th} sample. Coefficient of the input factor, obtained after fitting a curve, is denoted by k. Curve fitting is usually done by using least square method. In least square methods, coefficients of input parameters are obtained in such a way that the fitted curve has the minimal sum of deviation. Consider a set of input and output points (x_j,y_j) where $j = \{1,2,...,p\}$. The fitting curve $f(x)$ will have an error e_j at each data pointgiven by,

$$e_j = y_j - f(x_j), \tag{5.2}$$

Least Square method will produce coefficients k_j in such a way that the squared sum of the error is minimized i.e.,

$$\text{minimize}\Big(\sum_{i=1}^{p} e_j^2\Big) = \text{minimize}\Big(\sum_{i=1}^{p}(y_j - f(x_j))^2\Big). \tag{5.3}$$

The measure of extent to which regression model matches the observed data (model output) is given by R^2 or R-Squared value [27] given by,

$$R^2 = \frac{\sum_{j=1}^{p}(\hat{y_j} - \bar{y})^2}{\sum_{j=1}^{p}(y_j - \bar{y})^2}, \tag{5.4}$$

where $\bar{y}$ is the mean of observed data. R-Squared measures how good the fit is. It is measured between 0 to 1. Higher value indicates a good fit and that the model is accounting for most of the uncertainty in observed model output (y), where as value near to 0 indicates a bad fit.

Correlation provides a measure of strength of the relationship between input factors (x_j) and observed output factors (y_j). Some of the correlation coefficients used to study the relationships are Pearson's correlation coefficient (PCC) and Spearman's rank correlation coefficient (SRCC). PCC is the strength of the linear relationship between input factors and output factors[55]. It is given by,

$$PCC = \rho = \frac{\sum_{j=1}^{p}(x_{ij} - \bar{x}_i)(y_j - \bar{y})}{\sqrt{\sum_{j=1}^{p}(x_{ij} - \bar{x}_i)^2 \sum_{j=1}^{p}(y_j - \bar{y})^2}}, \tag{5.5}$$

where i and j are i^{th} input factor and j^{th} sample point. The total number of sample points is denoted by p. In case of regression analysis using linear model, R-Squared from Equation 5.4 equals square of PCC [59]. Positive PCC value shows that output factor value will increase with increasing input factor value. In the case of a negative PCC value, the output factor value will decrease with increasing input factor value. In certain cases there exist a non-linear relationship between parameters. A lower PCC value in such cases could be misleading. In this situation, Spearman's rank correlation coefficient (SRCC) denoted by r_s is used. It

assesses the monotonic behaviour between ranked input and output factors, whether they are linearly related or not. In other words, SRCC is the PCC of ranked variables[45]. Rank transformation is used to convert non-linear but monotonic relationship between input and output factors into a linear relationship. Values of factors are replaced by their ranks. The rank of the variables is the order number in which they appear when they are arranged in increasing order. Hence the sample with the lowest value will have the rank as one, the next larger will have the rank as two and so on. Figure 5.3 shows different cases of PCC and SRCC values.

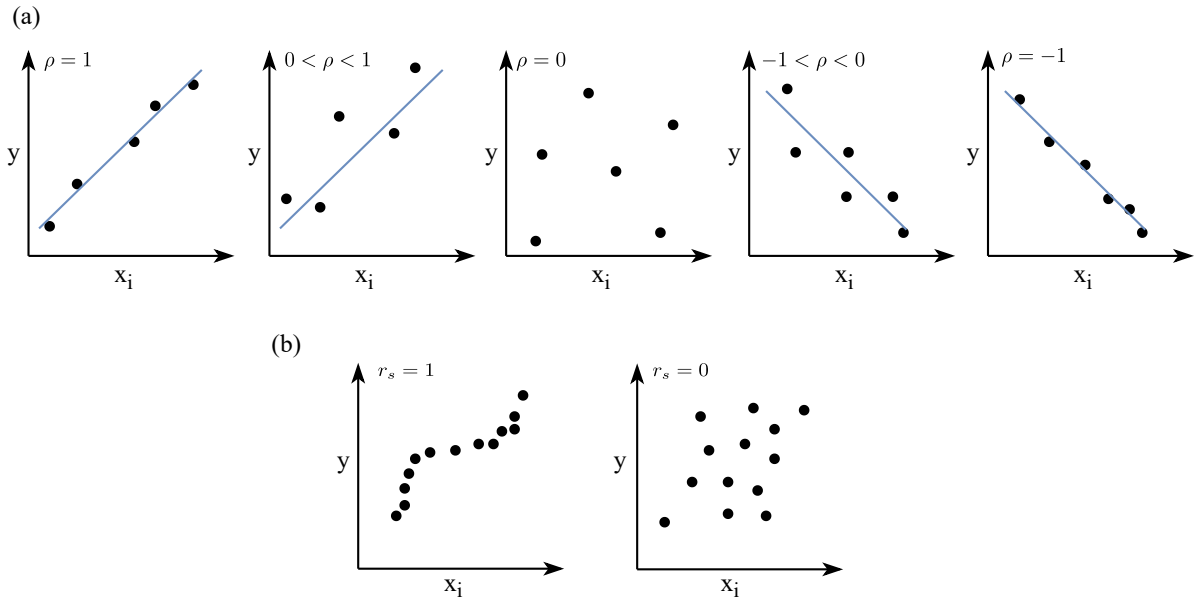


Figure 5.3: Different cases of: (a) PCC values (b) SRCC values.

Till now the discussion was focused on only bivariate regression, where response variables are related to only single design variables. In multiple linear, regression response (dependent) variable is expressed in terms of more than one design (independent) variables as follows,

$$y_j = \sum_i c_i x_i + \text{Intercept}, \tag{5.6}$$

where y_j is the j^{th} response variable, x_i is i^{th} design variable, c_i is the coefficient of the corresponding i^{th} design variable. In this work ordinary least square method (OLS) [65] is used to perform multiple linear regression analysis. This method gives best approximate of the regression line by minimizing square of error. Some important statistics used in the OLS regression analysis are as follows:

- **R-Squared**: This statistics shows how much the independent variable gMAS can be explained by changes in design variables. A good fit has higher value closer to 1.

- **Adjusted-R**: In case of linear regression, as more independent variables are added, the R-square value either stays same or increases. It never goes down. Fitted model can look more accurate even if the added independent variables are contributing very less. The adjusted R-squared penalizes R-squared by considering number of variables. Lower value indicates presence of non relevant independent variable.
- **t**: Standard error is the standard deviation of coefficient throughout the data points. t-value is the measurement of the precision with which this coefficient is measured. Higher **t** value signifies high significance of the coefficient.
- **P>|t|**: It is a measurement of how a coefficient is obtained from the considered model by chance. P-value of 0.45 for a design variable x_i denotes that there is 45% chance that design variable has no affect on the dependent variable.

In this chapter, aspects of DOE and sensitivity analysis are presented. The next chapter presents the study and results of an analysis where concepts discussed in this chapter and other previous chapters are used. All the topics discussed previously will be used to perform a successful analysis of an open hole tensile test specimen.

6 Numerical Experiments

This chapter presents the experimental setup and results of the work. At first the workflow of the study is discussed followed by initial DOE steps like design variable selection, model creation etc. Results of convergence analysis of a mesh used in analysis, is presented. The chapter ends with discussion of results.

6.1 Workflow

It is very beneficial to establish a clear project workflow while performing computer simulations and analysis using DOE. It helps in performing the analysis efficiently when there are multiple tools involved. In a project where various tasks are repetitive, a clear workflow helps in streamlining and automation. Figure 6.1 presents workflow and the tools adopted in this study.

The first step of designing an experiment is to define a clear objective. As discussed in the Chapter 1, the objective of the experiment is as follows,

Perform analysis to understand the influence of material parameters and notch geometry on an open hole specimen under tensile loading.

The next step is to identify design variables based on subject knowledge. More discussion on factor selection is presented in Section 6.3. Once the design variables are selected, a design matrix is constructed using a python[54] library called pyDOE2 [68]. The pyDOE2 library uses the LHD method to construct this design matrix. Based on this design matrix, LS-Dyna material cards and geometry cards are created. An open-source application called GMSH[20] is used for creating geometry. It is integrated into the workflow using its python application package interface (API). Once the geometry and material cards are created, they are combined to form different experimental runs. Each card is then sent to High Performance Computation (HPC) systems to run analysis. To run the analysis on HPC systems, sbatch scripts are used [16, 69]. After running all the simulations, different output factors are extracted from the LS-Dyna output file (d3plot). The data extraction is carried out using LS-Prepost macros and bash scripts [4]. To convert the extracted data into readable format, python libraries like pandas[49] and NumPy[46] are used. Regression and correlation analysis is performed on the extracted data. Python libraries like pandas, scikit-learn[64] and statsmodels[65] are used for this purpose. For better understanding, the data is visualized using matplotlib[38] and seaborn[66] python libraries. Based on observations from the results of regression and covariance analysis, the analysis conclusion is made in the end.

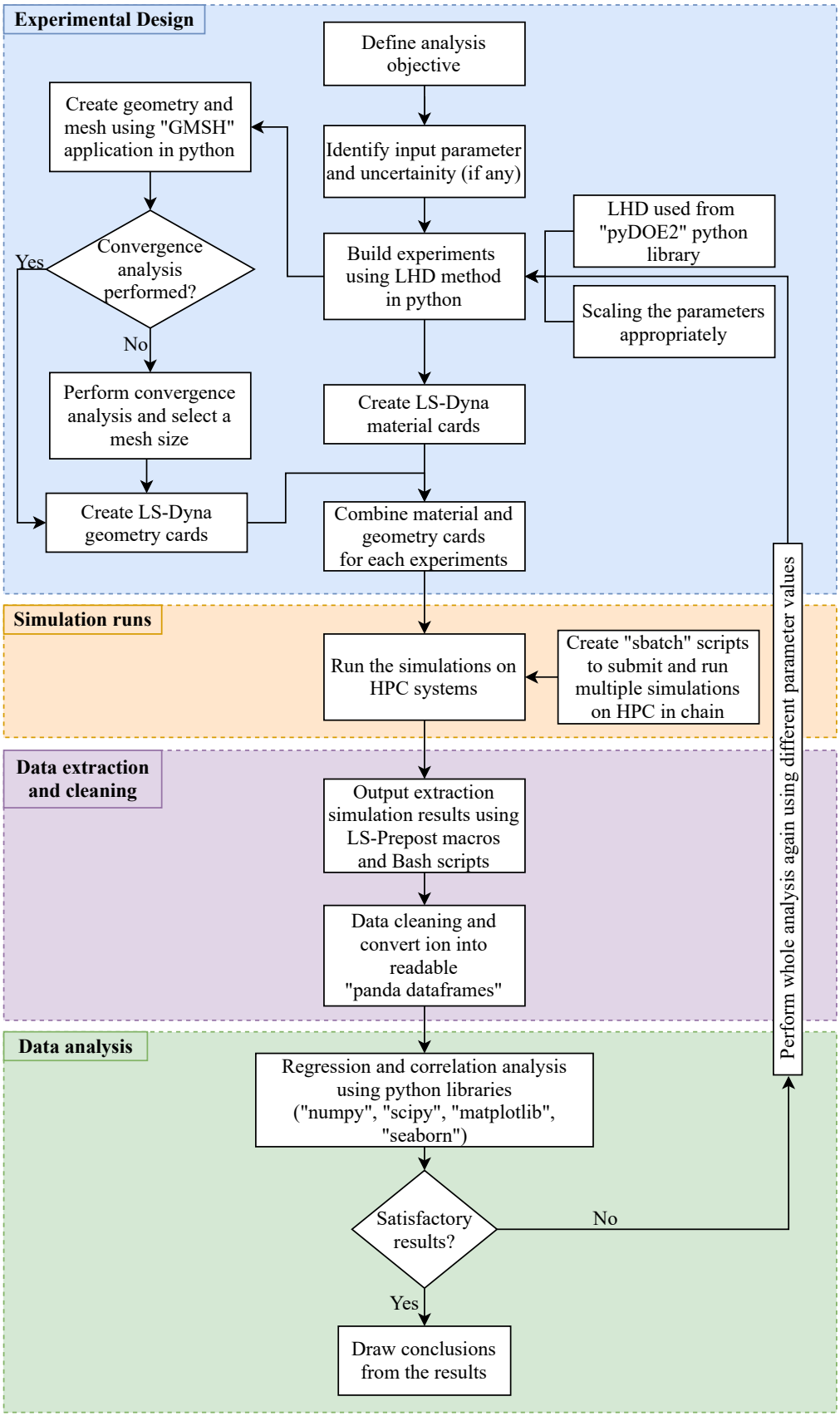


Figure 6.1: Project workflow.

The python scripts used in this study can be referred from Section A.4. These scripts can also be found at GitHub repository. In the next few sections, different aspects of geometry and parameters used in the analysis are discussed.

6.2 FEM Modelling

This section discusses details about specimen geometry and mesh used in the analysis. The specimen consists of an elliptic notch with variable radius Rx and a fixed radius Ry. Figure 6.2 shows the absolute dimensions of the specimen. It comprises six layers (plies) arranged in symmetric fashion, resulting in twelve layers as shown in Figure 6.3.

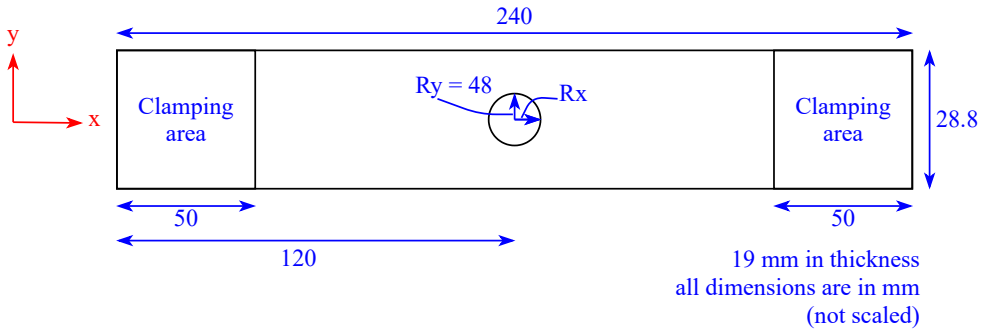


Figure 6.2: Specimen dimensions.

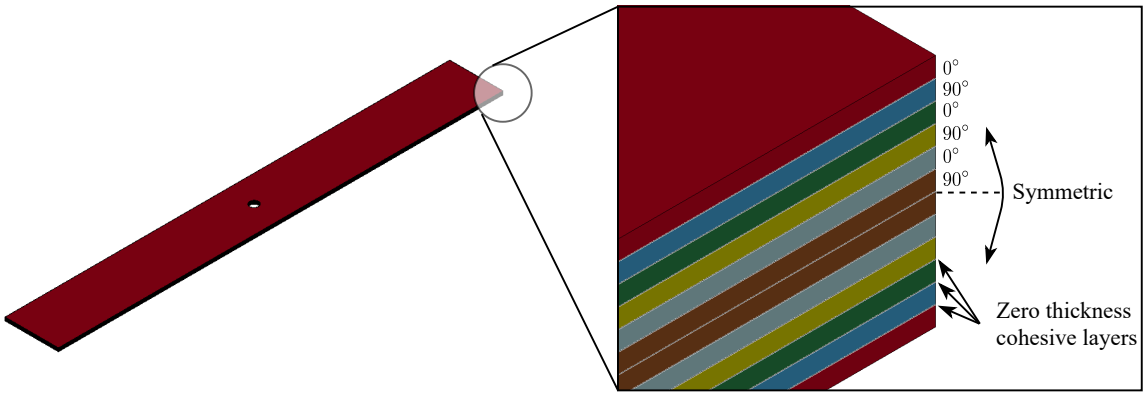


Figure 6.3: Different layers in the specimen.

All the layers on either side of the symmetric plane are oriented alternatively in 0° and 90° direction. The orientation of plies are set using vectors a and d as shown in Figure 6.4. As discussed in Subsection 3.2.1, axis-a denotes the fibre direction, axis-b is the in-plane transverse direction, and axis-c is the through-the-thickness direction. This option is set using $a1$, $a2$, $a3$, $d1$, $d2$, $d3$ in material card MAT_LAMINATED_FRACTURE_DAIMLER_PINHO as shown in Figure A.1. $a1$, $a2$, $a3$ are the components of vector a, while $d1$, $d2$, $d3$ are the components of vector d. These vectors are measured w.r.t global coordinates system. Co-

hesive element layers are modelled as zero thickness layers and are present in between each ply. Each ply is modelled with three elements in the thickness direction.

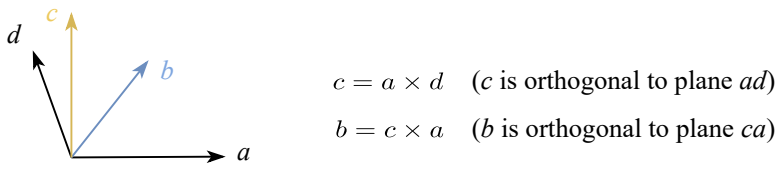


Figure 6.4: Material axes.

Mesh of the model is chosen by performing convergence analysis. Different meshes are created by changing the number of elements at the notch circumference. For the mesh convergence study, the specimen with an elliptic notch radius Rx=Ry=48mm is considered. Figure 6.5 shows different mesh configurations used for convergence study. Mesh convergence

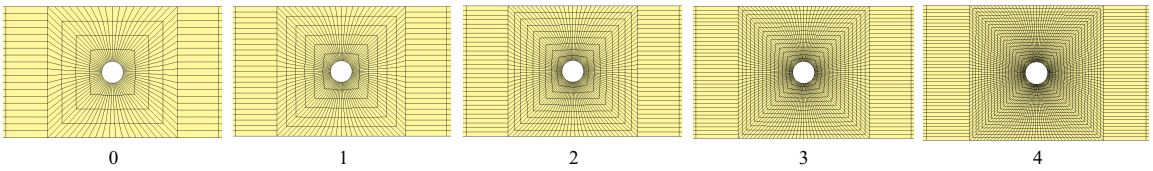


Figure 6.5: Different mesh configurations used in convergence study.

analysis is performed with all the configurations. Maximum load and stress concentration factor at the notch in one of the 0° plies, are used to compare the results. Figure 6.6(a) shows stress concentration factor at the notch. It can be observed that the difference in the stress concentration factor between consecutive meshes reduces to less than 1% for design point 3 and 4. Figure 6.6(b) shows the applied stress. The difference between maximum applied stress in design point 3 and 4 is less than 5%. No drastic difference in the results is observed after 3rd design point. Hence, mesh configuration of 3rd design point with 140 elements on the notch circumference, is used in the analysis. One side of the clamping area is constrained by fixing all the nodes on the top and bottom surface. The boundary conditions are defined on the nodes in clamping area. The load is applied to the nodes in clamping area present on the other side of the notch. Load is applied to the nodes on the top and bottom surface in the form of prescribed displacements.

6.3 Factor Selection

In this section, various design variables and response variables are discussed. Design variables are selected as per the discussion presented in Chapter 3 and Chapter 4. Most of the design variables are material properties of composite material and cohesive element layer. The design variables used in the analysis, along with their description and values, are presented in Table A.3. The base values of the design variables are referred from [63]. Figure 6.7

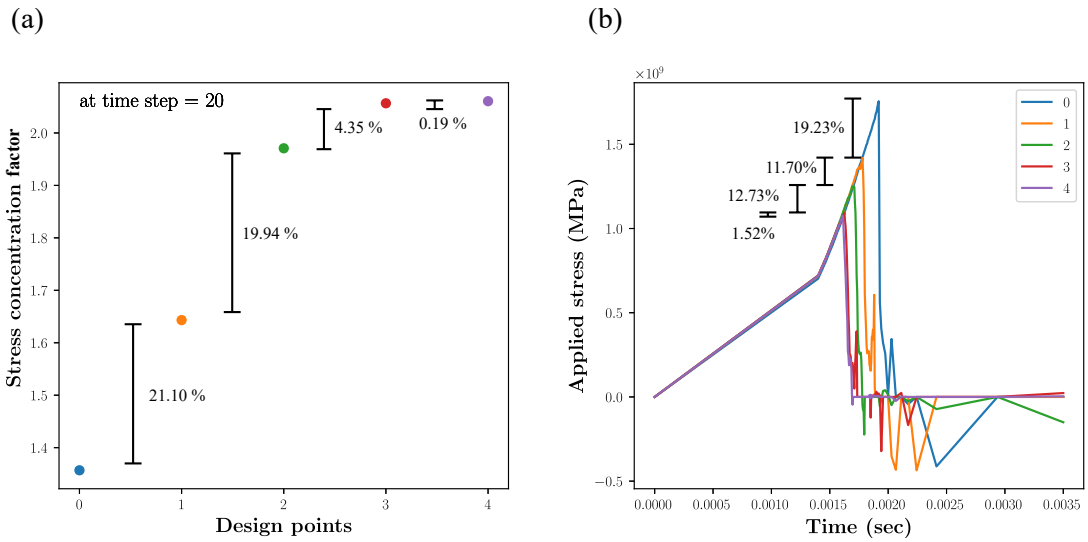


Figure 6.6: (a) Stress concentration factor at the notch (b) Applied stress, for a 0° ply.

shows the description of design variables like misalignment angle (MANG) and notch radius ratio (NOTCHR). Figure A.1 shows the composite material card used in the analysis. The

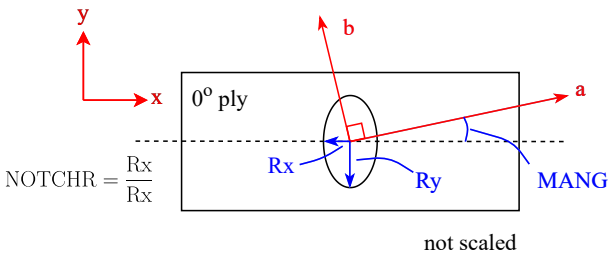


Figure 6.7: Design variables in the specimen.

material model is based on failure criteria from Pinho et al.[51, 52]. Discussion about this failure criteria can be found in Subsection 3.2.1. The definition of parameters is presented in Subsection A.1.1. The adhesive material is formulated using a zero thickness cohesive elements layer. Details about this formulation can be found in Subsection 3.2.2. Figure A.2 shows the material card used to model the cohesive element layer. As explained in Section 5.1, there are several possibilities of generating an LHD matrix. In this work, the optimum design matrix is generated in such a way that the unwanted correlation between each input parameter is minimized. Figure 6.8 shows the heatmap of input design variables depicting PCC values between them. It can be observed from the figure that the design variables have PCC values almost or equal to zero between them. Table A.3 and Table A.4 lists the definition of design variables (input parameters) and response variables (output parameters) used in the sensitivity analysis. All the response variable values are extracted at maximum applied stress. In the analysis, the longitudinal load-bearing capacity of composite degrades

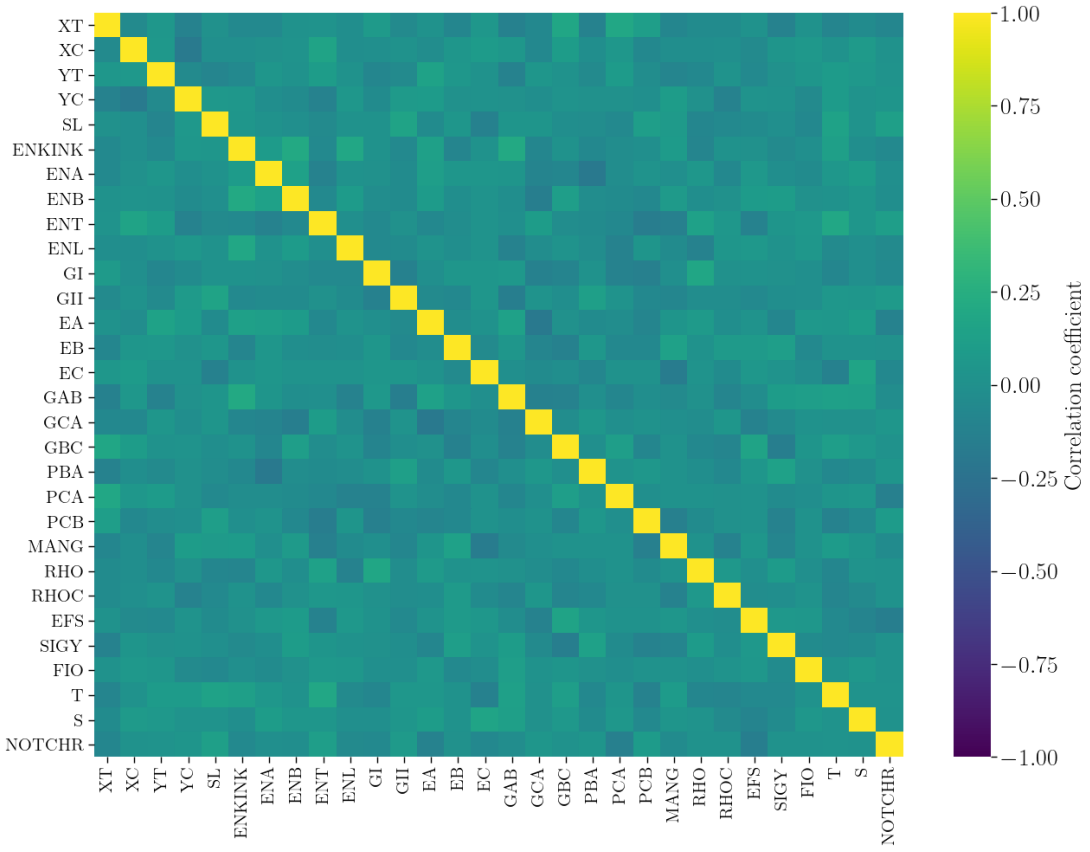


Figure 6.8: PCC heatmap of input design variables.

after this point. Hence it can be considered as a good point to study all the design and response variables.

In this section, the setup of the experiments is presented. Different aspects like workflow, python libraries, factor selection are discussed. The next step consists of performing analysis with this setup and extract the data. Once all the analyses are completed, the data extraction is done by creating LS-Prepost macros and running them using a bash script. The sample macros used to extract data can be found at **??**. As mentioned earlier, the extracted data is then cleaned and used to perform regression and sensitivity analysis. The next section presents the results of the work.

6.4 Results and Discussion

In this section, results of the sensitivity analysis performed on the database collected from analyses is presented. In this work Pearson's correlation coefficients (PCC), Spearman's ranked correlation coefficients (SRCC) and bivariate linear regression are used for preliminary assessment of the parameter influence. The later part of the discussion presents the results

of multiple linear regression. These results consist of the coefficient of the design variables of a hyper-plane equation.

Figure 6.9 shows heatmaps of PCC and SRCC values for input vs response variables, at maximum applied stress. For readability purpose, the heatmap is shown in compact form, where only response variables are displayed on the y-axis and only design variables are displayed on the x-axis. Each box contains PCC and SRCC values. For full heatmap, Figure A.3 and Figure A.4 can be referred. As discussed in Chapter 5, lower absolute values of PCC indicates a weaker influence of design variables. From the Figure 6.9, it can be observed that surprisingly very few design variables have significant correlation values (greater than 0.2 and less than -0.2). In this section regression results of only these strong correlations are presented. To generalize the interpretation of results, all the regression plots are normalized on both x and y axes using minimum and maximum values.

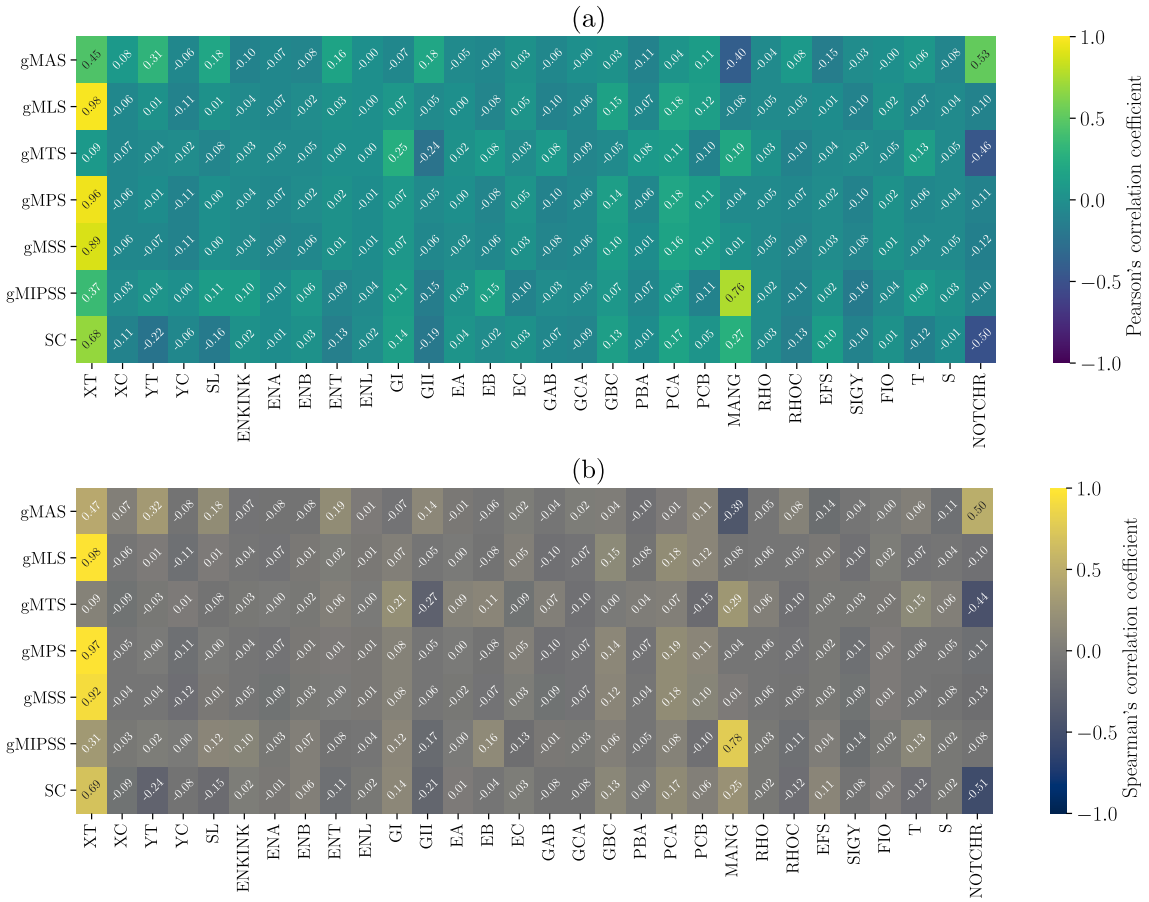


Figure 6.9: Heatmap plot of (a) PCC (b) SRCC, for input vs response variables.

- **Influence of Longitudinal Tensile Strength (XT)**
  The specimen in this experiment is loaded in the longitudinal direction. In such loading it is observed that tensile material properties of composite materials dominate all

the other properties. The same is observed here in the sensitivity and regression analysis. Tensile material property like longitudinal tensile strength (XT) has a stronger influence over all the stress values, including the stress concentration factor at the notch. Figure 6.10 shows regression plots and correlation values of XT vs response variables. Almost all the response variables show a positive correlation with XT. Maximum principal stresses(gMPS), maximum longitudinal stress(gMLS) and maximum shear stresses (gMSS) show very strong correlation with XT while maximum applied stress (gMAS), maximum in-plane shear stress (gMIPSS) and stress concentrations factor (SC) show moderate correlation. The result can be explained by the fact that as the strength of specimen increases, it can bear higher loads before it fails. This higher applied load generates higher stresses in the specimen.

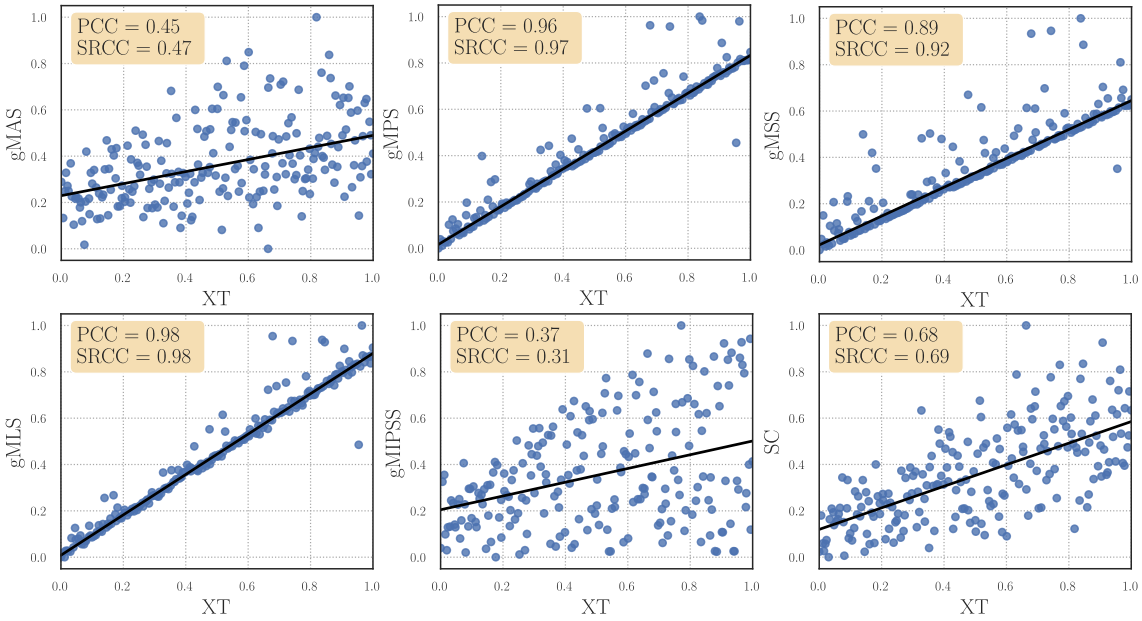


Figure 6.10: Regression plot: XT vs response variables.

- **Influence of Transverse Tensile Strength (YT)**
  Figure 6.11 shows regression plots for YT vs response variables where PCC values are more significant. Transverse tensile strength (YT) shows a moderate positive correlation with maximum applied stress (gMAS) while a negative correlation with stress concentration factor (SC) around the notch is observed. In cross-ply laminates, the longitudinal tensile load acts as a transverse load on 90° plies. The higher transverse strength prevents matrix cracking in 90° plies and increases its strength in transverse loads. Hence higher applied stress is observed as the YT increases. It is at the moment unclear why, YT shows almost no correlation with maximum principle stresses (gMPS). As SC is the ratio of gMPS and gMAS, the downward trend of the regression plot is observed for SC vs YT.

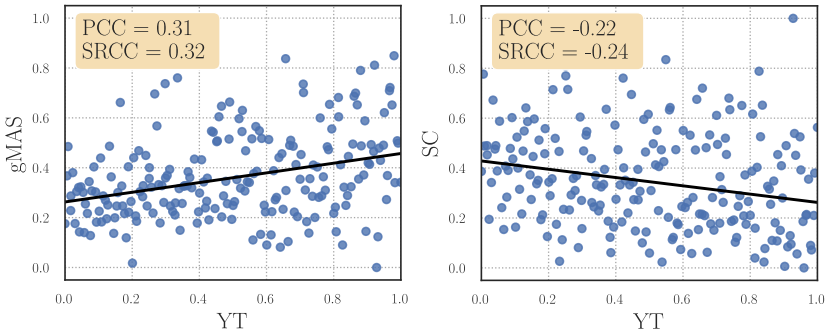


Figure 6.11: Regression plot: YT vs response variables.

- **Influence of Fracture Toughness of Cohesive Layers (GI,GII)**
  Regression plots for fracture toughness mode-I (GI) and mode-II (GII) are shown in Figure 6.12. The stronger correlation values are observed for GII as compared to GI. This indicates that in such loading conditions, mode-II (interlaminar shear) dominates mode-I (interlaminar tension). Figure 6.12(b) shows a positive correlation for maximum applied stresses (gMAS) while it shows negative correlations for maximum transverse stress (gMTS) and stress concentration factor (SC). In the case of mode-II behaviour, a higher GII value reduces the interlaminar shear failure. The interlaminar shear stresses are effectively distributed across all plies. Because of this, the specimen overall can withstand higher loads and stresses, indicated by a small positive correlation between GII and gMAS.

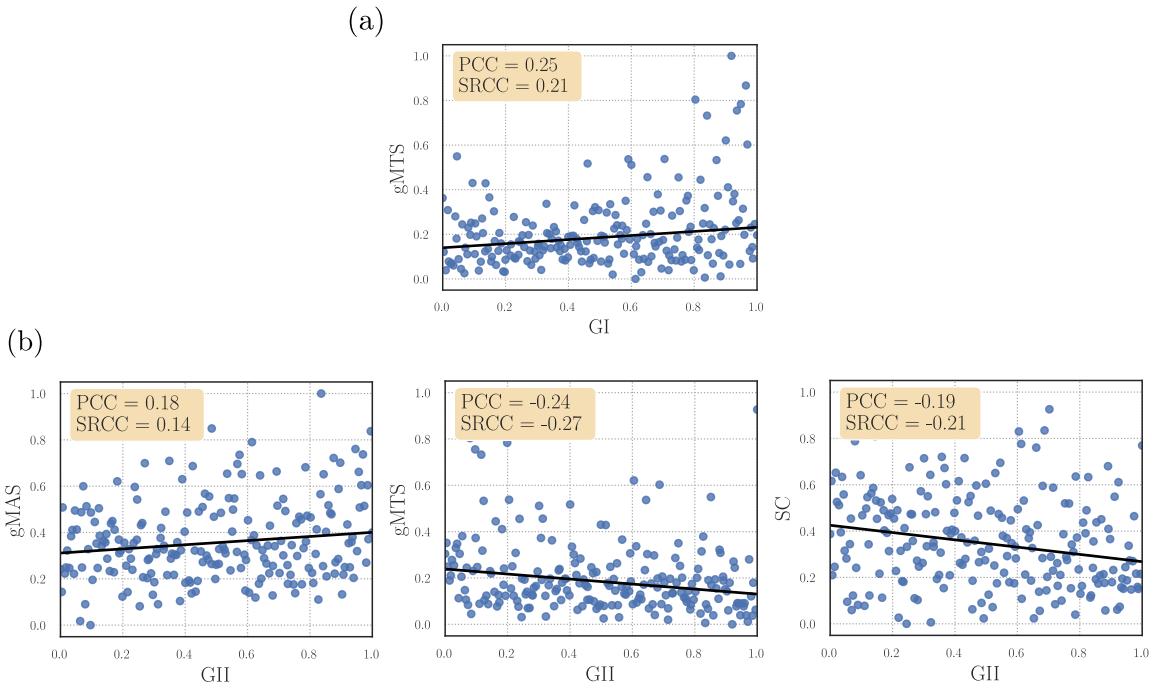


Figure 6.12: Regression plot: (a) GI vs response variables (b) GII vs response variables .

- **Influence of Misalignment Angle (MANG)**
  Misalignment angle is the small deviation in principle material axis orientation with respect to global x-axis or applied load direction as shown in Figure 6.7. Regression analysis plot along with PCC and SRC values for this parameter vs response variables are shown Figure 6.13. The strength of composite material depends upon

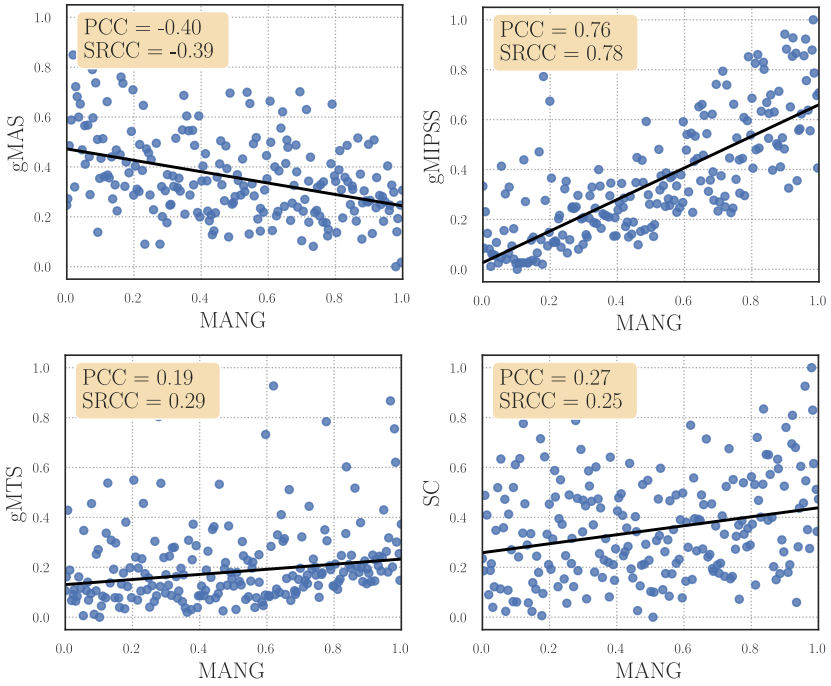


Figure 6.13: Regression plot: MANG vs response variables.

loading direction and fibre direction. As discussed in Chapter 3, during tensile loading along fibre direction, most amount of loads and stresses are taken by fibres. As the fibre orientation angle w.r.t loading axis increases (MANG, θ in Figure 6.7), load be taken by fibres decreases and shifts towards matrix material. As matrix material have lower strength, this decreases the strength and stiffness of composite structure. The same can be observed from a negative correlation between maximum applied stress (gMAS) and misalignment angle (MANG). In such cases, the in-plane shear stress and transverse stress increases, due to shear coupling. This is indicated by positive strong and weak correlation value between maximum in-plane shear stress (gMIPSS) and MANG, respectively. Weak positive correlation is observed between maximum shear stresses (gMTS) and MANG. SC is calculated as the ratio of maximum principle stresses (gMPS) and maximum applied stress (gMAS). In this case both have negative correlation with respect to MANG with gMAS having higher value (please refer to Figure 6.9 and Section A.3). Hence, as expected moderate positive correlation is observed between SC and MANG.

- **Influence of Notch Radius Ratio (NOTCHR)**
  The notch radius ratio is the ratio of radius in x-direction (longitudinal) and y-direction (transverse) of an elliptical notch as shown in Figure 6.7. Figure 6.14 shows regression plots of NOTCHR vs response variables. It is observed that the maximum applied stress (gMAS) shows positive correlation with respect to NOTCHR while maximum transverse stress (gMTS) and concentration factor (SC) shows negative correlation. Damages in OHT specimen originate from notch (or holes). As the notch radius ratio (NOTCHR) increases, the notch at the cross-section becomes more blunt. This helps in distributing stresses to matrix material and other fibres. As a result, the stress concentrations at the notch are decreased. This results in increased loading capacity of composite structure, showing higher maximum applied stress (gMAS).

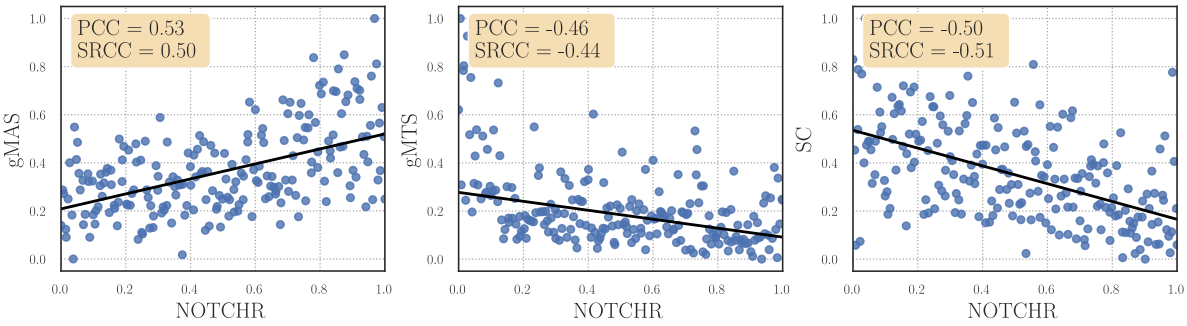


Figure 6.14: Regression plot: NOTCHR vs response variables.

In OHT specimen under uniaxial tensile test, material properties in the transverse direction (XC, YC) have very little correlation and influence over response variables. Moduli values (EA, EB, EC, GAB, GBC, GCA) and Poisson's ratio values (PBA, PCA, PCB) are also observed to have lower influence over response variables. Other parameters like the density of composite material (RHO) and adhesive material (RHOC), maximum effective strain for element layer failure (EFS), in-plane shear yield stress (SIGY), Peak traction in normal direction mode (T) and peak traction in tangential direction mode (S) shows similar trends of weak correlations with response variables. This can be observed in Figure 6.9. Toughness parameters (ENKINK, ENA, ENB, ENT, ENL) have almost no influence on the response variables. These findings were unexpected and suggest looking for an alternative approach to observe their effect. This can be due to not selecting an appropriate response variable that can show the clear influence of these variables like crack length in the case of toughness parameters. The selection process of response variables is also limited by FEM code implementation. For example, with this setup of the experiment, it is not possible to extract the crack length data in the specimen. Regression plots for all these design variables can be found in Section A.3.

As discussed in Section 5.2, ordinary least square method is used to perform multiple linear regression analysis in this work. Coefficient values obtained from multiple linear OLS regression is shown in Figure 6.15. The response variables will be largely affected by the changes

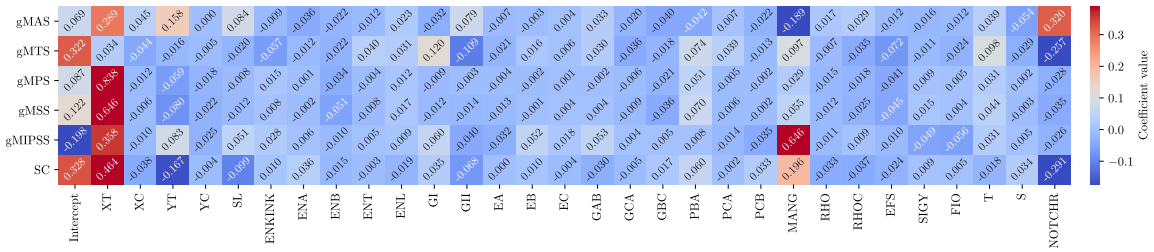


Figure 6.15: Multiple linear regression coefficients.

in design variables with higher coefficient values. The summary of the statistics generated from OLS multiple linear regression for maximum applied stress (gMAS) is presented in Figure 6.4. Here the value R-Squared = 0.780 shows that the fitted model explains 78% of the change in the dependent variable, gMAS. The results of multiple linear regression show a similar trend as the previously presented bivariate regression. It can be observed that most of the variables have lower t-value and higher P-value, signifying lower significance. These values indicate that these dependent variables (like ENKINK, ENA, ENB, EA, etc) can be dropped while fitting a linear model. Design variables like XT, XC, SL, GII, MANG and NOTCHR have higher t-values and lower P-values and hence can be considered in a multiple linear regression model. The results can be explained with the same reasoning, presented previously in bivariate analysis. The results of other response variables are presented in Subsection A.3.2.

Multiple Regression Summary

Results: Ordinary least squares

Model:	OLS	R-squared:	0.780
Dependent Variable:	gMAS	Adj. R-squared:	0.743
No. Observations:	210		
Df Model:	30		

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	0.0686	0.0636	1.0786	0.2822	-0.0569	0.1941
XT	0.2888	0.0236	12.2353	0.0000	0.2423	0.3354
XC	0.0454	0.0227	2.0040	0.0466	0.0007	0.0902
YT	0.1581	0.0230	6.8892	0.0000	0.1128	0.2034
YC	0.0002	0.0227	0.0104	0.9917	-0.0445	0.0449
SL	0.0840	0.0229	3.6691	0.0003	0.0388	0.1292

ENKINK	-0.0094	0.0238	-0.3950	0.6933	-0.0564	0.0376
ENA	-0.0356	0.0229	-1.5560	0.1215	-0.0808	0.0096
ENB	-0.0216	0.0229	-0.9447	0.3461	-0.0667	0.0235
ENT	-0.0119	0.0241	-0.4955	0.6209	-0.0595	0.0356
ENL	0.0234	0.0227	1.0341	0.3025	-0.0213	0.0681
GI	-0.0321	0.0233	-1.3783	0.1698	-0.0782	0.0139
GII	0.0788	0.0232	3.3944	0.0008	0.0330	0.1246
EA	-0.0070	0.0234	-0.2981	0.7660	-0.0531	0.0392
EB	-0.0026	0.0229	-0.1132	0.9100	-0.0477	0.0425
EC	0.0036	0.0229	0.1577	0.8749	-0.0417	0.0489
GAB	0.0334	0.0245	1.3618	0.1750	-0.0150	0.0818
GCA	-0.0201	0.0229	-0.8790	0.3806	-0.0653	0.0250
GBC	-0.0398	0.0238	-1.6749	0.0957	-0.0867	0.0071
PBA	-0.0422	0.0230	-1.8368	0.0679	-0.0875	0.0031
PCA	0.0074	0.0230	0.3204	0.7490	-0.0380	0.0527
PCB	-0.0220	0.0238	-0.9247	0.3563	-0.0690	0.0250
MANG	-0.1894	0.0236	-8.0231	0.0000	-0.2359	-0.1428
RHO	0.0173	0.0230	0.7524	0.4528	-0.0280	0.0626
RHOC	0.0291	0.0225	1.2926	0.1978	-0.0153	0.0735
EFS	-0.0122	0.0228	-0.5338	0.5941	-0.0572	0.0329
SIGY	-0.0162	0.0232	-0.6982	0.4860	-0.0619	0.0295
FIO	-0.0115	0.0221	-0.5231	0.6016	-0.0551	0.0320
T	0.0387	0.0237	1.6353	0.1037	-0.0080	0.0854
S	-0.0540	0.0227	-2.3735	0.0187	-0.0988	-0.0091
NOTCHR	0.3196	0.0225	14.1995	0.0000	0.2752	0.3641

In this section, results of regression and sensitivity analysis are discussed. Some design variables showed more influence on the response than others. Some results were explainable, and some results showed surprising behaviour. It is very important to keep in mind that composite material has a very complex failure mechanism. There is a possibility that results are not clear because of the influence of noise in the data. Hence, sometimes it is very difficult to find a clear correlation between different variables. There is also the possibility that the design variables might show a stronger correlation with some other response variables. Therefore, one should take the results as a grain of salt. It is very important to verify the results using physical experiments to assess the viability of results from virtual experiments. The verification with physical experiments is out of the scope of this work and hence it is skipped here.

In this chapter major work of this study is presented, starting from the setup of experiments to the results. Having discussed all the results, the next section discusses the conclusions of this work.

7 Conclusion

The present study analysed the failure behaviour of carbon fibre reinforced polymer open-hole tensile specimens by studying the effect of various parameters. This study started with the discussion of basic concepts of composite material. To summarize failure in composite materials, different failure mechanism and models were presented. To perform the study, design of experiments techniques were used. Design variables were selected to create an optimum (with minimum correlation) design matrix using the Latin hypercube design method. Sensitivity analysis measures like Pearson's correlation coefficient and Spearman's ranked correlation coefficient were used. The complete study followed a workflow created using scripting languages python and bash, different python libraries and software suite LS-Dyna and LS-PrePost.

This study has shown that material parameters that are in loading direction have a high influence over the response of the model. The type of the load (tensile or compressive, mode I or mode II) also determines, which parameters have a higher influence over the stresses developed in the composite materials. As a result, the effects of parameters like longitudinal tensile strength (XT), transverse tensile strength (YT), fracture toughness of a cohesive layer in shear mode (GII) were more pronounced than the other cohesive material parameters. Material orientation (MANG) also displayed high correlation values with response variables. This finding strongly suggests that material orientation with respect to loading direction also plays a significant role in composite structure analysis. Notched specimen with different elliptical radius were tested to study the effect of notch sizes. The sensitivity analysis results show very strong correlations between stress responses and notch radius ratio (NOTCHR). Lower correlations values between parameters like fracture toughness, Moduli, Poisson's ratio were observed. These findings suggest that either the response variables used to study were not suitable for these parameters or their influence got smeared due to the complex failure behaviour of composite materials. The present study extends the knowledge of the influence of various parameters on the failure of composite material and can serve as the base for future research. This work has also shown the effective use of interdisciplinary tools like the design of experiment and statistical analysis with physics simulation.

The current investigation is limited by the complex failure behaviour of composite material. There are many active ongoing research trying to establish an optimum failure model. Hence the results in this study should be used for preliminary analysis and must be considered cautiously. Future studies can consist of using different output responses and different material models that can capture the influence of parameters effectively. There are many possibilities in which this work can proceed further. Modern techniques like machine learn-

ing and artificial intelligence can be used to do clustering and classification analysis on the images or data points from analysis results. The natural progression of this work is to create an efficient meta-model that would help in predicting the response of the composite mater as close to physical testing. Deep learning methods can be used to create the meta-models using neural networks and generated data.

References

[1] Abish, John et al. Assessment of drilling-induced damage in CFRP under chilled air environment. *Materials and Manufacturing Processes* 33.12, Sept. 2018, pp. 1361–1368. issn: 1042-6914, 1532-2475. doi: 10.1080/10426914.2017.1415452.

[2] Abrate, Serge. Matrix cracking in laminated composites: A review. *Composites Engineering* 1.6, Jan. 1991, pp. 337–353. issn: 0961-9526. doi: 10.1016/0961-9526(91)90039-U. Visited on 10/04/2021,

[3] Andrew, J. Jefferson et al. Parameters influencing the impact response of fiber-reinforced polymer matrix composite materials: A critical review. *Composite Structures* 224, Sept. 2019, p. 111007. issn: 02638223. doi: 10.1016/j.compstruct.2019.111007.

[4] *Bash (Unix shell)*. Apr. 2021. url: https://en.wikipedia.org/w/index.php?title=Bash_(Unix_shell)&oldid=1018033668.

[5] Benzeggagh, M.L. and Kenane, M. Measurement of mixed-mode delamination fracture toughness of unidirectional glass/epoxy composites with mixed-mode bending apparatus. *Composites Science and Technology* 56.4, 1996, pp. 439–449. issn: 02663538. doi: 10.1016/0266-3538(96)00005-X.

[6] Bhat, Biliyar N., ed. *Aerospace Materials and Applications*. Reston ,VA: American Institute of Aeronautics and Astronautics, Inc., Jan. 2018. isbn: 978-1-62410-488-6 978-1-62410-489-3. doi: 10.2514/4.104893.

[7] Brauer, Delia et al. Degradable phosphate glass fiber reinforced polymer matrices: Mechanical properties and cell response. *Journal of materials science. Materials in medicine* 19, Feb. 2008, pp. 121–7. doi: 10.1007/s10856-007-3147-x.

[8] Camanho, P. P., Davila, C. G. and Moura, M. F. de. Numerical Simulation of Mixed-Mode Progressive Delamination in Composite Materials. *Journal of Composite Materials* 37.16, Aug. 2003, pp. 1415–1438. issn: 0021-9983. doi: 10.1177/0021998303034505.

[9] Camanho, P.P. et al. A finite fracture mechanics model for the prediction of the open-hole strength of composite laminates. *Composites Part A: Applied Science and Manufacturing* 43.8, Aug. 2012, pp. 1219–1225. issn: 1359835X. doi: 10.1016/j.compositesa.2012.03.004.

[10] *Can Carbon Fiber Replace Steel in the Automotive Industry? by KaganPittman*. url: https://www.engineering.com/story/can-carbon-fiber-replace-steel-in-the-automotive-industry, visited on 05/04/2021,

[11] Chang, Fu-Kuo and Chang, Kuo-Yen. A Progressive Damage Model for Laminated Composites Containing Stress Concentrations. *Journal of Composite Materials* 21.9, Sept. 1987, pp. 834–855. ISSN: 0021-9983. DOI: 10.1177/002199838702100904.

[12] Daniel, Isaac M. and Ishai, Ori. *Engineering mechanics of composite materials*. 2nd ed. New York: Oxford University Press, 2006. ISBN: 978-0-19-515097-1.

[13] Davies, G A O and Olsson, R. Impact on composite structures. *THE AERONAUTICAL JOURNAL*, 2004, p. 23.

[14] *DTDHandbook | Fundamentals of Damage Tolerance | Fracture Mechanics Fundamentals | Fracture Toughness - A Material Property*. URL: https://www.afgrow.net/applications/dtdhandbook/sections/page2_2_3.aspx, visited on 13/04/2021,

[15] Erçin, G H et al. Size effects on the tensile and compressive failure of notched composite laminates. *Composite Structures*, 2013, p. 9.

[16] *FEMSoftware - Compendium - Foswiki*. URL: https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/FEMSoftware, visited on 18/04/2021,

[17] *Figure 7.4 Micrographic matrix cracking in multidirectional laminate [20]*. URL: https://www.researchgate.net/figure/Micrographic-matrix-cracking-in-multidirectional-laminate-20_fig4_325024206.

[18] Flöck, J., Friedrich, K. and Yuan, Q. On the friction and wear behaviour of PAN- and pitch-carbon fiber reinforced PEEK composites. *Wear* 225-229, 1999, pp. 304–311. ISSN: 0043-1648. DOI: https://doi.org/10.1016/S0043-1648(99)00022-8.

[19] Gan, Yanjun et al. A comprehensive evaluation of various sensitivity analysis methods: A case study with a hydrological model. *Environmental Modelling & Software* 51, Jan. 2014, pp. 269–285. ISSN: 1364-8152. DOI: 10.1016/j.envsoft.2013.09.031.

[20] Geuzaine, Christophe and Remacle, Jean-François. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering* 79.11, 2009, pp. 1309–1331. ISSN: 1097-0207. DOI: https://doi.org/10.1002/nme.2579.

[21] Giunta, Anthony, Wojtkiewicz, Steven and Eldred, Michael. "Overview of Modern Design of Experiments Methods for Computational Simulations (Invited)". *41st Aerospace Sciences Meeting and Exhibit*. Reno, Nevada: American Institute of Aeronautics and Astronautics, Jan. 2003. ISBN: 978-1-62410-099-4. DOI: 10.2514/6.2003-649.

[22] Griffith, Alan Arnold and Taylor, Geoffrey Ingram. VI. The phenomena of rupture and flow in solids. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 221.582-593, Jan. 1921, pp. 163–198. DOI: 10.1098/rsta.1921.0006.

[23] HALLETT, Stephen R. et al. The open hole tensile test: a challenge for virtual testing of composites. *International Journal of Fracture* 158.2, Aug. 2009, pp. 169–181. ISSN: 1573-2673. DOI: 10.1007/s10704-009-9333-8.

[24] HAN, Geng et al. The failure mechanism of carbon fiber-reinforced composites under longitudinal compression considering the interface. *Science and Engineering of Composite Materials* 24.3, May 2017, pp. 429–437. ISSN: 2191-0359. DOI: 10.1515/secm-2015-0057.

[25] HASHIN, Z. and ROTEM, A. A Fatigue Failure Criterion for Fiber Reinforced Materials. *Journal of Composite Materials* 7.4, Oct. 1973, pp. 448–464. ISSN: 0021-9983. DOI: 10.1177/002199837300700404.

[26] HASHIN, Z. and SHTRIKMAN, S. A variational approach to the theory of the elastic behaviour of polycrystals. *Journal of the Mechanics and Physics of Solids* 10.4, Oct. 1962, pp. 343–352. ISSN: 0022-5096. DOI: 10.1016/0022-5096(62)90005-4.

[27] HELTON, J.C. et al. Survey of sampling-based methods for uncertainty and sensitivity analysis. *Reliability Engineering & System Safety* 91.10-11, Oct. 2006, pp. 1175–1209. ISSN: 09518320. DOI: 10.1016/j.ress.2005.11.017.

[28] HILL, R. A self-consistent mechanics of composite materials. *Journal of the Mechanics and Physics of Solids* 13.4, Aug. 1965, pp. 213–222. ISSN: 0022-5096. DOI: 10.1016/0022-5096(65)90010-4.

[29] HOFFMAN, Oscar. The Brittle Strength of Orthotropic Materials. *Journal of Composite Materials* 1.2, Apr. 1967, pp. 200–206. ISSN: 0021-9983. DOI: 10.1177/002199836700100210.

[30] HUI, Xinyu, XU, Yingjie and ZHANG, Weihong. An integrated modeling of the curing process and transverse tensile damage of unidirectional CFRP composites. *Composite Structures* 263, May 2021, p. 113681. ISSN: 0263-8223. DOI: 10.1016/j.compstruct.2021.113681. URL: https://www.sciencedirect.com/science/article/pii/S0263822321001422, visited on 02/05/2021,

[31] INAGAKI, MICHIO. "CHAPTER 4 - Carbon Fibers". *New Carbons - Control of Structure and Functions*. Ed. by INAGAKI, MICHIO. Oxford: Elsevier Science, 2000, pp. 82–123. ISBN: 978-0-08-043713-2. DOI: 10.1016/B978-008043713-2/50004-2.

[32] IOOSS, Bertrand and SALTELLI, Andrea. "Introduction to Sensitivity Analysis". *Handbook of Uncertainty Quantification*. Ed. by GHANEM, Roger, HIGDON, David and OWHADI, Houman. Cham: Springer International Publishing, 2017, pp. 1103–1122. ISBN: 978-3-319-12384-4 978-3-319-12385-1. DOI: 10.1007/978-3-319-12385-1_31.

[33] JOHNSON, M. E., MOORE, L. M. and YLVISAKER, D. Minimax and maximin distance designs. *Journal of Statistical Planning and Inference* 26.2, Oct. 1990, pp. 131–148. ISSN: 0378-3758. DOI: 10.1016/0378-3758(90)90122-B.

[34] Kar, Kamal K., ed. *Composite Materials*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017. ISBN: 978-3-662-49512-4 978-3-662-49514-8. DOI: 10.1007/978-3-662-49514-8. Visited on 03/04/2021,

[35] Karnik, S.R. et al. Delamination analysis in high speed drilling of carbon fiber reinforced plastics (CFRP) using artificial neural network model. *Materials & Design* 29.9, Oct. 2008, pp. 1768–1776. ISSN: 02613069. DOI: 10.1016/j.matdes.2008.03.014.

[36] *LS-DYNA | Livermore Software Technology Corp.* URL: http://www.lstc.com/products/ls-dyna.

[37] *Manuals*. URL: https://www.dynasupport.com/manuals, visited on 14/04/2021,

[38] *Matplotlib: Python plotting — Matplotlib 3.4.1 documentation*. URL: https://matplotlib.org/, visited on 18/04/2021,

[39] McKay, M. D., Beckman, R. J. and Conover, W. J. Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics* 21.2, May 1979, pp. 239–245. ISSN: 0040-1706. DOI: 10.1080/00401706.1979.10489755.

[40] *Mechanical Properties of Carbon Fibre Composite Materials*. URL: http://www.performance-composites.com/carbonfibre/mechanicalproperties_2.asp, visited on 05/04/2021,

[41] *Modeling Damage, Fatigue and Failure of Composite Materials*. Elsevier, 2016. ISBN: 978-1-78242-286-0. DOI: 10.1016/C2013-0-16521-X.

[42] Montgomery, Douglas C. Design and Analysis of Experiments, p. 749.

[43] Mori, T and Tanaka, K. Average stress in matrix and average elastic energy of materials with misfitting inclusions. *Acta Metallurgica* 21.5, May 1973, pp. 571–574. ISSN: 0001-6160. DOI: 10.1016/0001-6160(73)90064-3.

[44] Moure, M M. Damage evolution in open-hole laminated composite plates subjected to in-plane loads. *Composite Structures*, 2015, p. 10.

[45] Myers, Jerome L. and Well, A. *Research design and statistical analysis*. 2nd ed. Mahwah, N.J: Lawrence Erlbaum Associates, 2003. ISBN: 978-0-8058-4037-7.

[46] *NumPy*. URL: https://numpy.org/, visited on 18/04/2021,

[47] O'Higgins, R. M., McCarthy, M. A. and McCarthy, C. T. Comparison of open hole tension characteristics of high strength glass and carbon fibre-reinforced composite materials. *Composites Science and Technology*. Directions in Damage and Durability of Composite Materials, with regular papers 68.13, Oct. 2008, pp. 2770–2778. ISSN: 0266-3538. DOI: 10.1016/j.compscitech.2008.06.003.

[48] Omairey, Sadik L., Dunning, Peter D. and Sriramula, Srinivas. Development of an ABAQUS plugin tool for periodic RVE homogenisation. *Engineering with Computers* 35.2, Apr. 2019, pp. 567–577. ISSN: 0177-0667, 1435-5663. DOI: 10.1007/s00366-018-0616-4.

[49] *pandas - Python Data Analysis Library*. URL: https://pandas.pydata.org/, visited on 18/04/2021,

[50] Park, Soo-Jin and Seo, Min-Kang. "Types of Composites". *Interface Science and Technology*. Vol. 18. Elsevier, 2011, pp. 501–629. ISBN: 978-0-12-375049-5. DOI: 10.1016/B978-0-12-375049-5.00007-4.

[51] Pinho, S.T., Iannucci, L. and Robinson, P. Physically based failure models and criteria for laminated fibre-reinforced composites with emphasis on fibre kinking. Part II: FE implementation. *Composites Part A: Applied Science and Manufacturing* 37.5, May 2006, pp. 766–777. ISSN: 1359835X. DOI: 10.1016/j.compositesa.2005.06.008.

[52] Pinho, S.T., Iannucci, L. and Robinson, P. Physically-based failure models and criteria for laminated fibre-reinforced composites with emphasis on fibre kinking: Part I: Development. *Composites Part A: Applied Science and Manufacturing* 37.1, Jan. 2006, pp. 63–73. ISSN: 1359835X. DOI: 10.1016/j.compositesa.2005.04.016.

[53] Puck, A. and Schürmann, H. Failure analysis of FRP laminates by means of physically based phenomenological models. *Composites Science and Technology* 62.12, Sept. 2002, pp. 1633–1662. ISSN: 0266-3538. DOI: 10.1016/S0266-3538(01)00208-1.

[54] *Python*. URL: https://www.python.org/, visited on 27/02/2021,

[55] Qian, George and Mahdi, Adam. Sensitivity analysis methods in the biomedical sciences. *arXiv:2001.03965 [q-bio, stat]*, Jan. 2020,

[56] Rajak, Dipen Kumar et al. Recent progress of reinforcement materials: a comprehensive overview of composite materials. *Journal of Materials Research and Technology* 8.6, Nov. 2019, pp. 6354–6374. ISSN: 22387854. DOI: 10.1016/j.jmrt.2019.09.068.

[57] Reddy, J N. An Introduction to Continuum Mechanics, Second Edition, p. 480.

[58] Reuss, A. Berechnung der Fließgrenze von Mischkristallen auf Grund der Plastizitätsbedingung für Einkristalle .. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 9.1, 1929, pp. 49–58. ISSN: 1521-4001. DOI: https://doi.org/10.1002/zamm.19290090104.

[59] Rodgers, Joseph Lee and Nicewander, W Alan. Thirteen Ways to Look at the Correlation Coefficient, p. 11.

[60] Rybicki, E. F. and Kanninen, M. F. A finite element calculation of stress intensity factors by a modified crack closure integral. *Engineering Fracture Mechanics* 9.4, Jan. 1977, pp. 931–938. ISSN: 0013-7944. DOI: 10.1016/0013-7944(77)90013-3.

[61] Sacks, Jerome, Schiller, Susannah B. and Welch, William J. Designs for Computer Experiments. *Technometrics* 31.1, 1989, pp. 41–47. ISSN: 0040-1706. DOI: 10.2307/1270363.

[62] Santner, Thomas J., Williams, Brian J. and Notz, William I. *The Design and Analysis of Computer Experiments*. Springer Series in Statistics. New York, NY: Springer New York, 2018. ISBN: 978-1-4939-8845-7 978-1-4939-8847-1. DOI: 10.1007/978-1-4939-8847-1.

[63] Schmidt, Benjamin et al. Analysis of process-induced damage in remote laser cut carbon fibre reinforced polymers. *arXiv:2006.16115 [cond-mat, physics:physics]*, June 2020, URL: http://arxiv.org/abs/2006.16115.

[64] *scikit-learn: machine learning in Python — scikit-learn 0.24.1 documentation*. URL: https://scikit-learn.org/stable/index.html, visited on 18/04/2021,

[65] Seabold, Skipper and Perktold, Josef. "statsmodels: Econometric and statistical modeling with python". *9th Python in Science Conference*. 2010.

[66] *seaborn: statistical data visualization — seaborn 0.11.1 documentation*. URL: https://seaborn.pydata.org/#, visited on 18/04/2021,

[67] Shewry, M. C. and Wynn, H. P. Maximum entropy sampling. *Journal of Applied Statistics* 14.2, Jan. 1987, pp. 165–170. ISSN: 0266-4763. DOI: 10.1080/02664768700000020.

[68] Sjoegren, Rickard. *pyDOE2: Design of experiments for Python*. URL: https://github.com/clicumu/pyDOE2, visited on 18/04/2021,

[69] *Slurm Workload Manager - sbatch*. URL: https://slurm.schedmd.com/sbatch.html, visited on 18/04/2021,

[70] Sun, C. T and Jih, C. J. On strain energy release rates for interfacial cracks in bimaterial media. *Engineering Fracture Mechanics* 28.1, Jan. 1987, pp. 13–20. ISSN: 0013-7944. DOI: 10.1016/0013-7944(87)90115-9.

[71] Suresh, Masa et al. Fractographic Analysis of Tensile Failures of Aerospace Grade Composites. *Materials Research* 15, Dec. 2012, pp. 990–997. DOI: 10.1590/S1516-14392012005000141.

[72] *The Art of Engineering: The World's Largest Jet Engine Shows Off Composite Curves | GE News*. URL: http://www.ge.com/news/reports/the-art-of-engineering-the-worlds-largest-jet-engine-shows-off-composite-curves, visited on 25/02/2021,

[73] Tsai, Stephen W. and Wu, Edward M. A General Theory of Strength for Anisotropic Materials. *Journal of Composite Materials* 5.1, Jan. 1971, pp. 58–80. ISSN: 0021-9983. DOI: 10.1177/002199837100500106.

[74] Voigt, Woldemar. *Theoretische Studien über die Elasticitätsverhältnisse der Krystalle.* 1887.

[75] Whitney, J.M. and Nuismer, R.J. Stress Fracture Criteria for Laminated Composites Containing Stress Concentrations. *Journal of Composite Materials* 8.3, July 1974, pp. 253–265. ISSN: 0021-9983. DOI: 10.1177/002199837400800303.

[76] Xiao, Mengli. Tensile failure analysis and residual strength prediction of CFRP laminates with open hole, p. 23.

[77] Xiao, Mengli et al. Tensile failure analysis and residual strength prediction of CFRP laminates with open hole. *Composites Part B: Engineering* 126, Oct. 2017, pp. 49–59. ISSN: 13598368. DOI: 10.1016/j.compositesb.2017.05.082.

[78] Yangyang, QIAO and Yuanli, BAI. A Review on Failure Modeling Methods of Fiber Reinforced Polymer Matrix Composites. 9.1, 2018, p. 10.

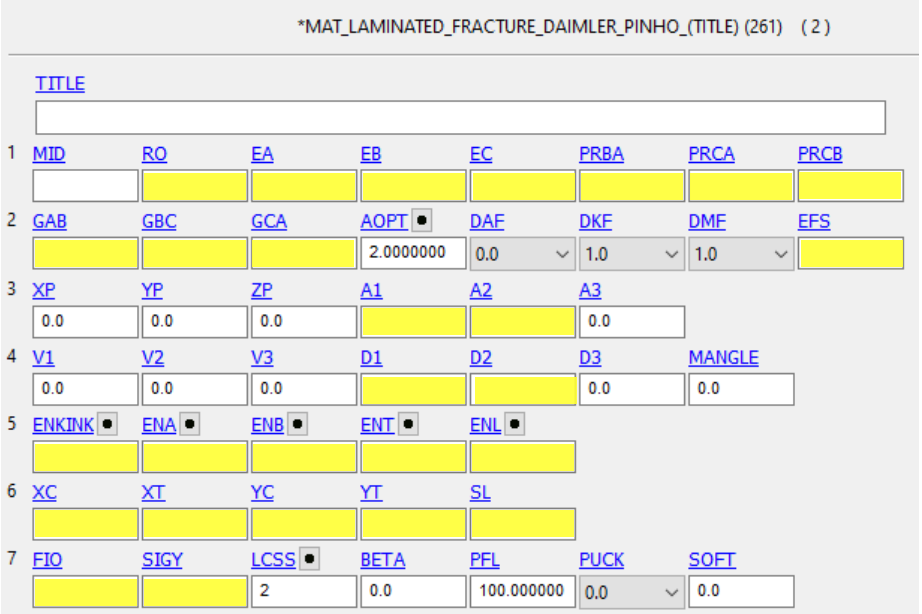
[79] Zanjani, Jamal Seyyed Monfared et al. Nano-engineered design and manufacturing of high-performance epoxy matrix composites with carbon fiber/selectively integrated graphene as multi-scale reinforcements. *RSC Advances* 6.12, Jan. 2016, pp. 9495–9506. ISSN: 2046-2069. DOI: 10.1039/C5RA23665G.

[80] Zhang, Di, Zheng, Xitao and Wu, Tianchi. Damage characteristics of open-hole laminated composites subjected to longitudinal loads. *Composite Structures* 230, Dec. 2019, p. 111474. ISSN: 02638223. DOI: 10.1016/j.compstruct.2019.111474.

A Appendix

A.1 Material Cards

A.1.1 Composite Material Card



Note- Options highlighted with yellow color are the design variables.

Figure A.1: Composite material card.

Parameter	Description
Card 1	
MID	Material ID
RO	Mass Density
EA	Young's modulus in longitudinal direction
EB	Young's modulus in transverse direction
EC	Young's modulus in through thickness direction
PBA	Poisson's ratio in ba
PCA	Poisson's ratio in ca
PCB	Poisson's ratio in cb
Card 2	
GAB	Shear modulus plane ab
GCA	Shear modulus plane ca
GBC	Shear modulus plane bc
AOPT	Material axes option
DAF	Integration point failure flag - longitudinal tensile fibre failure
DKF	Integration point failure flag - longitudinal compressive fibre failure

DMF	Integration point failure flag - transverse matrix failure
EFS	Maximum effective strain for element layer failure
Card 3	
XP, YP, ZP	Coordinates of point **p** for AOPT = 1 and 4
A1, A2, A3	Components of vector **a** for AOPT = 2
Card 4	
V1, V2, V3	Components of vector **v** for AOPT = 3
D1, D2, D3	Components of vector **d** for AOPT = 2
MANGLE	Material angle in degrees for AOPT = 0
Card 5	
ENKINK	Fracture toughness for longitudinal (fibre) compressive failure mode
ENA	Fracture toughness for longitudinal (fibre) tensile failure mode
ENB	Fracture toughness for intralaminar matrix tensile failure
ENT	Fracture toughness for intralaminar matrix transverse shear failure
ENL	Fracture toughness for intralaminar matrix longitudinal shear failure
Card 6	
XC	Longitudinal compressive strength, a-axis (positive value)
XT	Longitudinal tensile strength, a-axis
YC	Transverse compressive strength, b-axis (positive value)
YT	Transverse tensile strength, b-axis
SL	Longitudinal shear strength
Card 7	
FIO	Fracture angle in pure transverse compression
SIGY	In-plane shear yield stress
LCSS	Load curve ID or Table ID
BETA	Hardening parameter for in-plane shear plasticity
PFL	Percentage of shell or tshell layers which must fail until crashfront is initiated
PUCK	Flag for evaluation and post-processing of Puck's inter-fibrefailure criterion
SOFT	Softening reduction factor for material strength in crashfront elements

Table A.1: Composite material card.

A.1.2 Cohesive Element Material Card

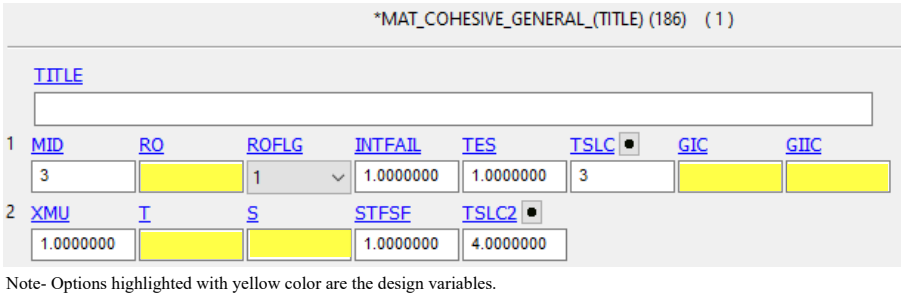


Figure A.2: Cohesive element card.

Parameter	Description
Card 1	
MID	Material ID
RO	Mass Density
ROFLG	Flag for whether density is specified per unit area or volume
INTFAIL	Number of integration points required for a cohesive element to be deleted
TES	Type of effective separation parameter
TSLC	Normalized traction-separation load curve ID
GIC	Fracture toughness / energy release rate G_{I}^{c} mode I
GIIC	Fracture toughness / energy release rate G_{II}^{c} mode II
Card 2	
XMU	Exponent appearing in the the Benzeggagh-Kenane failure criterion
T	Peak traction in normal direction (mode I)
S	Peak traction in tangential direction (mode II)
STFSF	Penetration stiffness multiplier for compression
TSLC2	Normalized traction-separation load curve ID for Mode II

Table A.2: Cohesive material card.

A.2 Experiment Parameters

A.2.1 Design Variables

Design variable	Description (Material)	Value
XT	Longitudinal tensile strength (M)	$1.79 \times 10^9 \pm 40\%$ N/m^2
XC	Longitudinal compressive strength (M)	$6.5 \times 10^9 \pm 40\%$ N/m^2
YT	Transverse tensile strength (M)	$4.2 \times 10^7 \pm 40\%$ N/m^2
YC	Transverse compressive strength(M)	$1.3 \times 10^8 \pm 40\%$ N/m^2
SL	Longitudinal shear strength (M)	$7.4 \times 10^7 \pm 40\%$ N/m^2
ENKINK	Fracture toughness for longitudinal (fibre) (M) compressive failure mode	$4.0 \times 10^7 \pm 80\%$ N/m
ENA	Fracture toughness for longitudinal (fibre)(M) tensile failure mode	$4.5 \times 10^7 \pm 80\%$ N/m
ENB	Fracture toughness for intralaminar (M) Matrix tensile failure	$2.5 \times 10^3 \pm 80\%$ N/m
ENT	Fracture toughness for intralaminar (M) Matrix transverse shear failure	$4.0 \times 10^3 \pm 80\%$ N/m
ENL	Fracture toughness for intralaminar (M) longitudinal shear failure	$4.0 \times 10^3 \pm 80\%$ N/m
GI	Fracture toughness mode I (C)	$500 \pm 80\%$ N/m
GII	Fracture toughness mode II (C)	$800 \pm 80\%$ N/m
EA	Young's modulus in longitudinal direction (M)	$1.21 \times 10^{11} \pm 20\%$ N/m^2

EB	Young's modulus in transverse direction (M)	$9.24 \times 10^9 \pm 20\%$ N/m^2
EC	Young's modulus in through thickness (M) direction	$9.24 \times 10^9 \pm 20\%$ N/m^2
GAB	Shear modulus plane ab (M)	$1.35 \times 10^{10} \pm 20\%$ N/m^2
GCA	Shear modulus plane ca (M)	$1.35 \times 10^{10} \pm 20\%$ N/m^2
GBC	Shear modulus plane bc (M)	$2.92 \times 10^9 \pm 20\%$ N/m^2
PBA	Poisson's ratio in ba (M)	$0.015 \pm 20\%$
PCA	Poisson's ratio in ca (M)	$0.015 \pm 20\%$
PCB	Poisson's ratio in cb (M)	$0.37 \pm 20\%$
MANG	Misalignment angle of material orientation (M)	$0°$ to $10°$ degrees
RHO	Mass density of composite material (M)	$1530 \pm 20\%$ kg/m^3
RHOC	Mass density of cohesive layer (C)	$6.50 \pm 20\%$ kg/m^3
EFS	Maximum effective strain for element layer failure (M)	-0.4 to -0.1
SIGY	In-plane shear yield stress (M)	$0.75 \times 10^6 \pm 40\%$ N/m^2
FIO	Fracture angle in pure transverse compression (M)	$51°$ to $55°$ degrees
T	Peak traction in normal direction mode (C)	3.7×10^7 to 4.37×10^7 N
S	Peak traction in tangential direction mode (C)	3.7×10^7 to 4.37×10^7 N
NOTCHR	Notch radius ratio Rx/Ry with Ry=48mm (C)	0.0104 to 0.2083

M: Composite material parameter, C: Cohesive material parameter.

Table A.3: Design variables

A.2.2 Response Variables.

Output variable	Description
gMAS	Maximum applied stress in x-direction before failure
gMLS	Maximum longitudinal stress globally at maximum gMAS
gMTS	Maximum transverse stress globally at maximum gMAS
gMPS	Maximum principle stress globally at maximum gMAS
gMSS	Maximum shear stress globally at maximum gMAS
gMIPSS	Maximum in-plane shear stress globally at maximum gMAS
SC	Stress concentration factor at the notch at maximum gMAS

Table A.4: Response variables.

A.3 Result Plots

A.3.1 Bivariate Regression

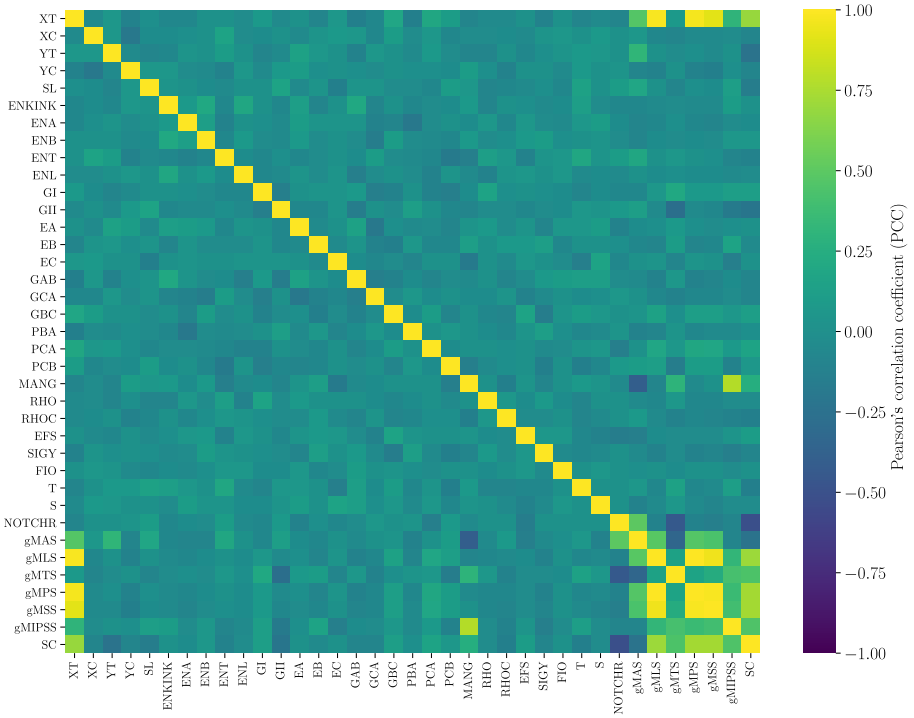


Figure A.3: Full heatmap plot of PCC values of design variables and response variables.

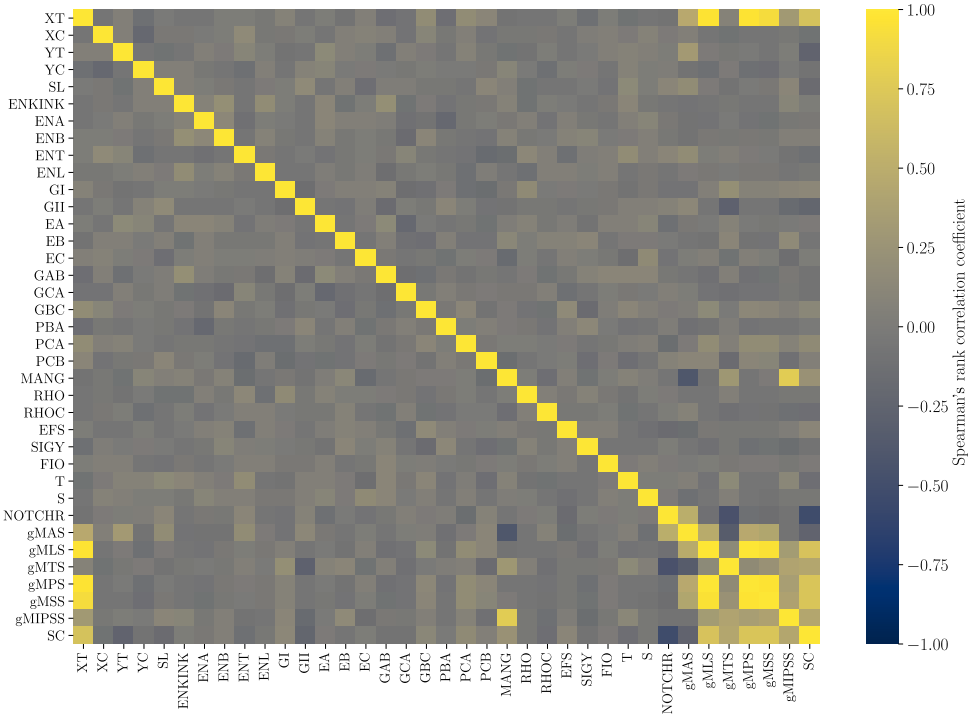


Figure A.4: Full heatmap plot of SRCC values of design variables and response variables.

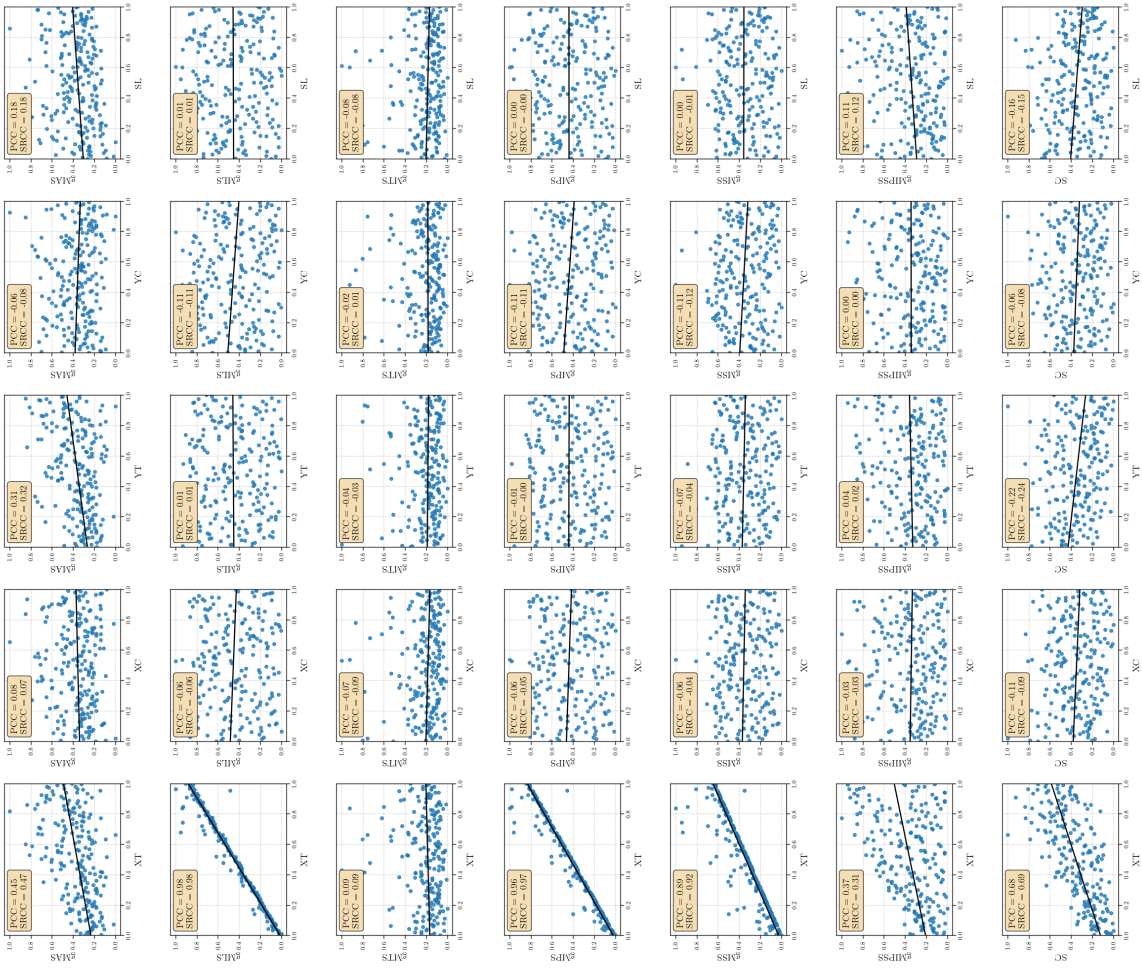


Figure A.5: Regression plots of design variables vs response variables.

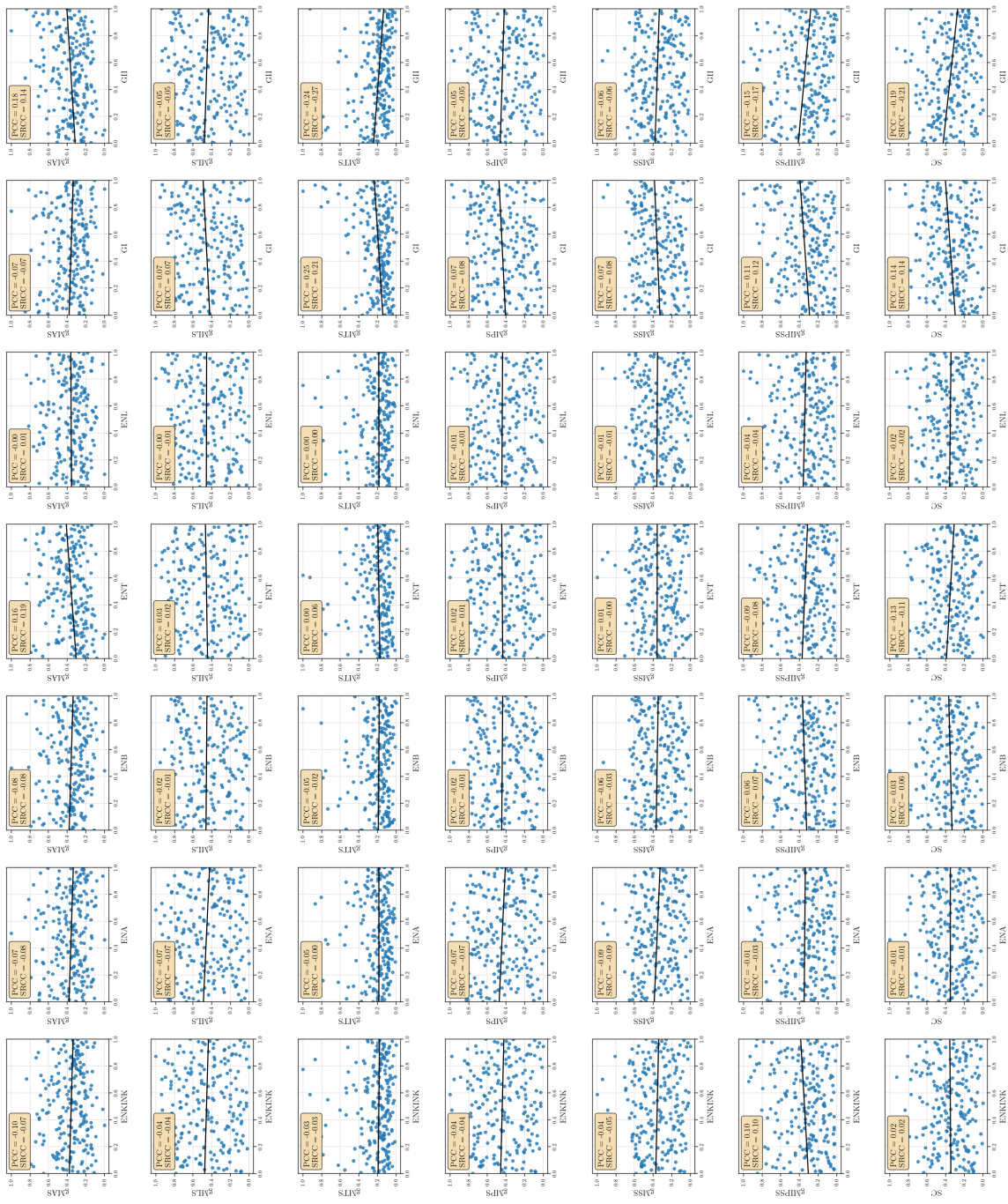


Figure A.6: Regression plots of design variables vs response variables.

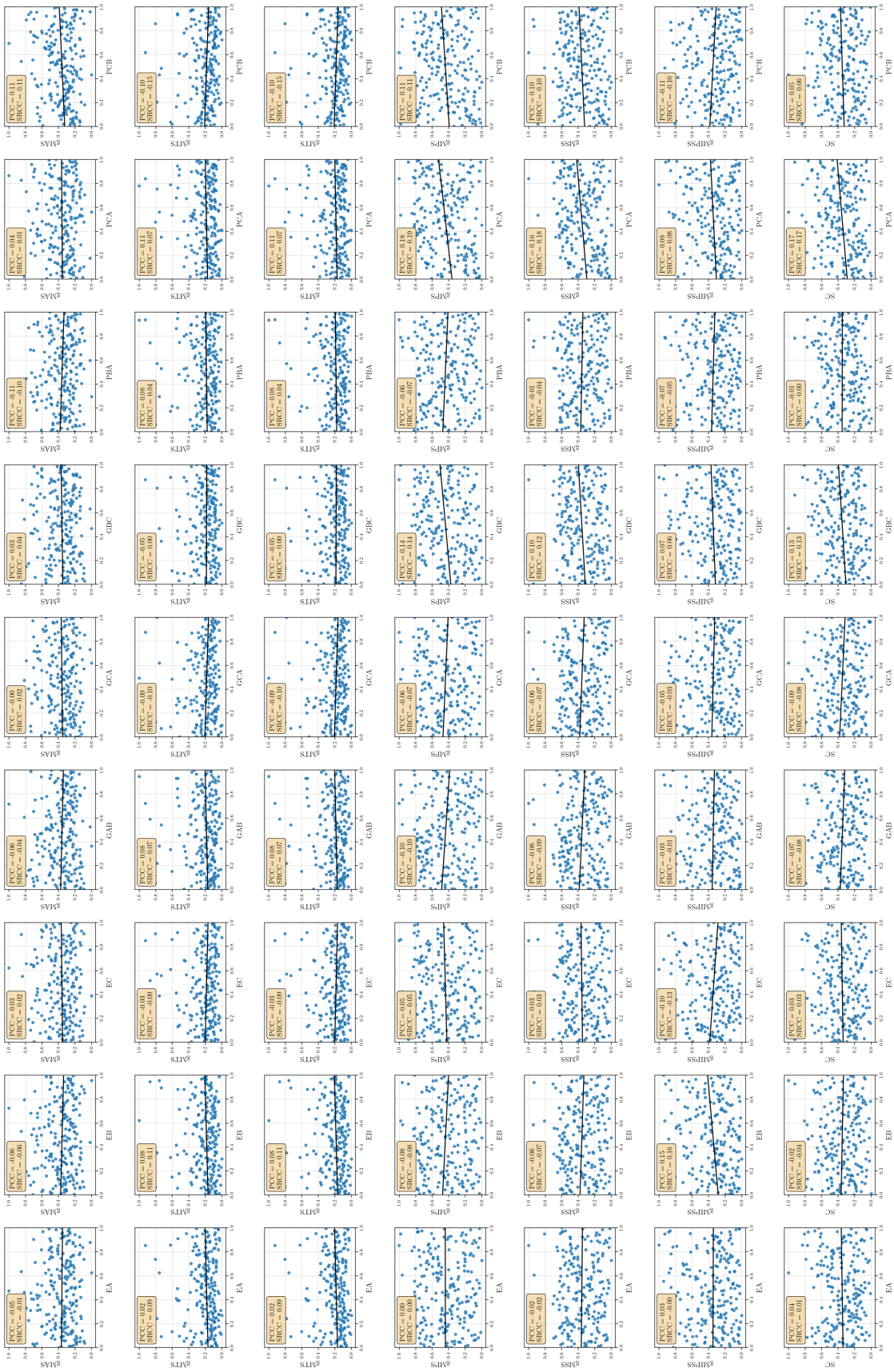


Figure A.7: Regression plots of design variables vs response variables.

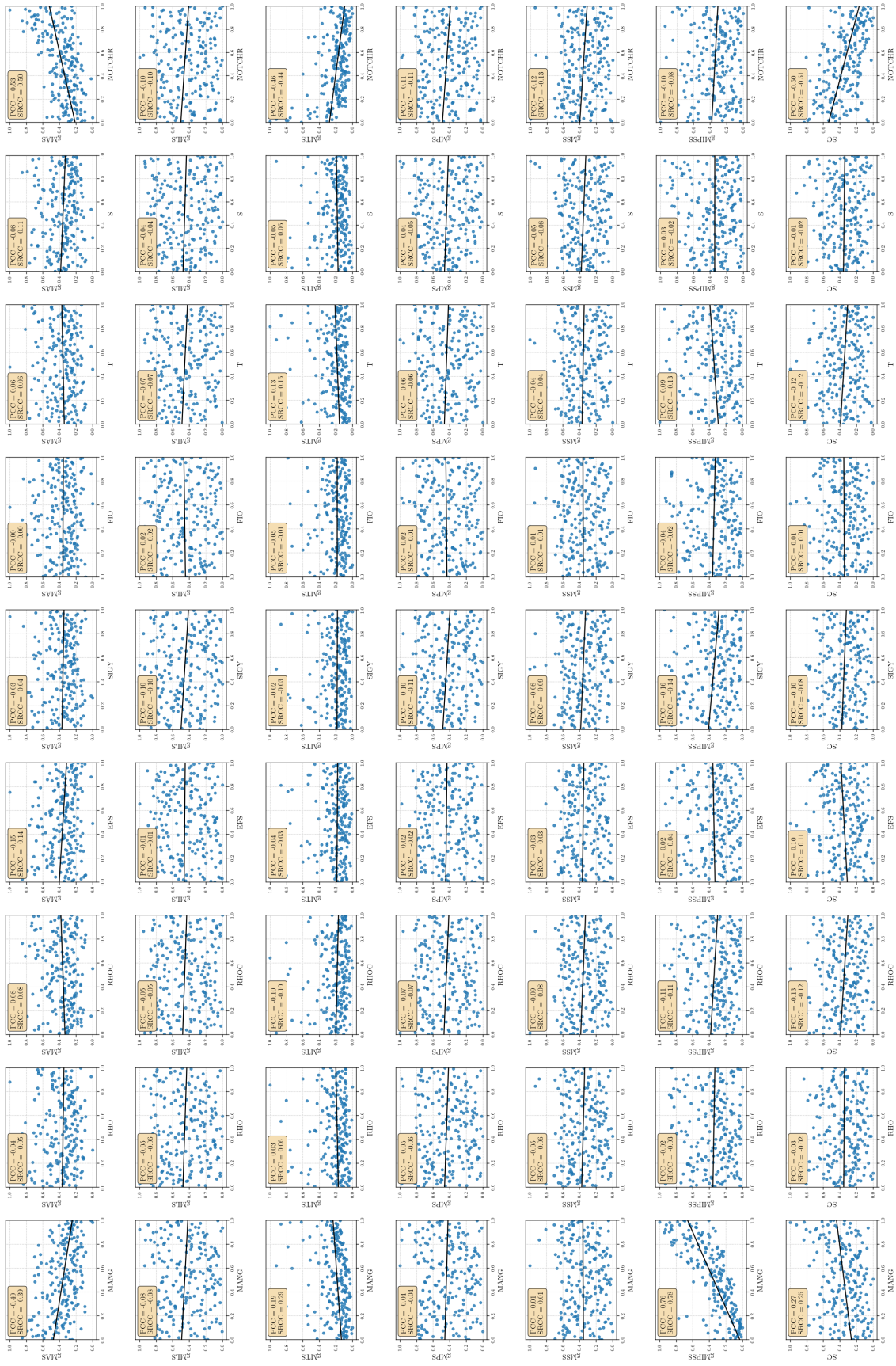


Figure A.8: Regression plots of design variables vs response variables.

A.3.2 Multiple Regression

Multiple Regression Summary

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Results: Ordinary least squares

Model:	OLS	R-squared:	0.780
Dependent Variable:	gMAS	Adj. R-squared:	0.743
No. Observations:	210		
Df Model:	30		

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	0.0686	0.0636	1.0786	0.2822	-0.0569	0.1941
XT	0.2888	0.0236	12.2353	0.0000	0.2423	0.3354
XC	0.0454	0.0227	2.0040	0.0466	0.0007	0.0902
YT	0.1581	0.0230	6.8892	0.0000	0.1128	0.2034
YC	0.0002	0.0227	0.0104	0.9917	-0.0445	0.0449
SL	0.0840	0.0229	3.6691	0.0003	0.0388	0.1292
ENKINK	-0.0094	0.0238	-0.3950	0.6933	-0.0564	0.0376
ENA	-0.0356	0.0229	-1.5560	0.1215	-0.0808	0.0096
ENB	-0.0216	0.0229	-0.9447	0.3461	-0.0667	0.0235
ENT	-0.0119	0.0241	-0.4955	0.6209	-0.0595	0.0356
ENL	0.0234	0.0227	1.0341	0.3025	-0.0213	0.0681
GI	-0.0321	0.0233	-1.3783	0.1698	-0.0782	0.0139
GII	0.0788	0.0232	3.3944	0.0008	0.0330	0.1246
EA	-0.0070	0.0234	-0.2981	0.7660	-0.0531	0.0392
EB	-0.0026	0.0229	-0.1132	0.9100	-0.0477	0.0425
EC	0.0036	0.0229	0.1577	0.8749	-0.0417	0.0489
GAB	0.0334	0.0245	1.3618	0.1750	-0.0150	0.0818
GCA	-0.0201	0.0229	-0.8790	0.3806	-0.0653	0.0250
GBC	-0.0398	0.0238	-1.6749	0.0957	-0.0867	0.0071
PBA	-0.0422	0.0230	-1.8368	0.0679	-0.0875	0.0031
PCA	0.0074	0.0230	0.3204	0.7490	-0.0380	0.0527
PCB	-0.0220	0.0238	-0.9247	0.3563	-0.0690	0.0250
MANG	-0.1894	0.0236	-8.0231	0.0000	-0.2359	-0.1428
RHO	0.0173	0.0230	0.7524	0.4528	-0.0280	0.0626
RHOC	0.0291	0.0225	1.2926	0.1978	-0.0153	0.0735
EFS	-0.0122	0.0228	-0.5338	0.5941	-0.0572	0.0329
SIGY	-0.0162	0.0232	-0.6982	0.4860	-0.0619	0.0295
FIO	-0.0115	0.0221	-0.5231	0.6016	-0.0551	0.0320
T	0.0387	0.0237	1.6353	0.1037	-0.0080	0.0854
S	-0.0540	0.0227	-2.3735	0.0187	-0.0988	-0.0091
NOTCHR	0.3196	0.0225	14.1995	0.0000	0.2752	0.3641

```
==================================================================

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

                Results: Ordinary least squares
==================================================================
Model:              OLS             R-squared:          0.423
Dependent Variable: gMTS            Adj. R-squared:     0.327
No. Observations:   210
Df Model:           30
------------------------------------------------------------------
               Coef.   Std.Err.     t     P>|t|    [0.025   0.975]
------------------------------------------------------------------
Intercept       0.3225    0.0985   3.2735  0.0013   0.1281   0.5169
XT              0.0338    0.0366   0.9253  0.3560  -0.0383   0.1060
XC             -0.0435    0.0351  -1.2392  0.2169  -0.1128   0.0258
YT             -0.0162    0.0355  -0.4564  0.6487  -0.0864   0.0539
YC             -0.0049    0.0351  -0.1389  0.8897  -0.0741   0.0644
SL             -0.0203    0.0355  -0.5730  0.5673  -0.0903   0.0497
ENKINK         -0.0571    0.0369  -1.5471  0.1236  -0.1299   0.0157
ENA            -0.0120    0.0354  -0.3391  0.7350  -0.0820   0.0579
ENB            -0.0222    0.0354  -0.6269  0.5315  -0.0920   0.0477
ENT             0.0396    0.0373   1.0624  0.2895  -0.0340   0.1132
ENL             0.0310    0.0351   0.8830  0.3784  -0.0383   0.1002
GI              0.1198    0.0361   3.3167  0.0011   0.0485   0.1911
GII            -0.1089    0.0359  -3.0290  0.0028  -0.1798  -0.0379
EA             -0.0212    0.0362  -0.5845  0.5596  -0.0927   0.0503
EB              0.0160    0.0354   0.4503  0.6530  -0.0539   0.0858
EC              0.0061    0.0355   0.1726  0.8632  -0.0640   0.0763
GAB             0.0299    0.0380   0.7871  0.4322  -0.0451   0.1049
GCA            -0.0362    0.0354  -1.0227  0.3078  -0.1062   0.0337
GBC            -0.0182    0.0368  -0.4951  0.6211  -0.0909   0.0544
PBA             0.0745    0.0356   2.0931  0.0378   0.0043   0.1446
PCA             0.0389    0.0356   1.0950  0.2750  -0.0312   0.1091
PCB            -0.0129    0.0369  -0.3487  0.7277  -0.0857   0.0599
MANG            0.0971    0.0366   2.6576  0.0086   0.0250   0.1693
RHO            -0.0069    0.0356  -0.1943  0.8461  -0.0771   0.0633
RHOC           -0.0351    0.0349  -1.0070  0.3153  -0.1039   0.0337
EFS            -0.0720    0.0354  -2.0370  0.0431  -0.1418  -0.0023
SIGY           -0.0109    0.0359  -0.3041  0.7614  -0.0817   0.0599
FIO            -0.0240    0.0342  -0.7036  0.4826  -0.0915   0.0434
T               0.0980    0.0367   2.6717  0.0082   0.0256   0.1703
S              -0.0294    0.0352  -0.8355  0.4046  -0.0989   0.0401
NOTCHR         -0.2566    0.0349  -7.3603  0.0000  -0.3254  -0.1878
==================================================================
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Results: Ordinary least squares

Model:	OLS	R-squared:	0.946
Dependent Variable:	gMPS	Adj. R-squared:	0.937
No. Observations:	210		
Df Model:	30		

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	0.0866	0.0442	1.9593	0.0516	-0.0006	0.1739
XT	0.8376	0.0164	51.0369	0.0000	0.8052	0.8700
XC	-0.0117	0.0158	-0.7415	0.4594	-0.0428	0.0194
YT	-0.0588	0.0160	-3.6827	0.0003	-0.0902	-0.0273
YC	-0.0180	0.0158	-1.1400	0.2558	-0.0490	0.0131
SL	-0.0077	0.0159	-0.4836	0.6292	-0.0391	0.0237
ENKINK	0.0155	0.0166	0.9342	0.3515	-0.0172	0.0482
ENA	0.0013	0.0159	0.0800	0.9364	-0.0301	0.0327
ENB	-0.0336	0.0159	-2.1162	0.0357	-0.0650	-0.0023
ENT	-0.0039	0.0167	-0.2346	0.8148	-0.0370	0.0291
ENL	0.0122	0.0158	0.7731	0.4405	-0.0189	0.0433
GI	-0.0085	0.0162	-0.5243	0.6007	-0.0405	0.0235
GII	-0.0025	0.0161	-0.1567	0.8757	-0.0344	0.0293
EA	-0.0040	0.0163	-0.2481	0.8044	-0.0361	0.0281
EB	-0.0024	0.0159	-0.1501	0.8809	-0.0338	0.0290
EC	0.0010	0.0160	0.0634	0.9495	-0.0305	0.0325
GAB	-0.0019	0.0171	-0.1087	0.9135	-0.0355	0.0318
GCA	-0.0058	0.0159	-0.3624	0.7175	-0.0372	0.0256
GBC	-0.0207	0.0165	-1.2530	0.2118	-0.0533	0.0119
PBA	0.0513	0.0160	3.2135	0.0016	0.0198	0.0828
PCA	-0.0053	0.0160	-0.3295	0.7421	-0.0368	0.0262
PCB	-0.0023	0.0166	-0.1365	0.8916	-0.0349	0.0304
MANG	0.0294	0.0164	1.7936	0.0746	-0.0029	0.0618
RHO	-0.0154	0.0160	-0.9649	0.3359	-0.0469	0.0161
RHOC	-0.0177	0.0156	-1.1302	0.2599	-0.0486	0.0132
EFS	-0.0413	0.0159	-2.5992	0.0101	-0.0726	-0.0099
SIGY	0.0091	0.0161	0.5631	0.5741	-0.0227	0.0408
FIO	0.0049	0.0153	0.3166	0.7519	-0.0254	0.0351
T	0.0311	0.0165	1.8893	0.0605	-0.0014	0.0636
S	0.0016	0.0158	0.1023	0.9186	-0.0296	0.0328
NOTCHR	-0.0284	0.0156	-1.8172	0.0709	-0.0593	0.0024

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Results: Ordinary least squares

Model:	OLS	R-squared:	0.843
Dependent Variable:	gMSS	Adj. R-squared:	0.817
No. Observations:	210		
Df Model:	30		

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	0.1218	0.0616	1.9773	0.0495	0.0002	0.2434
XT	0.6460	0.0229	28.2550	0.0000	0.6009	0.6911
XC	-0.0055	0.0220	-0.2519	0.8014	-0.0488	0.0378
YT	-0.0804	0.0222	-3.6176	0.0004	-0.1243	-0.0365
YC	-0.0220	0.0219	-1.0010	0.3182	-0.0653	0.0213
SL	-0.0117	0.0222	-0.5288	0.5976	-0.0555	0.0320
ENKINK	0.0076	0.0231	0.3309	0.7411	-0.0379	0.0532
ENA	-0.0021	0.0222	-0.0950	0.9244	-0.0458	0.0416
ENB	-0.0507	0.0221	-2.2891	0.0232	-0.0943	-0.0070
ENT	-0.0077	0.0233	-0.3298	0.7420	-0.0537	0.0383
ENL	0.0169	0.0219	0.7706	0.4420	-0.0264	0.0602
GI	-0.0116	0.0226	-0.5137	0.6081	-0.0562	0.0330
GII	-0.0144	0.0225	-0.6386	0.5239	-0.0587	0.0300
EA	-0.0127	0.0227	-0.5599	0.5762	-0.0574	0.0320
EB	-0.0008	0.0221	-0.0347	0.9724	-0.0445	0.0429
EC	0.0044	0.0222	0.1988	0.8427	-0.0394	0.0483
GAB	0.0036	0.0238	0.1495	0.8813	-0.0433	0.0504
GCA	-0.0093	0.0222	-0.4188	0.6759	-0.0530	0.0344
GBC	-0.0356	0.0230	-1.5444	0.1243	-0.0810	0.0099
PBA	0.0695	0.0222	3.1253	0.0021	0.0256	0.1134
PCA	-0.0061	0.0222	-0.2743	0.7842	-0.0500	0.0378
PCB	-0.0022	0.0231	-0.0942	0.9250	-0.0477	0.0434
MANG	0.0552	0.0229	2.4156	0.0167	0.0101	0.1003
RHO	-0.0115	0.0222	-0.5186	0.6047	-0.0554	0.0324
RHOC	-0.0251	0.0218	-1.1536	0.2502	-0.0682	0.0179
EFS	-0.0452	0.0221	-2.0442	0.0424	-0.0888	-0.0016
SIGY	0.0151	0.0224	0.6726	0.5021	-0.0292	0.0593
FIO	0.0038	0.0214	0.1757	0.8607	-0.0384	0.0459
T	0.0439	0.0229	1.9132	0.0573	-0.0014	0.0891
S	-0.0027	0.0220	-0.1221	0.9030	-0.0461	0.0408
NOTCHR	-0.0354	0.0218	-1.6235	0.1062	-0.0784	0.0076

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Results: Ordinary least squares

```
================================================================
Model:              OLS              R-squared:          0.814
Dependent Variable: gMIPSS           Adj. R-squared:     0.783
No. Observations:   210
Df Model:           30
----------------------------------------------------------------
                Coef.   Std.Err.     t     P>|t|    [0.025   0.975]
----------------------------------------------------------------
Intercept      -0.1979    0.0793  -2.4956  0.0135  -0.3543  -0.0414
XT              0.3576    0.0294  12.1546  0.0000   0.2996   0.4157
XC             -0.0097    0.0283  -0.3435  0.7316  -0.0655   0.0460
YT              0.0828    0.0286   2.8961  0.0042   0.0264   0.1393
YC             -0.0252    0.0282  -0.8913  0.3739  -0.0809   0.0306
SL              0.0512    0.0285   1.7953  0.0743  -0.0051   0.1076
ENKINK          0.0284    0.0297   0.9577  0.3395  -0.0302   0.0870
ENA             0.0056    0.0285   0.1968  0.8442  -0.0507   0.0619
ENB            -0.0102    0.0285  -0.3569  0.7216  -0.0664   0.0460
ENT             0.0050    0.0300   0.1663  0.8681  -0.0542   0.0642
ENL             0.0094    0.0282   0.3327  0.7398  -0.0463   0.0651
GI              0.0598    0.0291   2.0573  0.0411   0.0024   0.1172
GII            -0.0401    0.0289  -1.3867  0.1673  -0.0972   0.0170
EA             -0.0320    0.0292  -1.0969  0.2741  -0.0895   0.0256
EB              0.0518    0.0285   1.8176  0.0708  -0.0044   0.1081
EC              0.0184    0.0286   0.6431  0.5210  -0.0380   0.0748
GAB             0.0532    0.0306   1.7393  0.0837  -0.0072   0.1135
GCA             0.0037    0.0285   0.1312  0.8958  -0.0525   0.0600
GBC             0.0049    0.0296   0.1660  0.8683  -0.0536   0.0634
PBA             0.0084    0.0286   0.2927  0.7701  -0.0481   0.0649
PCA            -0.0137    0.0286  -0.4776  0.6335  -0.0701   0.0428
PCB            -0.0351    0.0297  -1.1832  0.2383  -0.0937   0.0235
MANG            0.6462    0.0294  21.9671  0.0000   0.5881   0.7042
RHO            -0.0111    0.0286  -0.3863  0.6997  -0.0676   0.0454
RHOC            0.0087    0.0281   0.3108  0.7563  -0.0466   0.0641
EFS            -0.0096    0.0285  -0.3368  0.7367  -0.0657   0.0466
SIGY           -0.0487    0.0289  -1.6887  0.0930  -0.1057   0.0082
FIO            -0.0560    0.0275  -2.0347  0.0434  -0.1102  -0.0017
T               0.0305    0.0295   1.0346  0.3022  -0.0277   0.0888
S               0.0053    0.0283   0.1870  0.8519  -0.0506   0.0612
NOTCHR         -0.0259    0.0281  -0.9214  0.3581  -0.0812   0.0295
================================================================

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

                Results: Ordinary least squares
================================================================
Model:              OLS              R-squared:          0.854
```

```
Dependent Variable: SC               Adj. R-squared:     0.830
No. Observations:   210
Df Model:           30
------------------------------------------------------------------------
               Coef.   Std.Err.     t      P>|t|    [0.025   0.975]
------------------------------------------------------------------------
Intercept      0.3281    0.0596    5.5018  0.0000   0.2104   0.4458
XT             0.4644    0.0221   20.9779  0.0000   0.4207   0.5080
XC            -0.0384    0.0213   -1.8068  0.0725  -0.0803   0.0035
YT            -0.1667    0.0215   -7.7479  0.0000  -0.2092  -0.1243
YC            -0.0042    0.0212   -0.1973  0.8438  -0.0461   0.0377
SL            -0.0993    0.0215   -4.6245  0.0000  -0.1417  -0.0569
ENKINK         0.0104    0.0223    0.4658  0.6420  -0.0337   0.0545
ENA            0.0363    0.0215    1.6908  0.0926  -0.0061   0.0786
ENB           -0.0153    0.0214   -0.7151  0.4755  -0.0576   0.0270
ENT           -0.0031    0.0226   -0.1389  0.8897  -0.0477   0.0414
ENL           -0.0191    0.0212   -0.9007  0.3689  -0.0611   0.0228
GI             0.0352    0.0219    1.6111  0.1089  -0.0079   0.0784
GII           -0.0683    0.0218   -3.1396  0.0020  -0.1113  -0.0254
EA             0.0002    0.0219    0.0101  0.9920  -0.0431   0.0435
EB             0.0096    0.0214    0.4471  0.6553  -0.0327   0.0519
EC            -0.0042    0.0215   -0.1955  0.8452  -0.0467   0.0382
GAB           -0.0299    0.0230   -1.2988  0.1957  -0.0753   0.0155
GCA           -0.0049    0.0215   -0.2282  0.8197  -0.0472   0.0374
GBC            0.0172    0.0223    0.7698  0.4424  -0.0268   0.0612
PBA            0.0603    0.0215    2.8004  0.0057   0.0178   0.1028
PCA           -0.0018    0.0215   -0.0827  0.9342  -0.0443   0.0407
PCB            0.0331    0.0223    1.4825  0.1400  -0.0110   0.0772
MANG           0.1960    0.0221    8.8591  0.0000   0.1524   0.2397
RHO           -0.0330    0.0215   -1.5300  0.1278  -0.0755   0.0095
RHOC          -0.0374    0.0211   -1.7744  0.0777  -0.0791   0.0042
EFS           -0.0237    0.0214   -1.1050  0.2707  -0.0659   0.0186
SIGY           0.0088    0.0217    0.4047  0.6862  -0.0341   0.0516
FIO            0.0046    0.0207    0.2247  0.8225  -0.0362   0.0455
T             -0.0180    0.0222   -0.8092  0.4195  -0.0618   0.0258
S              0.0338    0.0213    1.5847  0.1148  -0.0083   0.0758
NOTCHR        -0.2906    0.0211  -13.7685  0.0000  -0.3323  -0.2490
========================================================================
```

A.4 Code Repository

Following source code used in the work can also be found at GitHub repository

A.4.1 Step-0: Initial Setup and Input Variables

```
#==============================================================================
# Author: Apurv Kulkarni
```

```
#-------------------------------------------------------------------------------
# Analysis setup
#===============================================================================

def inputfactors():
    #===========================================================================
    # Project directory setup
    #===========================================================================

    # Initializing input parameter dictionary
    ip = {}

    # Analysis Type
    # 1 : Material parameters as the only design variable
    # 2 : Notch geometry as the only design variable
    # 3 : Material parameter and Notch geometry as the only design variable
    ip['analysis_type'] = 3

    # Main project folder name (contains all the files where database will be created
    )
    project_name = 'project_xyz'
    ip['project_name'] = project_name

    # Flag for setting if the python scripts are on "local location" or on "remote
    location"
    # remote_setup = 0
    # ip['remote_setup'] = remote_setup

    # Database main folder with project folder
    project_path = f"D:\\Academics\\MasterThesisData\\DataAnalysis\\{project_name}\\"
    ip['project_path'] = project_path

    # Database main folder with project folder
    # project_path = '/home/apku868a/' + project_name
    # ip['project_path'] = project_path

    # Path for LS-Dyna result folder
    # This is the main folder which contains subfolders with run id as their
    # name as shown below:
    #
    # <ip['lsdyna_result_path']>
    # |
    # `-- 1
    # |   `-- d3plot files (LS_Dyna results)
    # `-- 2
    #     `-- d3plot files
    # ip['lsdyna_result_path'] = ip['project_path'] + 'results/'
    ip['lsdyna_result_path'] = f"/scratch/ws/0/apku868a-mt/{project_name}/"

    # App path for LS-PrePost
    ip['lsdyna_app'] = 'lspp'
    # ip['lsdyna_app'] = '/home/apku868a/lsprepost4.8_common/lspp48'

    # LS-Dyna card destination
    ip['card_destination_main']= ip['project_path'] + "cards/"
```

```
56      ip['card_destination']= ip['card_destination_main']+ip['project_name']+'/'
57
58      # Directory for saving "serverJob" script lists (if job_submission_type=1).
59      # Please refer to "mt_2_CreateHpcJobs.py" for more info
60      ip['job_dst']= ip['project_path'] + "hpc_jobs/"
61
62      # Location of macros for data cleaning and extraction
63      ip['macro_dst_newFormat'] = ip['project_path'] + "macro\\" + "newFormat/"
64      ip['macro_dst_data'] = ip['project_path'] + "macro\\" + "dataExtraction/"
65      ip['csv_destination']   = ip['project_path'] + "output_csv/"
66
67      # Result location on local machine
68      ip['results_location']  = ip['project_path'] + "results/"
69
70      # Directory where, data extracted and cleaned, from all the csv ouput,
71      # is saved
72      ip['extractedDatabase'] = ip['project_path'] + 'extractedDatabase/'
73
74      # If analysis is run on HPC server
75      ip['hpc_loc'] = "/scratch/ws/0/apku868a-mt/"
76
77      # Plot file location for data analysis
78      ip['plot_loc'] = ip['project_path'] + 'plots/'
79
80      # Default geometry and material location
81      # For analysis_type = 1 : default geometry is required
82      # For analysis_type = 2 : default material is required
83      ip['default_cards'] = 'default_cards/'
84
85      ip['geo_dest'] = ip['project_path'] + 'geometry/'
86      if ip['analysis_type'] == 1:
87          ip['geo_srcfilename'] = ip['default_cards'] + 'default_geometry.k'
88      elif ip['analysis_type'] == 2:
89          ip['mat_srcloc'] = ip['default_cards'] + 'default_material.k'
90          ip['mat_dst'] = ip['project_path'] + 'material/'
91
92      #=========================================================================
93      # Input parameter setup
94      #=========================================================================
95      #=========================================================================
96      # Type val        | Variation type | Distribution type | Requirements
97      #-------------------------------------------------------------------------
98      #   10             percentage(%)      log uniform       'value','vari'
99      #   11             percentage(%)      linear            'value','vari'
100     #   20             absolute values    log uniform       'min_val','max_val'
101     #   21             absolute values    linear            'min_val','max_val'
102     # 3(experimental)  percentage(%)      discrete levels   'levels'
103     #-------------------------------------------------------------------------
104     # 'value'   : basevalue
105     # 'vari'    : % variation
106     # 'min_val' : minimum value
107     # 'max_val' : maximum value
108     # 'levels'  : discrete levels.
109     #             Eg: var:{'type':3,'levels':[5,8,9]}. 'var' will have only
110     #             these discrete values assigned in random order.
```

```
111     # ========================================================================
112
113     variation1 = 20
114     variation2 = 40
115     variation3 = 80
116
117     factor_list = {
118         # Pinho material model
119         #-------------------------------------------------------------------
120         # Mass Density
121         'rho' :{'type':11, 'value': 1530,    'vari':variation1},
122         # Young's modulus in longitudinal direction (a)
123         'ea'  :{'type':11, 'value': 1.21e+11,'vari':variation1},
124         # _||_ in transverse direction (b)
125         'eb'  :{'type':11, 'value': 9.24e+9, 'vari':variation1},
126         # _||_ in through thickness direction (c)
127         'ec'  :{'type':11, 'value': 9.24e+9, 'vari':variation1},
128         # Shear modulus ab
129         'gab' :{'type':11, 'value': 1.35e+10,'vari':variation1},
130         # Shear modulus ca
131         'gca' :{'type':11, 'value': 1.35e+10,'vari':variation1},
132         # Shear modulus bc
133         'gbc' :{'type':11, 'value': 2.92e+9, 'vari':variation1},
134         # Maximum effective strain for element layer failure
135         'efs' :{'type':21, 'min_val':-0.4, 'max_val':-0.1},
136         # Poisson's ratio in ba
137         'pba' :{'type':11, 'value': 0.015,   'vari':variation1},
138         # Poisson's ratio in ca
139         'pca' :{'type':11, 'value': 0.015,   'vari':variation1},
140         # Poisson's ratio in cb
141         'pcb' :{'type':11, 'value': 0.37,    'vari':variation1},
142         # Misalignment angle
143         'mang':{'type':21, 'min_val':0, 'max_val': 10},
144         # Fracture toughness for longitudinal (fiber) compressive failure mode.
145         'enk' :{'type':11, 'value': 4e+7,    'vari':variation3},
146         # Fracture toughness for longitudinal (fiber) tensile failure mode.
147         'ena' :{'type':11, 'value': 4.5e+7,  'vari':variation3},
148         # Fracture toughness for intralaminar matrix tensile failure.
149         'enb' :{'type':11, 'value': 2.5e+3,  'vari':variation3},
150         # Fracture toughness for intralaminar matrix transverse shear failure.
151         'ent' :{'type':11, 'value': 4e+3,    'vari':variation3},
152         # Fracture toughness for intralaminar matrix longitudinal shear failure.
153         'enl' :{'type':11, 'value': 4e+3,    'vari':variation3},
154         # Longitudinal compressive strength, a-axis (positive value).
155         'xc'  :{'type':11, 'value': 6.5e+9,  'vari':variation2},
156         # Longitudinal tensile strength, a-axis.
157         'xt'  :{'type':11, 'value': 1.79e+9, 'vari':variation2},
158         # Transverse compressive strength, b-axis (positive value).
159         'yc'  :{'type':11, 'value': 1.3e+8,  'vari':variation2},
160         # Transverse tensile strength, b-axis.
161         'yt'  :{'type':11, 'value': 4.2e+7,  'vari':variation2},
162         # Longitudinal shear strength.
163         'sl'  :{'type':11, 'value': 7.4e+7,  'vari':variation2},
164         # In-plane shear yield stress
165         'sig' :{'type':11, 'value': 0.75e+6, 'vari':variation2},
```

```
        # Fracture angle in pure transverse compression
        'fio' :{'type':21, 'min_val':51, 'max_val': 55},

        # Cohesive material parameters
        #-----------------------------------------------------------------------
        # Mass Density of cohesive material
        'rhC' :{'type':11, 'value': 6.50,    'vari':variation1},
        # Fracture toughness / energy release rate   for mode I.
        'gi'  :{'type':11, 'value': 500,     'vari':variation3},
        # Fracture toughness / energy release rate   for mode II.
        'gii' :{'type':11, 'value': 800,     'vari':variation3},
        # peak traction in normal direction mode
        't'   :{'type':21, 'min_val':0.37E+08, 'max_val':4.37E+08},
        # Peak traction in tangential direction (mode II).
        's'   :{'type':21, 'min_val':0.37E+08, 'max_val':4.37E+08},

        # Geometry parameters
        #-----------------------------------------------------------------------
        # Longitudinal radius
        'rad' :{'type':21,'value':'nan','min_val':0.0005, 'max_val':0.01}
        }

    ip['factor_list'] = factor_list

    # Total number of experiments (experiment ID: 0,1,..., max_runs-1)
    ip['max_runs'] = 210

    # MAximum number of iterations for generating optimum LHD matrix
    ip['lhs_iteration'] = 1000

    # For frequency of output (d3plot files): [[time,time_step_size],[],..]
    ip['output_file_curve'] = [
        [0,0.0014],[0.0005, 1.00E-04],[0.0009, 1.00E-05],[0.00128, 5.00E-06],
        [0.00192, 5.00E-06],[0.0022, 8.00E-05],[0.0025, 7.00E-04],
        [0.0035, 0.001]
        ]

    return ip


def createProjDirStruct(ip):
    #=============================================================================
    # Creating project directory structure
    #=============================================================================
    import os
    import shutil

    assert(type(ip)==dict)
    # Creating project directory structure
    os.makedirs(ip['project_path'], exist_ok=True)
    os.makedirs(ip['card_destination'], exist_ok=True)
    for run in range(0,ip['max_runs']):
        os.makedirs(ip['card_destination'] + f'{run}/', exist_ok=True)
    os.makedirs(ip['job_dst'], exist_ok=True)
    # os.makedirs(ip['results_location'], exist_ok=True)
```

```
    os.makedirs(ip['macro_dst_newFormat'], exist_ok=True)
    os.makedirs(ip['macro_dst_data'], exist_ok=True)
    os.makedirs(ip['csv_destination'], exist_ok=True)
    for run in range(0,ip['max_runs']):
            os.makedirs(ip['csv_destination'] + f'{run}/', exist_ok=True)
    os.makedirs(ip['extractedDatabase'], exist_ok=True)
    os.makedirs(ip['geo_dest'], exist_ok=True)
    os.makedirs(ip['plot_loc'], exist_ok=True)
    if ip['analysis_type']==1:
        shutil.copy2(ip['geo_srcfilename'], ip['geo_dest']+'geo.k')
    elif ip['analysis_type']==2 or 3:
        for run in range(0,ip['max_runs']):
            os.makedirs(ip['geo_dest'] + f'{run}/', exist_ok=True)
        if ip['analysis_type']==2:
            os.makedirs(ip['mat_dst'], exist_ok=True)
            shutil.copy2(ip['mat_srcloc'], ip['mat_dst'])
```

Listing A.1: Input Factors

A.4.2 Step-1: Database and LS-Dyna Card Creation

```
#==============================================================================
# Author: Apurv Kulkarni
#------------------------------------------------------------------------------
# Creating LHD experiment and LS-Dyna cards
#==============================================================================

# Loading dependencies
import os
import pandas as pd
import numpy as np
import pyDOE2 as doe
from shutil import rmtree
from mt_0_input_factors_all import *
from utilities.mt_x_projectUtilities import *
from utilities.mt_x_lsppMacro import newCardFormat
from utilities.mt_x_OHT_StructuredMesh_GmshLsdyna import create_geo
from datetime import datetime

now = datetime.now()
current_time = now.strftime("%H:%M:%S")

# Get input data
ip = inputfactors()

# Create project directory structure
createProjDirStruct(ip)

project_name        = ip['project_name']
project_path        = ip['project_path']
factor_list         = ip['factor_list']
card_destination    = ip['card_destination']
geometry_loc        = ip['geo_dest']
ana_type            = ip['analysis_type']
max_runs            = ip['max_runs']
```

```
35
36 print("Project: ",project_name)
37 print("Project_path: ",project_path)
38 print("Number of input factors: ",len(ip['factor_list']))
39 print("Total number of experiments: ", ip['max_runs'])
40
41 ###############################################################################
42 # Create design matrix
43 ###############################################################################
44
45 # Get all input factors (design variables)
46 factor_key = list(factor_list.keys())
47
48 # Flags
49 create_database_flag = 1
50 create_card_flag = 1
51 create_material_dataframe = 1
52
53 # Creating min/max values for factors
54 factor_list = factor_minmax(factor_list)
55
56 # Create LHD matrix
57 seed = 74 #np.random.randint(0,100)
58 print("Random seed: ", seed)
59 with open(project_path+'seed_value.info','a') as myseed:
60     str_seed = current_time + ' - ' + str(seed) + '\n'
61     myseed.write(str_seed)
62
63 sample_space = doe.lhs(n=len(factor_list),samples=ip['max_runs'],
64                        criterion='correlation',iterations=ip['lhs_iteration'],
65                        random_state=seed).transpose()
66
67 # Transforming LHD samples into experimental samples
68 sample_trafo = transform_samplespace(sample_space, factor_list)
69
70 # Creating material angles (vectors) from misalignment angles
71 # a_1,d_1 - vectors of 0deg ply
72 # a_2,d_2 - vectors of 90deg ply
73 if ana_type == 1 or ana_type == 3:
74     mang = sample_trafo['mang']
75     a1_ang = mang
76     a1_ang = np.asarray(a1_ang)
77     a_1 = direction_vector(a1_ang)
78     sample_trafo['a_1'] = a_1
79     d1_ang = a1_ang + 90
80     d_1 = direction_vector(d1_ang)
81     sample_trafo['d_1'] = d_1
82     a_2 = d_1
83     sample_trafo['a_2'] = a_2
84     d2_ang = a1_ang + 180
85     d_2 = direction_vector(d2_ang)
86     sample_trafo['d_2'] = d_2
87
88     factor_key.extend(['a_1','d_1','a_2','d_2'])
89
```

```
    # Formatting sample according to LS-Dyna layout (10 nonospaced length)
    sample_formatted = format_val(sample_trafo)

################################################################################
# Creating analysis cards
################################################################################
# Material card:
# mid 1.... ply 0deg
# mid 2.... ply 90deg
# mid 3.... cohesive element layer

# Initializing database dictionary
database = {}

# if remote_setup:
saveMacroBash_str = ''

for run in range(0,max_runs):

    # Creating database
    #-------------------------------------------------------------------------
    layerAll_database = {}
    layer1_database = {}
    layer2_database = {}
    cohesive_mat_database = {}
    if create_database_flag == 1:

        # common material paramters
        layerAll_material_parameters = factor_key.copy()
        layerAll_material_parameters.remove('a_1')
        layerAll_material_parameters.remove('a_2')
        layerAll_material_parameters.remove('d_1')
        layerAll_material_parameters.remove('d_2')
        for key in layerAll_material_parameters:
            layerAll_database[key] = sample_formatted[key][run]

        # Composite material - 0deg plies
        layer1_database = layerAll_database.copy()
        mid = 1
        layer1_database['mid'] = ' '*(10 - len(str(mid))) + str(mid)
        for key in ['a_1','d_1']:#splitting angle vectors into sub components
            layer1_database[f'{key[0]}1'] = sample_formatted[key][run][0]
            layer1_database[f'{key[0]}2'] = sample_formatted[key][run][1]

        # Composite material - 90deg plies
        layer2_database = layerAll_database.copy()
        mid = 2
        layer2_database['mid'] = ' '*(10 - len(str(mid))) + str(mid)
        for key in ['a_2','d_2']:#splitting angle vectors into sub components
            layer2_database[f'{key[0]}1'] = sample_formatted[key][run][0]
            layer2_database[f'{key[0]}2'] = sample_formatted[key][run][1]

        # Cohesive material
        cohesive_mat_database = layerAll_database.copy()
        mid = 3 # adding material id
```

```
145         cohesive_mat_database['mid'] = ' '*(10 - len(str(mid))) + str(mid)
146 
147     if create_material_dataframe == 1:
148         # for creating data frame
149         database[run] = cohesive_mat_database.copy()
150         database[run].pop('mid')
151 
152         # converting strings into float values
153         for key in database[run]:
154             database[run][key] = float(database[run][key])
155 
156     # Creating cards
157     #-------------------------------------------------------------------------
158     if create_card_flag == 1:
159 
160         # Get materaial data from database
161         # ---------------------------------------------------------------------
162         # 0 deg ply
163         l1_str = create_mat_card(mat_database=layer1_database, typ=1)
164         # 90 deg ply
165         l2_str = create_mat_card(mat_database=layer2_database, typ=1)
166         # cohesive layer
167         coh_str = create_mat_card(mat_database=cohesive_mat_database, typ=2)
168         # Other information
169         misc_str = create_mat_card(typ=3)
170 
171         mat_str = l1_str + l2_str + coh_str + misc_str
172 
173         # Get geometry
174         # ---------------------------------------------------------------------
175         geo_file_loc = geometry_loc + f'{run}/'*(ana_type!=1)
176         geo_full_loc = geo_file_loc + 'geo.k'
177 
178         # Create new geometry if radius is a design varible
179         if ana_type == 2 or ana_type == 3:
180             radius = float(sample_formatted['rad'][run])
181 
182             create_geo(Rx               = radius,
183                        filename         = 'geo',
184                        file_dst         = geo_file_loc,
185                        numCurveElem     = 35,
186                        numClampXElem    = 4,
187                        numMainbodyXElem = 12,
188                        numCSElem        = 20)
189 
190         # Get string containing mesh information from the file
191         with open(geo_full_loc) as mygeo:
192             geo_str = mygeo.read()
193             geo_str = geo_str[:-5]      # removing "*END" from geomtery file
194 
195         # Get string with curve info about d3plot output generation frequency
196         output_curve_string = dyna_output_curve(plot = ip['output_file_curve'])
197 
198         # Create final card with geometry and material data
199         # ---------------------------------------------------------------------
```

```
        full_str = geo_str + output_curve_string + mat_str + "\n*END"

        card_dst = (card_destination + str(run) + '\\'*(os.name=='nt') +
                    '/' * (os.name=='posix'))
        filename_main = card_dst + f'{run}.k'
        with open(filename_main,'w+') as myfile:
            myfile.write(full_str)

        # "create_geo" function fails to convert the geometry file into new
        # format following macro can be run to convert the cards into
        # new format. This can happen when setup is on remote server (HPC)
        macrofilename = f'{run}.cfile'
        saveMacroBash_str += newCardFormat(
            macro_file      = ip['macro_dst_newFormat']+ macrofilename,
            dyna_card_loc   = filename_main,
            lspp_loc        = ip['lsdyna_app'])

        # Cleaning temporary temporary geomerty files
        if ana_type == 2 or ana_type == 3:
            rmtree(geometry_loc + str(run))

    print(run, 'Complete')

# Dave input design matrix
if create_material_dataframe == 1:
    df = pd.DataFrame.from_dict(database, orient='index')
    df.to_csv(ip['extractedDatabase'] + "material_dataframe.csv")

#
with open(ip['macro_dst_newFormat'] + 'saveMacro_bash.sh','w+') as mybash:
    mybash.write(saveMacroBash_str)

print("Complete .....")
print("########################################################################")
print(f"""1) Please run
\t\t cd {ip['macro_dst_newFormat']}
\t\t dos2unix *.sh
\t\t chmod -R 775 *
   if required.""")
print(f"""2) Please run \n\t\t {ip['macro_dst_newFormat']}saveMacro_bash.sh
   to convert cards into new format. """)
print("""3) Please run 'mt_2_createHpcJobs' for creating HPC job scripts as a
   next step in the analysis workflow.""")
```

A.4.3 Step-2: Creating Sbatch scripts for HPC Job Submission

```
#=============================================================================
# Author: Apurv Kulkarni
#-----------------------------------------------------------------------------
# Creating jobs to be run on HPC
# Not required to run if analysis is done locally
# Note: This doesn't use TUD's serverJob script
#=============================================================================

from utilities.mt_x_projectUtilities import creat_chain_sbatch, path_format_to_linux
```

```
from mt_0_input_factors_all import inputfactors

ip = inputfactors()
project_name = ip['project_name']
total_runs = ip['max_runs']
hpc_scratch_location = ip['hpc_loc']
main_card_dst = hpc_scratch_location + ip['project_name'] + '/'

# Job submition type
# 0: Manual job submission. Creates sbatch scripts. Requires
#    rsync and running sbatch manually
# 1: Uses University's "serverJobScript" submition script.
#    Creates lists of jobs in "job" folder.
job_submition_type = 1

# Setup for HPC sbatch files
numCPU          = 30
numNodes        = 2
simTime         = '04:00:00'
account_name    = 'p_fkv_laser'
user_email      = 'apurv.kulkarni@mailbox.tu-dresden.de'

if job_submition_type==0:

    # Creating analysis runs in chunks
    runsPerSubmission = 1
    chunks = int(total_runs/runsPerSubmission)
    if total_runs%chunks != 0:
        raise ValueError("Number of chuncks must be divisible by total runs")
    print('Total chunks: ', chunks)

    chunkBashNameList = ""
    for i in range(1,chunks+1):
        start_id = int((i-1)*total_runs/chunks)
        end_id = int(i*total_runs/chunks-1)

        # Job file
        for i in range(start_id,end_id+1):
            s1 = f"""#!/bin/bash

    #SBATCH --cpus-per-task=1
    #SBATCH --threads-per-core=1
    #SBATCH --ntasks {numCPU}
    #SBATCH --nodes={numNodes}
    #SBATCH --mem=12G
    #SBATCH --time={simTime}
    #SBATCH --account={account_name}
    #SBATCH --output={i}.log
    #SBATCH --job-name={i}
    #SBATCH --open-mode=append
    #SBATCH --mail-user={user_email}
    #SBATCH --mail-type=Begin,End,Fail

    # update info file (job has started):
    echo "Job {i+1} - running ($(pwd)/{i}.k)" > {i}.info
```

```
65
66      # load SLURM environment
67      ml modenv/scs5
68
69      # load program and additional modules if needed
70      ml LS-DYNA/12.0.0
71
72      # call program
73      srun mpp-dyna i={i}.k
74      EXITSTATUS=$?
75
76      # update info file (print exit status):
77      if [ "$EXITSTATUS" == "0" ]; then JOBSTATUS="finished"; else JOBSTATUS="cancelled
        ";fi
78      echo "Job {i+1} - """
79
80              s2 = "${JOBSTATUS} "
81              s3 = f"""($(pwd)/{i}.k)" > {i}.info
82
83      # clean up:
84      rm jobID
85      """
86              batch = s1+s2+s3
87
88              job_i = ip['card_destination']+f'{i}/runJob.sh'
89              with open(job_i,'w+') as myjob:
90                  myjob.write(batch)
91
92              # runChain bash file
93              s = ''
94              for i in range(start_id,end_id+1):
95                  s+= " " + ip['hpc_loc'] + ip["project_name"] + f"/{i}/"
96              chain_str = creat_chain_sbatch(s)
97              chunkBashName = 'runChain_'+f'{start_id}_{end_id}' + '.sh'
98              with open(ip['card_destination'] + chunkBashName,'w+') as mychain:
99                  mychain.write(chain_str)
100
101         # carefull about indentation here
102         chunkBashNameList += ('source"' + chunkBashName + '"\n' +
103                               f'echo "{chunkBashName} started"\n sleep 45m\n')
104
105     # Commands to be run after all analaysis runs are completed
106     postSimcomands = """
107     source "rm -r */mes*"
108     source "rm -r */d3dump*"
109     source "rm -r */d3full*"
110     source "rm -r */d3hsp*"
111     source "rm -r */load*"
112     source "rm -r */kill*"
113     source "rm -r */bg*"
114     source "rm -r */adptmp"
115     """
116
117     # Creating main chain submission file
118     mainBash_str = ("#!/bin/bash\n"+chunkBashNameList+
```

```
                    postSimcomands)
    with open(ip['card_destination']+'runChainChunk.sh','w+') as mychain:
            mychain.write(mainBash_str)

    print("Complete...")
    print(f"""1) On local machine:
    \t\t rsync {path_format_to_linux(ip['card_destination_main'])} {ip['hpc_loc']}

    2) On HPC server
    \t\t cd {main_card_dst}
    \t\t dos2unix *.sh
    \t\t dos2unix */*.sh
    \t\t chmod -R 775 *
    \t\t ./runChainChunk.sh
    """)

    print("""Once all the analysis are complete,
    please run 'mt_3_createLsppMacro.py'""")

elif job_submition_type==1:
    job_loc = ip['job_dst']

    results_location = ip['lsdyna_result_path']

    ## Create job for HPC
    job_name = ip['project_name']

    card_id = list(range(0,ip['max_runs']))
    list_name_full = job_loc + job_name + '.list'
    with open(list_name_full,'w+') as mylist:
        for run in card_id:
            card_loc = path_format_to_linux(
                ip['card_destination']+f'{run}/')
            mylist.write(f'serverJob --processors {numCPU} --nodes {numNodes} -j {
    card_loc}{run}.k --jobDir {run} --downloadDir {run} --time {simTime}\n')

    with open(f"{job_loc}commands.txt",'w+') as mylist:
        s = f"""To run analysis
\t\t cd {job_loc}
\t\t serverJob -j {job_name}.list --array \n
To get results:
\t\t cd {ip['project_path']}results/
\t\t serverJob -j {job_loc}{job_name}.list --array -d
"""
        print(s)
        mylist.write(s)
```

A.4.4 Step-3: Creating LS-PrePost Macros for Data Extraction

```
#================================================================================
# Author: Apurv Kulkarni
#--------------------------------------------------------------------------------
# Creating LS-Prepost macros for data extraction
#================================================================================

```

```
from mt_0_input_factors_all import inputfactors
ip = inputfactors()
import os
import utilities.mt_x_lsppMacro as mc
#%%
project_name = ip["project_name"]
project_path = ip["project_path"]
lsdyna_result_path = ip["lsdyna_result_path"]

macro_dst = ip['macro_dst_data']
local_csv_dst = ip['csv_destination']

os.makedirs(macro_dst, exist_ok=True)
os.makedirs(local_csv_dst, exist_ok=True)

print("output CSV destination: ",local_csv_dst)
print("Macro destination: ",macro_dst)

start_id=0
end_id=ip["max_runs"]-1

part_id_0 = [*list(range(1,6,2)),*list(range(8,13,2))]  # 0deg ply
part_id_90 = [*list(range(2,7,2)),*list(range(7,13,2))] # 90deg ply
part_id_c = [13]                                        # cohesive layer

for exp_id in range(start_id,end_id+1):
    macro_dst   = macro_dst
    csv_dst     = local_csv_dst + f"{exp_id}/"
    result_loc  = lsdyna_result_path + f"{exp_id}/"
    os.makedirs(csv_dst, exist_ok=True)

    macroMain = ""

    # Global applied force
    macroMain += mc.global_bodyforce_100(filename="gForce",
                                         csv_destination=csv_dst)

    # Global internal energy
    macroMain += mc.global_internal_energy(filename="gIntEnergy",
                                           csv_destination=csv_dst)

    # Maximum stress values for 0 deg ply
    macroMain += mc.get_history_data(part_id=[*part_id_0], val=1,
                                     filename="0MLS", destination_csv=csv_dst)
    macroMain += mc.get_history_data(part_id=[*part_id_0], val=2,
                                     filename="0MTS",destination_csv=csv_dst)
    macroMain += mc.get_history_data(part_id=[*part_id_0], val=4,
                                     filename="0MIPSS",destination_csv=csv_dst)
    macroMain += mc.get_history_data(part_id=[*part_id_0], val=9,
                                     filename="0MES",destination_csv=csv_dst)
    macroMain += mc.get_history_data(part_id=[*part_id_0], val=13,
                                     filename="0MSS",destination_csv=csv_dst)
    macroMain += mc.get_history_data(part_id=[*part_id_0], val=14,
                                     filename="0MPS",destination_csv=csv_dst)

```

```
62      # Maximum stress values for 90 deg ply
63      macroMain += mc.get_history_data(part_id=[*part_id_90], val=1,
64                                       filename="90MLS",destination_csv=csv_dst)
65      macroMain += mc.get_history_data(part_id=[*part_id_90], val=2,
66                                       filename="90MTS",destination_csv=csv_dst)
67      macroMain += mc.get_history_data(part_id=[*part_id_0], val=4,
68                                       filename="90MIPSS",destination_csv=csv_dst)
69      macroMain += mc.get_history_data(part_id=[*part_id_90], val=9,
70                                       filename="90MES",destination_csv=csv_dst)
71      macroMain += mc.get_history_data(part_id=[*part_id_90], val=13,
72                                       filename="90MSS",destination_csv=csv_dst)
73      macroMain += mc.get_history_data(part_id=[*part_id_90], val=14,
74                                       filename="90MPS",destination_csv=csv_dst)
75
76      # Maximum stress values for cohesive layer
77      macroMain += mc.get_history_data(part_id=[*part_id_c], val=1,
78                                       filename="cMLS",destination_csv=csv_dst)
79      macroMain += mc.get_history_data(part_id=[*part_id_c], val=2,
80                                       filename="cMTS",destination_csv=csv_dst)
81      macroMain += mc.get_history_data(part_id=[*part_id_0], val=4,
82                                       filename="cMIPSS",destination_csv=csv_dst)
83      macroMain += mc.get_history_data(part_id=[*part_id_c], val=9,
84                                       filename="cMES",destination_csv=csv_dst)
85      macroMain += mc.get_history_data(part_id=[*part_id_c], val=13,
86                                       filename="cMSS",destination_csv=csv_dst)
87      macroMain += mc.get_history_data(part_id=[*part_id_c], val=14,
88                                       filename="cMPS",destination_csv=csv_dst)
89
90      # Global maximum strain values
91      macroMain += mc.get_history_data(part_id=[*part_id_0,*part_id_90,*part_id_c],
92                                       val=57, filename="gMax_xStrain",
93                                       destination_csv=csv_dst,
94                                       data_file_suffix='_0')
95      macroMain += mc.get_history_data(part_id=[*part_id_0,*part_id_90,*part_id_c],
96                                       val=58, filename="gMax_yStrain",
97                                       destination_csv=csv_dst,
98                                       data_file_suffix='_90')
99      macroMain += mc.get_history_data(part_id=[*part_id_0,*part_id_90,*part_id_c],
100                                      val=77, filename="gMax_PrincStrain",
101                                      destination_csv=csv_dst,
102                                      data_file_suffix='_c')
103
104     # Nodal displacements
105     macroMain += mc.nodal_data_disp(nodeid=119495, filename = "header_disp",
106                                     destination_csv=csv_dst)
107
108     # Element data near notch
109     macroMain += mc.getNotchElemDataGlobal(csv_dst)
110
111     # Get elementwise fringe data
112     macroMain += mc.get_fringe_data([0,12],9,'',[1,2])
113
114     # Wrapping macro with open and close file commands
115     macro_str = mc.macro_wrapper(macroMain,file_loc=result_loc)
116
```

```
    # Wriritng macros to a file
    with open(macro_dst+f"{exp_id}.cfile","w+") as mymacro:
        mymacro.write(macro_str)

# Creating bash scripts to run the macros
total_runs = ip["max_runs"]
# runsPerSubmission = total_runs/4            # number of runs per bash scripts (to run multiple extractions at a time)
chunks = 1#int(total_runs/runsPerSubmission)  # number of chunks for extraction
if total_runs%chunks != 0:
    raise ValueError("Number of chuncks must be divisible by total runs")

chunkBashfilenameList = ""
for i in range(1,chunks+1):
    chunk_start_id = int((i-1)*total_runs/chunks)
    chunk_end_id   = int(i*total_runs/chunks-1)
    mc.write_bash(chunk_start_id,chunk_end_id,
                  macro_destination=macro_dst,
                  file_suffix=f"_{chunk_start_id}{chunk_end_id}",
                  lspp_loc=ip['lsdyna_app'] )

######### Commands to run
print("Complete...\n")
print("Run following commands to proceed:")
i=1
print(f"1) cd {macro_dst}")
i+=1
print(f"{i}) dos2unix *.sh")
i+=1
print(f"{i}) chmod -R 775 *")
i+=1
print(f"{i}) run batch scripts")
```

A.4.5 Step-4: Data Extraction and Cleaning

```
#=============================================================================
# Author: Apurv Kulkarni
#-----------------------------------------------------------------------------
# Data extraction and data cleaning of csv database extracted from results
#=============================================================================

import pandas as pd
import numpy as np
import sys
from utilities.mt_x_projectUtilities import *
import os
from mt_0_input_factors_all import inputfactors

ip = inputfactors()
csv_dest = ip['csv_destination']

start_exp_id = 0
end_exp_id   = ip['max_runs']-1

max_val = {}
```

```
max_val_idx = {}
max_stress = pd.DataFrame()
df_op = {}

for i in range(start_exp_id,end_exp_id+1):

    dfo = pd.DataFrame()
    fileName = {}
    csv_destination = csv_dest + f"{i}/"

    # Force_values
    df = pd.read_csv(csv_destination + "gForce.csv" ,skiprows=1)
    df.drop(df.columns[[-1]],axis=1, inplace=True)
    df['gMAS'] = df['X-Force']/df['Area']
    dfo = df.copy()
    dfo.drop(['Area','X-Force','Y-Force','Resultant Force'],axis=1, inplace=True)

    # Internal energy
    df = pd.read_csv(csv_destination + "gIntEnergy.csv" ,skiprows=1)
    df.drop(df.columns[[-1]],axis=1, inplace=True)
    df.columns = ['Time', 'gIE']
    df_frames = [dfo,df[df.columns[[1]][0]]]
    dfo = pd.concat(df_frames,axis=1)

    # extracting ply-wise values
    for ii in ['0','90','c']:
        # x stress
        dfo = getData(csv_destination, ii +'MLS', dfo)

        # y stress
        dfo = getData(csv_destination, ii +'MTS', dfo)

        # in-plane stress
        if ii != 'c':
            dfo = getData(csv_destination, ii +'MIPSS', dfo)

        # principle stress
        dfo = getData(csv_destination, ii + 'MPS', dfo)

        # shear stress
        dfo = getData(csv_destination,ii + 'MSS',dfo)

    # getting global values
    # Global maximum longitudinal stress observed in elements
    cols = ['0MLS','90MLS','cMLS']
    dfo['gMLS'] = dfo[cols].max(axis=1)

    # Global maximum transverse stress observed in elements
    cols = ['0MTS','90MTS','cMTS']
    dfo['gMTS'] = dfo[cols].max(axis=1)

    # Global maximum principle stress observed in elements
    cols = ['0MPS','90MPS','cMPS']
    dfo['gMPS'] = dfo[cols].max(axis=1)

```

```
    # Global maximum shear stress (Vin tresca)
    cols = ['0MSS','90MSS','cMSS']
    dfo['gMSS'] = dfo[cols].max(axis=1)

    # Global maximum in-plane shear stress
    cols = ['0MIPSS','90MIPSS']
    dfo['gMIPSS'] = dfo[cols].max(axis=1)

    # Stress concentration
    df_pStress = pd.read_csv(csv_destination + "gNotchElem_maxPrinc_max.csv",
                             index_col=False, skiprows=1, sep=',')
    df_pStress = df_pStress.drop(df_pStress.columns[2:], axis=1)
    df_pStress.columns = ['Time','maxPrincStress']

    df_nStress = pd.DataFrame(dfo['gMAS'])
    dfo['SC'] = df_pStress['maxPrincStress']/df_nStress['gMAS']

    # Get nodal displacements of moving head
    df = pd.read_csv(csv_destination + "header_disp.csv", index_col=False, skiprows
    =1, sep=',')
    df = df.drop(df.columns[5:], axis=1)
    df.columns = ['Time','xDispHeader','yDisp','zDisp','resDisp']
    df_frames = [dfo,df[df.columns[1:]]]
    dfo = pd.concat(df_frames,axis=1)

    # Values of elements near notch
    # Maximum value of damage activation function  and damage variables
    dfo = getDamageVarData(file=csv_destination+"gNotchElem_fa_max.csv",
                           header='MFA', dfo=dfo)
    dfo = getDamageVarData(file=csv_destination+"gNotchElem_fmat_max.csv",
                           header='MFMAT', dfo=dfo)
    dfo = getDamageVarData(file=csv_destination+"gNotchElem_da_max.csv",
                           header='MDA', dfo=dfo)
    dfo = getDamageVarData(file=csv_destination+"gNotchElem_dmat.csv",
                           header='MDMAT', dfo=dfo)

    # Maximum longitudinal and transvserse stress value
    dfo = getDamageVarData(file=csv_destination + "gNotchElem_xStress_max.csv",
                           header='notch_xMaxStress', dfo=dfo)
    dfo = getDamageVarData(file=csv_destination + "gNotchElem_yStress_max.csv",
                           header='notch_yMaxStress', dfo=dfo)

    # Cleaning database
    dfo = dfo.replace([np.inf, -np.inf], np.nan)

    # Creating directory
    op_dataFrame_dir = ip['extractedDatabase']
    try: os.mkdir(csv_dest + op_dataFrame_dir)
    except: pass

    # saving the dataframe to dictionary
    df_op[i] = dfo

    # getting index of maximum applied stress
    temp = dfo.nlargest(n=1,columns='gMAS')
```

```
    temp['max_stress_idx'] = temp.index.values[0]
    temp.index.values[0] = i                          # assgning experiment number as
    index column
    if i == list(range(start_exp_id,end_exp_id+1))[0]:
         max_stress=temp.copy(deep=True)
    else:
        max_stress = max_stress.append(temp,ignore_index=False)

    # Maximum value database (dictionary database)
    columns = dfo.columns.values
    max_val[i] = {}
    max_val_idx[i] = {}
    for col in columns:
        temp2 = dfo.nlargest(n=1,columns=col)
        max_val[i][col] = temp2[col].values[0]
        max_val_idx[i][col] = temp2.index.values[0] + 1 # index starts at 0 but state
     id starts at 1

    print(i, '\tcomplete')

# converting maximum value dictionary database into dataframe
df_max_val = pd.DataFrame.from_dict(max_val,orient='index')
df_max_val_idx = pd.DataFrame.from_dict(max_val_idx,orient='index')

# Saving dictionary of dataframes
import pickle
with open(f"{op_dataFrame_dir}df_full.pickle", 'wb') as handle:
    pickle.dump(df_op, handle, protocol=pickle.HIGHEST_PROTOCOL)

# Saving dataframes
if 'gMAS' in dfo.columns:
    max_stress.to_csv(f"{op_dataFrame_dir}/df_maxLoad.csv")
    df_max_force_norm = min_max_scaling(max_stress)
    df_max_force_norm.to_csv(f"{op_dataFrame_dir}/df_maxLoad_norm.csv")

df_max_val.to_csv(f"{op_dataFrame_dir}/df_maxVal.csv")
df_max_val_idx.to_csv(f"{op_dataFrame_dir}/df_maxVal_idx.csv")
df_max_val_norm = min_max_scaling(df_max_val)
df_max_val_norm = df_max_val_norm.to_csv(f"{op_dataFrame_dir}/df_maxVal_norm.csv")
```

A.4.6 Step-5: Data Analysis

```
#=============================================================================
# Author: Apurv Kulkarni
#-----------------------------------------------------------------------------
# Data analysis using scatterplot, heatmap, regression, OLS, PCC, SRCC
#=============================================================================

from utilities.mt_x_projectUtilities import *
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc
from mt_0_input_factors_all import inputfactors
from statsmodels.formula.api import ols
```

```
14 import seaborn as sns
15
16 # for Latex fonts
17 rc('font',**{'family':'sans-serif','sans-serif':['Helvetica']})
18 rc({'font.size':20})
19 plt.rc('text', usetex=True)
20 plt.rc('font', family='serif')
21 fontsize=14
22
23 # for timestamp on saved plots filename
24 import datetime
25 timestamp = datetime.datetime.now().strftime("%Y%m%d%H%M%S")
26
27 # get input data
28 ip = inputfactors()
29
30 # Project name
31 project_name = ip['project_name']
32
33 extractedDatabse_loc = ip['extractedDatabase']
34
35 plot_path = ip['plot_loc']
36
37 # Flags to create corresponding plot type
38 #-------------------------------------------------------------------------------
39
40 scatterPlot     = 1     # Scatter plot flag
41 heatmapFlag     = 1     # heatmap flag
42 multiLinearReg  = 1     # Multiple linear regrression (OLS)
43
44 # Whether to normalize the plots (for both x- and y-axes)
45 normalize       = 1
46
47 # Getting extracted data
48 #-------------------------------------------------------------------------------
49 # Initializing dataframes
50 df_maxLoad_all = pd.DataFrame()
51 df_maxVal_all = pd.DataFrame()
52 df_input_all = pd.DataFrame()
53
54 df_loc = extractedDatabse_loc
55
56 df_filename = 'material_dataframe.csv'
57 df_ = pd.read_csv(df_loc + df_filename)
58 df_.drop(df_.columns[[0]], axis=1, inplace=True)
59 if 'rad' not in df_.columns:
60     df_['rad'] = 0.0024
61 df_frames = [df_input_all,df_]
62 df_input_all = pd.concat(df_frames,axis=0)
63
64 # cleaning database
65 df_input_all = df_input_all.replace([np.inf, -np.inf], np.nan)
66
67 df_filename = 'df_maxLoad.csv'
68 df_ = pd.read_csv(df_loc + df_filename)
```

```
df_.drop(df_.columns[[0]], axis=1, inplace=True)
df_frames = [df_maxLoad_all,df_]
df_maxLoad_all = pd.concat(df_frames,axis=0)
# df_maxLoad_all = df_maxLoad_all.drop(columns='MFA')


# Selecting design variables to plot
# Uncomment or comment to skip or plot, respectively
# ------------------------------------------------------------------------

df_input_all_new = pd.DataFrame()

df_input_all_new['XT']      = df_input_all['xt']
df_input_all_new['XC']      = df_input_all['xc']
df_input_all_new['YT']      = df_input_all['yt']
df_input_all_new['YC']      = df_input_all['yc']
df_input_all_new['SL']      = df_input_all['sl']

df_input_all_new['ENKINK']  = df_input_all['enk']
df_input_all_new['ENA']     = df_input_all['ena']
df_input_all_new['ENB']     = df_input_all['enb']
df_input_all_new['ENT']     = df_input_all['ent']
df_input_all_new['ENL']     = df_input_all['enl']
df_input_all_new['GI']      = df_input_all['gi']
df_input_all_new['GII']     = df_input_all['gii']

df_input_all_new['EA']      = df_input_all['ea']
df_input_all_new['EB']      = df_input_all['eb']
df_input_all_new['EC']      = df_input_all['ec']
df_input_all_new['GAB']     = df_input_all['gab']
df_input_all_new['GCA']     = df_input_all['gca']
df_input_all_new['GBC']     = df_input_all['gbc']
df_input_all_new['PBA']     = df_input_all['pba']
df_input_all_new['PCA']     = df_input_all['pca']
df_input_all_new['PCB']     = df_input_all['pcb']

df_input_all_new['MANG']    = df_input_all['mang']
df_input_all_new['RHO']     = df_input_all['rho']
df_input_all_new['RHOC']    = df_input_all['rhC']
df_input_all_new['EFS']     = df_input_all['efs']
df_input_all_new['SIGY']    = df_input_all['sig']
df_input_all_new['FIO']     = df_input_all['fio']
df_input_all_new['T']       = df_input_all['t']
df_input_all_new['S']       = df_input_all['s']
df_input_all_new['NOTCHR']  = df_input_all['rad'] / 0.0024

xvar = df_input_all_new.columns

# Selecting output variables
#-------------------------------------------------------------------------------
# Select which database is required for plotting
# level = 'laminate': global values
#       = 'ply'     : ply values (0deg,90deg,cohesive layer)
#       = 'both'    : a;; values
level = 'laminate'
```

```
124
125 df_maxLoad_all_new = pd.DataFrame()
126
127 if level == 'laminate':
128
129     df_maxLoad_all_new['TimeMAS']   = df_maxLoad_all['Time']
130
131     df_maxLoad_all_new['gMAS']      = df_maxLoad_all['gMAS']
132     df_maxLoad_all_new['gMLS']      = df_maxLoad_all['gMLS']
133     df_maxLoad_all_new['gMTS']      = df_maxLoad_all['gMTS']
134     df_maxLoad_all_new['gMPS']      = df_maxLoad_all['gMPS']
135     df_maxLoad_all_new['gMSS']      = df_maxLoad_all['gMSS']
136     df_maxLoad_all_new['gMIPSS']    = df_maxLoad_all['gMIPSS']
137     df_maxLoad_all_new['SC']        = df_maxLoad_all['SC']
138     df_maxLoad_all_new['gIE']       = df_maxLoad_all['gIE']
139
140
141 if level == 'ply' or level == 'both':
142     df_maxLoad_all_new['0MLS']      = df_maxLoad_all['0MLS']
143     df_maxLoad_all_new['90MLS']     = df_maxLoad_all['90MLS']
144     df_maxLoad_all_new['cMLS']      = df_maxLoad_all['cMLS']
145
146     df_maxLoad_all_new['0MTS']      = df_maxLoad_all['0MTS']
147     df_maxLoad_all_new['90MTS']     = df_maxLoad_all['90MTS']
148     df_maxLoad_all_new['cMTS']      = df_maxLoad_all['cMTS']
149
150     df_maxLoad_all_new['0MPS']      = df_maxLoad_all['0MPS']
151     df_maxLoad_all_new['90MPS']     = df_maxLoad_all['90MPS']
152     df_maxLoad_all_new['cMPS']      = df_maxLoad_all['cMPS']
153
154     df_maxLoad_all_new['0MSS']      = df_maxLoad_all['0MSS']
155     df_maxLoad_all_new['90MSS']     = df_maxLoad_all['90MSS']
156     df_maxLoad_all_new['cMSS']      = df_maxLoad_all['cMSS']
157
158     df_maxLoad_all_new['0MIPSS']    = df_maxLoad_all['0MIPSS']
159     df_maxLoad_all_new['90MIPSS']   = df_maxLoad_all['90MIPSS']
160
161
162 yvarMaxLoad = df_maxLoad_all_new.columns
163
164 df_maxLoad_frames = [df_input_all_new,df_maxLoad_all_new]
165 df_maxLoad = pd.concat(df_maxLoad_frames,axis=1)
166
167 if normalize:
168     df_maxLoad = min_max_scaling(df_maxLoad)
169
170 # Scatter plot
171 if scatterPlot:
172     scale='linear' # plotting on linear or log scale
173     yVarTotal = df_maxLoad_all_new.columns
174
175     # plotting for each output variable one at a time
176     for i in yVarTotal:
177         var = [*xvar,i]
178         df_maxLoad_ = df_maxLoad[var]
```

```
179
180         analysis_plot(df        = df_maxLoad_,
181                       xvar      = xvar,
182                       yvar      = [i],
183                       typ       = 1,
184                       file_name= (plot_path+f'{xvar[0]}VS{i}_ScatterPlot.png'),
185                       scale     = scale)
186
187         print(xvar, ' vs ', i, ' complete...')
188
189 if heatmapFlag:
190     x_tick= xvar
191     y_tick= yvarMaxLoad
192     total_var = [*x_tick,*y_tick]
193     heatmap(data  = df_maxLoad[total_var],
194             x_var = x_tick,
195             y_var = y_tick,
196             destination=plot_path,
197             filename='heatmap.png',
198             compact=True,
199             fig_kws={'figsize':[40,20],'dpi':150},
200             anno_kws={'annot':True,'size':17,'rotation':0},
201             cbar_kws={'label':'Correlation coefficient','label_size':30,
202                       'tick_size':19},
203             tick={'size':20,'xrotation':0},
204             cmap_='viridis')
205
206
207 if multiLinearReg:
208     # Ordinary least square (OLS)
209     # y_i = intercept + c1x1 + c2x2 + ... + cnxn
210
211     # Initializing coefficient dataframe
212     coeff = pd.DataFrame()
213
214     # RHS of multiple linear regression equation
215     # Note :intercept is added automatically in OLS
216     RHS = ' + '.join(xvar[:])
217
218     i=0
219     for yvar in yVarTotal:
220
221         # Fitting model
222         model = ols(f"{yvar} ~ {RHS}", data=df_maxLoad_all_new).fit()
223
224         # Get coefficients of the fitted model
225         coeff[yvar] = model.params
226
227         # Writing model summary
228         if i == 0:
229             typ = 'w+'
230         else:
231             typ = 'a'
232         with open(plot_path + 'MultivarRegInfo.txt',typ) as summary:
233             summary.write(
```

```
    '\nxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx' +
    '\nxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n\n'+
    str(model.summary2())
    )
        i+=1

    # Creating heatmap of coefficient of fitted data
    plt.figure(figsize=(15, 3), dpi=100)
    sns.heatmap(coeff.transpose(), cmap='coolwarm', annot=True, fmt=".3f",
                annot_kws={'size':9,'rotation':45},robust=True,
                cbar_kws={'label': 'Coefficient value'},)
    plt.savefig(plot_path + 'MultipleLinearRegressionCoefficients.png')
    plt.close()
```

A.4.7 Project Utilities

```
# ============================================================================
# Author: Apurv Kulkarni
# ----------------------------------------------------------------------------
# Contains all the utilities used in completing thesis
# ============================================================================

import numpy as np
import pandas as pd
from scipy.stats import loguniform
nan = np.nan
import os
import sys
import re
import matplotlib.pyplot as plt


#=============================================================================
# Experiment and LS-Dyna card utilities
#=============================================================================
def factor_minmax(factor_list):
    """Creates minimum and maximum values"""
    def min_max(value,variation,type_variation=10):
        value = np.asarray(value)
        if type_variation in [10,11]: # % variation
            var_val = [(1+variation/100)* value, (1-variation/100)* value]
            max_val = max(var_val)
            min_val = min(var_val)
        # elif type_variation in [40,41]: # absolute variation
        #     max_val = value + variation
        #     min_val = value - variation
        return min_val,max_val

    for i in factor_list:
        # if factor_list[i]['type'] in [10,11]:
        if 'min_val' not in factor_list[i] and 'max_val' not in factor_list[i]:
            factor_list[i]['min_val'],factor_list[i]['max_val'] = \
                min_max(factor_list[i]['value'],factor_list[i]['vari'],
                        factor_list[i]['type'])
    return factor_list
```

```
40
41 def direction_vector(theta):
42     """Angle to vector"""
43     theta = np.asarray(theta)
44     return np.round([np.cos(theta*np.pi/180),
45                      np.sin(theta*np.pi/180)],10).transpose()
46
47
48 def transform_samplespace(sample_space,factor_list,angle_keys=[]):
49     from scipy.stats import loguniform
50
51     def scale(val,new_min,new_max,scale=''):
52         if scale=='log':
53             new_min = np.log10(new_min)
54             new_max = np.log10(new_max)
55         old_min,old_max=0,1
56         old_range = old_max-old_min
57         new_range = new_max-new_min
58         scaled_val = (val-old_min) * new_range / old_range + new_min
59         if scale=='log':
60             scaled_val = 10**scaled_val
61         return scaled_val
62
63     sample_loguniform={}
64     for count,fac in enumerate(factor_list):
65         temp_fac = factor_list[fac]
66
67         # Linear scaling
68         if temp_fac['type'] in [11,21]:
69             sample_ = scale(val=sample_space[count],new_min=temp_fac['min_val'],
70                             new_max=temp_fac['max_val']).tolist()
71
72         # Log scaling
73         elif temp_fac['type'] in [10,20]:
74             if temp_fac['min_val'] < 0 and temp_fac['max_val'] < 0:
75                 temp_val = [abs(temp_fac['max_val']),abs(temp_fac['min_val'])]
76                 min_val,max_val = min(temp_val),max(temp_val)
77                 sample_ = loguniform(min_val,max_val,loc=0,
78                                      scale=1).ppf(sample_space[count])*-1
79                 sample_.tolist()
80             else:
81                 sample_ = loguniform(temp_fac['min_val'],
82                                      temp_fac['max_val'],loc=0,
83                                      scale=1).ppf(sample_space[count]).tolist()
84         # Discrete levels
85         elif temp_fac['type'] in [3]:
86             levels = temp_fac['levels']
87             num_level = len(levels)
88             chunks = np.linspace(0,1,num_level+1)
89             range_list=[]
90             for i in range(len(chunks)-1):
91                 range_list.append([chunks[i],chunks[i+1]])
92             sample_ = pd.cut(sample_space[count], bins=num_level,
93                              include_lowest=True, labels=levels).to_list()
94         sample_loguniform[fac] = sample_
```

```
95      return sample_loguniform
96
97  def format_val(array=None):
98      """Formats value in 10 digits spaces as per LS-Dyna cards
99
100     Returns
101     -------
102     Formatted value
103     """
104
105     def format_val_i(val):
106         if len(str(val)) > 10:
107             if val < 0:
108                 fval = str(format(val,'6.3E'))
109             else:
110                 fval = str(format(val,'6.4E'))
111         else:
112             fval = ' '*(10 - len(str(val))) + str(val)
113
114         return fval
115
116     formatted_dic = {}
117     for key in array:
118         val = array[key]
119         formatted_val = []
120         for val1 in val:
121             if type(val1) != np.ndarray:
122                 formatted_val.append(format_val_i(val1))
123             else:
124                 temp_vector = []
125                 for i in val1:
126                     temp_vector.append(format_val_i(i))
127                 formatted_val.append(temp_vector)
128         formatted_dic[key] = formatted_val
129     return formatted_dic
130
131
132 def create_mat_card(mat_database=None,typ=None,filename='default',meth='w+'):
133     if typ == 1:
134         m = mat_database
135         card_data = f"""$#
136 *MAT_LAMINATED_FRACTURE_DAIMLER_PINHO
137 $#     mid        ro        ea        eb        ec      prba      prca      prcb
138 {m['mid']}{m['rho']}{ m['ea']}{ m['eb']}{ m['eb']}{m['pba']}{m['pca']}{m['pcb']}
139 $#     gab       gbc       gca      aopt       daf       dkf       dmf       efs
140 {m['gab']}{m['gbc']}{m['gca']}         2       0.0       1.0       1.0{m['efs']}
141 $#      xp        yp        zp        a1        a2        a3
142        0.0       0.0       0.0{m['a1'] }{ m['a2']}       0.0
143 $#      v1        v2        v3        d1        d2        d3    mangle
144        0.0       0.0       0.0{m['d1'] }{ m['d2']}       0.0
145 $#  enkink       ena       enb       ent       enl
146 {m['enk']}{m['ena']}{m['enb']}{m['ent']}{m['enl']}
147 $#      xc        xt        yc        yt        sl
148 { m['xc']}{ m['xt']}{ m['yc']}{ m['yt']}{ m['sl']}
149 $#     fio      sigy      lcss      beta       pfl      puck      soft
```

```
{m['fio']}{m['sig']}          2     0.000   100.000     0.000     0.000
$#\n"""
    if typ == 2:
        m = mat_database
        card_data = f"""$#
*MAT_COHESIVE_GENERAL
$#     mid        ro     roflg   intfall       tes      tslc       gic       giic
{m['mid']}{m['rhC']}          1       1.0       1.0         3{m ['gi']}{m['gii']}
$#     xmu         t         s     stfsf     tslc2
       1.0{m ['t'] }{m ['s'] }       1.0       4.0
$#\n"""
    if typ==3:
        card_data = f"""$#
*CONTROL_TIMESTEP
$#  dtinit    tssfac      isdo    tslimt     dt2ms      lctm     erode     ms1st
      0.00       0.90         0      0.00-1.000E-08         0         0         0
$#  dt2msf   dt2mslc     imscl    unused    unused     rmscl
       0.0         0         0                           0.0
$#
*DEFINE_CURVE_TITLE
shear
$#    lcid      sidr       sfa       sfo      offa      offo    dattyp     lcint
         2         0     1.400     0.900       0.0       0.0         0         0
$#                a1                  o1
                  0             0e+08
              0.0007          0.0533e+08
              0.0016          0.1277e+08
              0.0028          0.1949e+08
              0.0044          0.2788e+08
              0.0067          0.3638e+08
              0.0102          0.4440e+08
              0.0137          0.5136e+08
              0.0185          0.5670e+08
              0.0243          0.6050e+08
              0.0284          0.6311e+08
              0.0328          0.6482e+08
              0.0374          0.6595e+08
              0.0423          0.6677e+08
              0.0472          0.6758e+08
              0.0522          0.6869e+08
              0.0571          0.7038e+08
              0.0621          0.7229e+08
              0.0673          0.7422e+08
              0.0727          0.7620e+08
              0.0808          0.7942e+08
              0.0946          0.8574e+08
              0.1058          0.9060e+08
              0.1179          0.9688e+08
              0.1297          1.0325e+08
              0.1426          1.0972e+08
              0.1695          1.2442e+08
$#
*DEFINE_CURVE_TITLE
fail_cohesive
$#    lcid      sidr       sfa       sfo      offa      offo    dattyp     lcint
```

```
         3         0       1.0       1.0       0.0       0.0         0         0
$#                a1                  o1
                0.0                 0.0
                0.2                 1.0
                1.0                 0.0
$#
*DEFINE_CURVE_TITLE
fail_cohesive
$#    lcid      sidr       sfa       sfo      offa      offo    dattyp     lcint
         4         0       1.0       1.0       0.0       0.0         0         0
$#                a1                  o1
                0.0                 0.0
                0.2                 1.0
                0.4                 0.9
                0.7                 0.6
                1.0                 0.0
$#"""
    return card_data


def format_val_i(val):
        if len(str(val)) > 10:
            if val < 0:
                fval = str(format(val,'6.3E'))
            else:
                fval = str(format(val,'6.4E'))
        else:
            fval = ' '*(10 - len(str(val))) + str(val)

        return fval

def dyna_output_curve(lcid=7, sfa=1, sfo=1,plot = []):
    lcid = format_val_i(lcid)
    sfa  = format_val_i(sfa)
    sfo  = format_val_i(sfo)
    curve =f"""$#
*DEFINE_CURVE_TITLE
Force
$#    lcid      sidr       sfa       sfo      offa      offo    dattyp     lcint
{  lcid  }         0{  sfa   }{  sfo   }       0.0       0.0         0         0
$#                a1                  o1
"""
    for coord in plot:
        #print("ip- ",coord[0],coord[1])
        f_x = format_val_i(coord[0])
        f_y = format_val_i(coord[1])
        #print(f_x,f_y)
        plotval = " "*10 + f_x + " "*10 + f_y + "\n"
        #print(plotval)
        curve = curve + plotval
    curve += "$#\n"
    #print(curve)

    return curve

```

```
#================================================================================
# HPC jobs submition utilities
#================================================================================
def path_format_to_linux(path):
    """Converts a windows path to linux machine path"""
    match_obj = re.search('[a-zA-Z]:',path)
    drive_letter = path[match_obj.span()[0]]
    drive_letter = drive_letter.lower()
    path = path.replace(match_obj.group(0), f'/mnt/{drive_letter}/')
    path = path.replace('\\','/')
    path = path.replace('//','/')
    return path

def creat_chain_sbatch(list_str):
    """Creates main submisison file for HPC systems"""
    s1 = f'JOBDIRS="{list_str}"'
    s2 = r"""
 DEPENDENCY=""
 PREVIOUSJOBDIR="" # is used to identify restart jobs
 for CURRENTJOBDIR in $JOBDIRS; do
   JOB_CMD="sbatch"
   if [ -n "$DEPENDENCY" ] && ! true; then
    JOB_CMD="$JOB_CMD --dependency afterany:$DEPENDENCY"
   fi
   JOB_CMD="$JOB_CMD run*.sh 2>&1"
   OUT=`cd $CURRENTJOBDIR; $JOB_CMD`
   DEPENDENCY=`echo $OUT | awk '{print $4}'`
   if [ "$CURRENTJOBDIR" == "$PREVIOUSJOBDIR" ]; then
     echo $DEPENDENCY >> $CURRENTJOBDIR/jobID
   else
   echo $DEPENDENCY > $CURRENTJOBDIR/jobID
   fi
   PREVIOUSJOBDIR=$CURRENTJOBDIR
 done
"""
    s= s1+s2
    return s


#================================================================================
# Data extraction and cleaning utilities
#================================================================================
def min_max_scaling(df,allColumns=False):
    """Scales the given dataframe :
        > columnwise (allColumns=False) or
        > on a global scale (allColumns=True)"""

    # copy the dataframe
    df_norm = df.copy()

    if allColumns == False:
        # apply min-max scaling
        for column in df_norm.columns:
            df_norm[column] = ((df_norm[column] - df_norm[column].min()) /
                               (df_norm[column].max() - df_norm[column].min()))
```

```
315     else:
316         minVal = min(df_norm.min())
317         maxVal = max(df_norm.max())
318         for column in df_norm.columns:
319             df_norm[column] = (df_norm[column] - minVal) / (maxVal - minVal)
320 
321     return df_norm
322 
323 
324 # Data extraction utilities
325 def getData(file_dst,var,dfo):
326     df = pd.read_csv(file_dst+var+'.csv' ,skiprows=7, delim_whitespace=True)
327     df.drop(df.columns[2:],axis=1, inplace=True)
328     df = df.head(-1)
329     df.columns = ['Time', var]
330     df = df.astype(float)
331     df_frames = [dfo,df[df.columns[[1]][0]]]
332     dfo = pd.concat(df_frames,axis=1)
333     return dfo
334 
335 def getDamageVarData(file,header,dfo):
336     df = pd.read_csv(file, index_col=False, skiprows=1, sep=',')
337     df = df.drop(df.columns[-1], axis=1)
338     df =  df.max(axis=1)
339     dfo[header] = df
340     return dfo
341 
342 #================================================================================
343 # Plot utilities
344 #================================================================================
345 
346 # Scatter + regression + correlation plots
347 #--------------------------------------------------------------------------------
348 def analysis_plot(df, file_name='fig.png', xvar=[], yvar=[], typ=1, title='fig',
349                   grid=True, scale='linear',r2score_flag=False,order=1,ci=False,
350                   text={'data':'','rel_xyloc':[0,0],'fontsize':19}):
351 
352     """Analysis plot used for data analysis. It plots scatter plots, regression
353        plots, and correlation values(PCC and SRCC)"""
354 
355     from scipy import stats
356     import seaborn as sns
357 
358     if typ==1:
359         def corrfunc(x, y, ax, corr_type = 'pearson',r2_score_flag=False,order=1):
360             if r2score_flag:
361                 # R-squared statistics
362                 from sklearn.metrics import r2_score
363                 import numpy as np
364                 fit = np.polyfit(df[col], df[row], order)
365                 # polynomial:
366                 # fit[0]*x^n + fit[1]*x^(n-1) + fit[2]*x^(n-2) + ...
367                 poly = np.poly1d(fit)
368                 y_pred = poly(df[col])
369                 r2_score = r2_score(df[row], y_pred)
```

```
370
371             # place a text box in upper left in axes coords (0.05,0.95)
372             props = dict(boxstyle='round', facecolor='wheat', alpha=1)
373             ax.text(0.05, 0.95,
374                     "r2={:.4f}\npoly. order={:.0f}".format(r2_score,order),
375                     transform=ax.transAxes, fontsize=14,
376                     verticalalignment='top', bbox=props)
377
378             return fit
379         else:
380             # Correlation statistics
381             pcc,_ = stats.pearsonr(x, y)
382             srcc,_ = stats.spearmanr(x,y)
383
384             # place a text box in upper left in axes coords (0.05,0.95)
385             props = dict(boxstyle='round', facecolor='wheat', alpha=1)
386             ax.text(0.05, 0.95, "PCC = {:.2f} \nSRCC = {:.2f}".format(pcc,srcc),
387                     transform=ax.transAxes, fontsize=14,
388                     verticalalignment='top', bbox=props)
389             return 0
390
391     col_len = len(xvar)
392     row_len = len(yvar)
393     sns.set_context("paper", font_scale=1)
394     fig, axes = plt.subplots(row_len, col_len, figsize=(4*col_len,3.6*row_len))
395     for row_num, row in enumerate(yvar):
396         for col_num,col in enumerate(xvar):
397             if row_len == 1 and col_len == 1:
398                 ax = axes
399             elif row_len == 1:
400                 ax = axes[col_num]
401             elif col_len==1:
402                 ax = axes[row_num]
403             else:
404                 ax = axes[row_num,col_num]
405
406             if r2score_flag:
407                 ax.scatter(df[col], df[row], alpha=0.5)
408                 fit = corrfunc(df[col], df[row], ax,
409                                r2_score_flag=True,order=order)
410                 # polynomial of
411                 # fit[0]*x^n + fit[1]*x^(n-1) + fit[2]*x^(n-2) + ...
412                 poly = np.poly1d(fit)
413                 x=np.linspace(min(df[col]),max(df[col]),num = 100)
414                 y = poly(x)
415                 ax.plot(x,y,color='black',linestyle='-',linewidth=2)
416                 ax.set_xlabel(col)
417                 ax.set_ylabel(row)
418             else:
419                 g = sns.regplot(x=df[col], y=df[row], ax=ax,
420                                 line_kws={'color': 'black'},
421                                 robust=True, ci=ci)
422                 corrfunc(df[col], df[row], g)
423                 g.set_xlabel(xlabel=col,fontsize=14)
424                 g.set_ylabel(ylabel=row,fontsize=14)
```

```
                    g.set_xscale(scale)
                    g.set_yscale(scale)

                if grid:
                    ax.grid(b=True, which='both', color='0.65', linestyle='dotted')

        fig.suptitle(title,fontsize=20,weight="bold")
        plt.tight_layout(pad=2)

        if text['data']!='':
            plt.text(x=text['rel_xyloc'][0], y=text['rel_xyloc'][1],
                     s=text['data'], fontsize=text['fontsize'], ha="left",
                     transform=fig.transFigure,wrap=True)
        # plt.subplots_adjust(top=0,bottom=0)

    elif typ==2:
        # Plots full pairplot of input and output variables
        # source: https://stackoverflow.com/questions/48139899/correlation-matrix-plot-with-coefficients-on-one-side-scatterplots-on-another
        def corrdot(*args, **kwargs):
            corr_r = args[0].corr(args[1], 'pearson')
            corr_text = f"{corr_r:2.2f}".replace("0.", ".")
            ax = plt.gca()
            ax.set_axis_off()
            marker_size = abs(corr_r) * 10000
            ax.scatter([.5], [.5], marker_size, [corr_r], alpha=0.6, cmap="coolwarm",
                       vmin=-1, vmax=1, transform=ax.transAxes)
            font_size = abs(corr_r) * 40 + 5
            ax.annotate(corr_text, [.5, .5,],  xycoords="axes fraction",
                        ha='center', va='center',fontsize=font_size)

        sns.set(style='white', font_scale=1.6)


        g = sns.PairGrid(df, aspect=1.5, diag_sharey=False, despine=False)

        g.set(xscale=scale,yscale=scale)
        g.map_lower(sns.regplot, lowess=True, ci=ci,
                    line_kws={'color': 'red', 'lw': 1},
                    scatter_kws={'color': 'black', 's': 20})
        g.map_diag(sns.distplot, kde_kws={'color': 'black'})
        g.map_upper(corrdot)

    plt.savefig(file_name)
    plt.close()

    return 0

# Heatmap for correlation values
#-------------------------------------------------------------------------------
def heatmap(data,destination,filename,x_var='',y_var='',compact=False,
            title={'label':'','size':30},
            fig_kws={'figsize':[27,27],'dpi':150},
            anno_kws={'annot':True,'size':15,'rotation':45},
            cbar_kws={'label':'cbar_label','label_size':20,'tick_size':19},
```

```
            tick={'size':19,'xrotation':0},
            cmap_='viridis',
            text={'data':'','rel_xyloc':[0,0],'fontsize':19},
            mask={'mask':False,'absThreshold':0.1},
            method='pearson'):

    import seaborn as sns
    import matplotlib.pyplot as plt

    if 'annot' not in anno_kws.keys():
        anno_kws['annot'] = False

    plt.figure(figsize=(fig_kws['figsize'][0], fig_kws['figsize'][1]),
               dpi=fig_kws['dpi'])

    # Correlation values from the dataframe
    corr_ = round(data.corr(method = method),2)

    # To plot full heatmap or only output vs input values
    if compact:
        try:corr_.drop(columns=y_var, inplace=True)
        except:pass
        try:corr_.drop(x_var, axis=0,inplace=True)
        except: pass

    # For applying filter in plots with some threshold values
    if mask['mask']:
        mask_filter1 = corr_> mask['absThreshold']
        mask_filter2 = corr_< -1 * mask['absThreshold']
        mask_filter = mask_filter1 | mask_filter2
        corr_ = corr_[mask_filter]

    if anno_kws['annot']:
        dataplot = sns.heatmap(corr_, cmap=cmap_, annot=True, fmt='.2f',
                               annot_kws={'size':anno_kws['size'],
                                          'rotation':anno_kws['rotation']},
                               cbar_kws={'label': cbar_kws['label']},
                               vmin=-1, vmax=1)
    else:
        dataplot = sns.heatmap(corr_, cmap=cmap_,
                               cbar_kws={'label': cbar_kws['label']},
                               vmin=-1, vmax=1)

    dataplot.figure.axes[-1].yaxis.label.set_size(cbar_kws['label_size'])
    dataplot.figure.axes[-1].tick_params(labelsize=cbar_kws['tick_size'])

    # dataplot.tick_params(axis='y',labelsize=tick['size'])
    plt.xticks(fontsize=tick['size'], rotation=tick['xrotation'])
    plt.yticks(fontsize=tick['size'],rotation=0)
    plt.tight_layout()

    # Plotting annotation texts (if any)
    if text['data']!='':
        plt.text(text['rel_xyloc'][0],text['rel_xyloc'][1], text['data'],
                 ha='left',fontsize=text['fontsize'],wrap=True)
```

```
534
535     plt.title(title['label'], fontsize=title['size'])
536     # plt.subplots_adjust(top=0.91,bottom=0.3)
537     file_name = destination+filename
538     plt.savefig(file_name)
539     plt.close()
540
541
542 # Fitting linear and quadratic surface for interaction plots
543 # ------------------------------------------------------------------------
544
545 def fitSurfLinearQuadratic(dataFull,xvar,yvar,file_name='',title=''):
546     import scipy.linalg
547     import pandas as pd
548
549     print(' '.join(xvar), ' vs ', yvar[0])
550
551     fig = plt.figure(figsize=(12,4))
552     fig.suptitle(title, fontsize=14)
553
554     data = dataFull[[*xvar,*yvar]]
555
556     if type(data) == pd.core.frame.DataFrame:
557         x_label = data.columns[0]
558         y_label = data.columns[1]
559         z_label = data.columns[2]
560         data = data.values
561
562     # Fitting 1st order surface
563     A1 = np.c_[data[:,0], data[:,1], np.ones(data.shape[0])]
564     C1,resid1,_,_ = scipy.linalg.lstsq(A1, data[:,2])    # coefficients
565
566     mn = np.min(data, axis=0)
567     mx = np.max(data, axis=0)
568     numPoints = 200
569     X,Y = np.meshgrid(np.linspace(mn[0], mx[0], numPoints),
570                       np.linspace(mn[1], mx[1], numPoints))
571     XX = X.flatten()
572     YY = Y.flatten()
573     Z1 = C1[0]*X + C1[1]*Y + C1[2]
574     r2 = 1 - resid1 / (Z1.size * Z1.var())
575     print('Linear surface r2 score: ',r2)
576
577     # Fitting quadratic surface
578     A2 = np.c_[np.ones(data.shape[0]), data[:,:2],
579                np.prod(data[:,:2], axis=1), data[:,:2]**2]
580     C2,resid2,_,_ = scipy.linalg.lstsq(A2, data[:,2])
581     Z2 = np.dot(np.c_[np.ones(XX.shape), XX, YY, XX*YY, XX**2, YY**2], C2).reshape(X.
    shape)
582
583     r2 = 1 - resid2 / (Z2.size * Z2.var())
584     print('Quadratic surface r2 score: ',r2)
585
586     # Figure settings
587     elev = 20   # rotation about horizontal axis
```

```
588     azim = -130 # rotation about vertical axis
589
590     # plotting linear fitted surface on LHS
591     ax = fig.add_subplot(121, projection='3d')
592     surf1 = ax.plot_surface(X, Y, Z1,cmap='viridis',rstride=1, cstride=1,
593                             linewidth=0, alpha=0.7)
594
595     # Add a color bar which maps values to colors.
596     fig.colorbar(surf1, shrink=0.5, aspect=5)
597     ax.scatter(data[:,0], data[:,1], data[:,2], c='g', s=5)
598     plt.xlabel(x_label)
599     plt.ylabel(y_label)
600     ax.set_zlabel(z_label)
601     ax.title.set_text('Linear surface fit')
602     ax.axis('auto')
603     ax.axis('tight')
604     ax.elev = elev
605     ax.azim = azim
606
607     # plotting quadratic surface on RHS
608     ax = fig.add_subplot(122, projection='3d')
609     surf2 = ax.plot_surface(X, Y, Z2,cmap='viridis', rstride=1, cstride=1,
610                             linewidth=0, alpha=0.7)
611
612     # Add a color bar which maps values to colors.
613     fig.colorbar(surf2, shrink=0.5, aspect=5)
614     ax.scatter(data[:,0], data[:,1], data[:,2], c='g', s=5)
615     plt.xlabel(x_label)
616     plt.ylabel(y_label)
617     ax.set_zlim3d(0, 1)
618     ax.set_ylim3d(0, 1)
619     ax.set_xlim3d(0, 1)
620     ax.set_zlabel(z_label)
621     ax.title.set_text('Quadratic surface fit')
622     ax.axis('auto')
623     ax.axis('tight')
624     ax.elev = elev
625     ax.azim = azim
626
627     plt.savefig(file_name)
628     plt.close()
```

A.4.8 LS-Prepost Macro Utilities

```
1  # ================================================================================
2  # Author: Apurv Kulkarni
3  # --------------------------------------------------------------------------------
4  # Contains LS-PrePost macros used in completing thesis
5  # ================================================================================
6
7  import os
8  def write_bash(start_id,end_id,macro_destination,lspp_loc='/home/apku868a/lsprepost4
       .8_common/lspp48',file_suffix='',macro_suffix=''):
9      """Creates bash script containig code to run marcros using LS-PrePost application
       .
```

```
    Args:
        start_id (int): start id of the experiment
        end_id (int): last id of the experiment
        macro_destination (str): location of the macro to be saved
        lspp_loc (str, optional): Location of ls-prepost app that will be called to run macro. Defaults to '/home/apku868a/lsprepost4.8_common/lspp48'.
        file_suffix (str, optional): suffix of the bash script filename. Defaults to ''.
        macro_suffix (str, optional): suffix of the macro files. Defaults to ''.
    """

    # Adding preamble
    s_temp = f"{start_id}..{end_id}"
    s = f"""#!/bin/bash
for i in {{{s_temp}}}
do
"""
    # Adding command to run macros
    s+=f"""    echo ########################  $i{macro_suffix}.cfile
    ########################
echo "$i{macro_suffix}.cfile - running" > {start_id}_{end_id}.info
{lspp_loc} -nographics c="$i{macro_suffix}.cfile"
done
"""
    # Creating bash script
    batch_macro_name = macro_destination+f'batch_LSPost{file_suffix}.sh'
    if os.path.isfile(batch_macro_name):
        os.remove(batch_macro_name)
    with open(batch_macro_name,'a+') as mybatch:
        mybatch.write(s)
    return 0


def macro_wrapper(commands,file_loc):
    """ Adds initla lintes and ending lines to macro data. LAso adds line that opens d3Plot file.

    Args:
        commands (str): All the macro commands
        file_loc (str): d3Plot file location / Results location.
    """
    s1 = f"""*lsprepost macro command file
*macro begin macro_post\n
$# ============================
open d3plot "{file_loc}d3plot"\n"""
    s2 = "\n*macro end"

    return s1 + commands + s2


def newCardFormat(macro_file,dyna_card_loc,lspp_loc):
    save_macro = f"""*lsprepost macro command file
*macro begin macro_post
openc keyword "{dyna_card_loc}"

```

```
elemedit createnode accept
genselect target node
elemedit delenode delete
elemedit delenode accept 1
Build Rendering data
genselect clear

save keywordabsolute 0
save keywordbylongfmt 0
save keywordbyi10fmt 0
save outversion 10
save keyword specialsubsystem "{dyna_card_loc}" 1
*macro end
"""

    with open(macro_file,'w+') as mysave:
        mysave.write(save_macro)

    run_cmd = f'{lspp_loc} -nographics c="{macro_file}"' + '\n'

    return run_cmd


def global_bodyforce_100(filename,csv_destination):
    """Model is cut at plane:
    origin:  0.12 0.042 0.00093
    normals: 1.000  0.000  0.000

    This extracts following components from the model cut from above plane:
        3  - resultant force,
        4  - normal (body) force (x),
        5  - shear (body) force (y),
        13 - cs area

    Args:
        csv_destination (str): Desitnation path for output database csv
        name (str): output csv file name.

    Returns:
        str: string with macro for data extraction
    """
    s = f"""splane dep0 0.12 0.042 0.00093  1.000  0.000  0.000
splane setbasept 0.12 0.042 0.00093
splane drawcut
$# Extracting: resultant force, x-force,y-force,crossection area
splane xsection 3 4 5 13
xyplot 1 savefile ms_csv "{csv_destination}{filename}.csv" 1 all
xyplot 1 donemenu
deletewin 1
splane done
"""
    return s

def elem_force(elem_id,name,csv_destination):
    """Extracting forces in an element.
```

```
    1- x-force
    2- y-force
    3- Resultant force

    Args:
        elem_id (int): Element ID
        name (str): Output data filename
        csv_destination (str): Output data file location
    """
    s = f"""
genselect target element
genselect element add element {elem_id}/0
etime 1
addplot
etime 2
addplot
etime 3
xyplot 1 savefile ms_csv "{csv_destination}{name}.csv" 1 all
clearpick
"""
    return s

def global_internal_energy(filename,csv_destination):
    """Extracts the global internal energy data

    Args:
        filename (str): Output data filename
        csv_destination (str): Output data file location
    """
    s = f"""gtime 2
xyplot 1 savefile ms_csv_multiple "{csv_destination}{filename}.csv" 1 all

"""
    return s


def get_history_data(part_id,val,filename,destination_csv,data_file_suffix='',
    curve_operation_id=0):
    """Get part data

    Args:
        part_id (int): Part ID whose data needs to be extracted
        val (int): Data ID that needs to be extracted.
                    1 - Longitudinal stress (x-stress)
                    2 - Transverse stress (y-stress)
                    4 - In-plane stress (xy-stress)
                    9 - Effective stress
                   13 - Max shear stress (Von Tresca)
                   14 - Principle stress

        filename (str): Output data filename
        destination_csv (str): Output data file location
        data_file_suffix (str, optional): Suffix of data filename. Defaults to ''.
        curve_operation_id (int): Operations to be performed on the collected time
    curve. Defaults to 0 (maximum curve).
```

```
                                    0 - Maximum of all extracted curves
                                    1 - Average of all extracted curves
    """
    assert(type(part_id)==list)
    s_heading = f"$# {filename} =================================\n"
    s0 = """\ngenselect clear all\n"""
    s1 = '\n'
    for part in part_id:
        s1 += f"""genselect part add part {part}/0""" + '\n'

    if curve_operation_id==0:
        curve_operation = 'max_curve'
    elif curve_operation_id==0:
        curve_operation = 'average_curves'

    s2 = f"""
maxvalue {val}
xyplot 1 select all
xyplot 1 operation {curve_operation} all
xyplot 1 operation save "{destination_csv}{filename}{data_file_suffix}.csv" MSoft_CSV
deletewin 1
"""
    s = s_heading + s0+s1+s2+s0+"\n"
    return s



def get_fringe_data(state_id,fringe_val,macro_dest,part_ID=[]):
    """Extract fringe data elementwise for selected layer

    Args:
        state_id (list): <[lower id, upper id]> - Time steps to extract element level
     data
        fringe_val ([type]): fringe value that needs to be extracted.
                    1 - Longitudinal stress (x-stress)
                    2 - Transverse stress (y-stress)
                    4 - In-plane stress (xy-stress)
                    9 - Effective stress
                   13 - Max shear stress (Von Tresca)
                   14 - Principle stress
                   81 - fibre damage
                   83 - matrix damage
                   85 - damage variable fibres
                   87 - damage variable matrix
        macro_dest ([type]): [description]
        part (str, optional): [description]. Defaults to 'all'.
    """
    if part_ID==[]:
        s0="postmodel off\npostmodel on \n"
    else:
        s0="postmodel off\n+M "
        for i in part_ID:
            s0 += ' ' + str(i)
        s0 += ' \n'

```

```
223     s1 = f"fringe {fringe_val}\npfringe\n"
224 
225     list_ = list(range(state_id[0],state_id[1]+1))
226     s2 = ''
227     for i in list_:
228         s2 += f"""output "{macro_dest}{i}.csv" {i} 1 0 1 0 0 0 0 1 0 0 0 0 0 0
    1.000000\n"""
229     s = s0+s1+s2
230     return s
231 
232 
233 def nodal_data_disp(nodeid,filename,destination_csv):
234     """Extracting nodal displacement
235 
236     ntime 5 - Resultant displacement
237     ntime 6 - x displacement
238     ntime 7 - y displacement
239     ntime 8 - z displacement
240 
241     Args:
242         nodeid (int): Node ID
243         filename (str): Name of output data file
244         destination_csv (str): Location of output data file
245     """
246     s = f"""
247 $# {filename}
248 genselect node add node {nodeid}/0
249 ntime 5
250 addplot
251 ntime 6
252 addplot
253 ntime 7
254 addplot
255 ntime 8
256 xyplot 1 select all
257 xyplot 1 savefile ms_csv "{destination_csv}{filename}.csv" 1 all
258 deletewin 1
259 clearpick
260 """
261     return s
262 
263 def getNotchElemDataGlobal(csv_dst):
264     """Extracting element data near notches
265 
266     Args:
267         csv_dst (str): Output datafile location
268     """
269 
270     # range of coordinates over which element values are extracted
271     x1 = 0.118
272     x2 = 0.122
273 
274     y1 = 0.0115
275     y2 = 0.0175
276 
```

```
    s = f"""
$# Notch element data

postmodel off
postmodel on

genselect target element solid
switch2selection
clearpick
genselect target element solid

genselect element add plane in 0.119865 0.0105 7.91667e-05 0 0 1 0.01 0
genselect element remove plane in 0.12 {y1} 0 0 -1 0 1 1
genselect element remove plane in 0.12 {y2} 0 0 1 0 1 1
genselect element remove plane in {x1} 0.0144 0 -1 0 0 1 1
genselect element remove plane in {x2} 0.0144 0 1 0 0 1 1

maxvalue 1
xyplot 1 operation max_curve all
xyplot 1 savefile ms_csv "{csv_dst}gNotchElem_xStress_max.csv" 1 all
deletewin 1

maxvalue 2
xyplot 1 operation max_curve all
xyplot 1 savefile ms_csv "{csv_dst}gNotchElem_yStress_max.csv" 1 all
deletewin 1

maxvalue 14
xyplot 1 operation max_curve all
xyplot 1 savefile ms_csv "{csv_dst}gNotchElem_maxPrinc_max.csv" 1 all
deletewin 1

maxvalue 81
xyplot 1 operation max_curve all
xyplot 1 savefile ms_csv "{csv_dst}gNotchElem_fa_max.csv" 1 all
deletewin 1

maxvalue 83
xyplot 1 operation max_curve all
xyplot 1 savefile ms_csv "{csv_dst}gNotchElem_fmat_max.csv" 1 all
deletewin 1

maxvalue 85
xyplot 1 operation max_curve all
xyplot 1 savefile ms_csv "{csv_dst}gNotchElem_da_max.csv" 1 all
deletewin 1

maxvalue 87
xyplot 1 operation max_curve all
xyplot 1 savefile ms_csv "{csv_dst}gNotchElem_dmat.csv" 1 all
deletewin 1

etime 87
xyplot 1 operation average_curves all
xyplot 1 savefile ms_csv "{csv_dst}gNotchElem_dmat_avg.csv" 1 all
```

```
deletewin 1

clearpick
postmodel on
"""
    return s
```

A.4.9 Mesh Utilities

```
#=============================================================================
# Author: Apurv Kulkarni
#-----------------------------------------------------------------------------
# OHT Analysis Geometry
#-----------------------------------------------------------------------------
# Creating OHT geometry using gmsh api. The first layer is created using gmsh
# api. The subsequent plies and cohesive layers are created using python loops.
# The cohesive elements are zero thickness elements with 8 nodes. 4-nodes from
# bottom and top are shared with plies respectively
#=============================================================================
# Geometry and Mesh controls|
#---------------------------
#                        <numMainbodyXElem>    <numMainbodyXElem>
#     <numClampXElem>       ---------                ---------
#     ------------------------------------------------------------------------
#       |                   |         |\        /|         |                  |
#   |   |                   |         | \      / |         |                  |
#   |   |                   |         |  \____/--|-    <numCSElem>             |
#   |   |                   |         |  /    \  |         |                  |
# Ly    |                   |         |  \____/  |         |                  |
#   |   |                   |         |  /  | \  |         |                  |
#   |   |                   |         | /       \ |         |                  |
#   |   |                   |         |<numCurveElem>       |                  |
#       ------------------------------------------------------------------------
#        -------clp-------                                        -------clp-------
#        ----------------------------- Lx ---------------------------------
#
#=============================================================================

def create_geo(Rx=0.0024,file_dst="D:\\",filename='sample_temp',
                onlypreview=0, save_mesh=0,meshname='mymesh',
                preview_geom_mesh=0,log_flag=0, numCurveElem=25,
                numClampXElem =4, numMainbodyXElem= 10, numCSElem = 15):
    """
   Args:
        Rx (float, optional): Longitudinal radius of elliptical hole.
        Defaults to 0.0024.

        file_dst (str, optional): Destination path of generated geometry file.
        Defaults to "D:\".

        filename (str, optional): Filename of generated geometry file.
        Defaults to 'sample_temp'.

        onlypreview (int, optional): Only previews the geometry in GMSH
        application no saving. Defaults to 0.
```

```
47
48          save_mesh (int, optional): Flag to save the mesh (different from the
49          LS-Dyna geometry and mesh). Defaults to 0.
50
51          meshname (str, optional): Mesh name. Defaults to 'mymesh'.
52
53          preview_geom_mesh (int, optional): Previews the gemetry (and mesh) then
54          saves it. User needs to close preview window manually to proceed.
55          Defaults to 0.
56
57          log_flag (int, optional): Whethre to generate GMSH log or not.
58          Defaults to 0.
59
60          numCurveElem (int, optional): Mesh control- Number of elements on the
61          hole circumference (in each of the 4 sectors). Defaults to 25.
62
63          numClampXElem (int, optional): Mesh control- Number of element on
64          clamping area in x-direction. Defaults to 4.
65
66          numMainbodyXElem (int, optional): Mesh control- Number of element on
67          mainbody area in x-direction. Defaults to 10.
68
69          numCSElem (int, optional): Mesh control- Number of elements in cross
70          sectional part of notch mesh. Defaults to 15.
71      """
72      # Importing utilities
73      import gmsh                # install the api using => "pip install gmsh"
74      import numpy as np
75      np.set_printoptions(precision=20)
76      from time import process_time
77
78      try:from mt_x_mainKeywordString import mainFile
79      except:from utilities.mt_x_mainKeywordString import mainFile
80
81      import subprocess
82      import os
83      import time
84
85      start =  process_time()
86
87      # Model geometry details
88      Lx = 0.240               # Length of geometry in x-direction
89      Ly = 0.0288              # Width of geometry in y-direction
90      Lz = 0.0019              # Thickness of geometry in z-direction
91      clp = 0.05               # Claming distance
92      numPly=12                # Number of composite material layers(plies)
93      LzPly = Lz/numPly        # Thickness of each ply
94      numCohElem = numPly-1    # Number of cohesive element layers (numPly-1)
95      Rx=Rx                    # Radius of elliptical notch in x-direction
96      Ry=0.0024                # Radius of elliptical notch in y-direction
97      MeshSizeFactor = 1       # Resulatant = MeshSize * meshSizeFactor
98
99      os.makedirs(file_dst, exist_ok=True)
100
101     # lsprepost application location (if not in path)
```

```
102     lspp_loc = "C:\Program Files\LSTC\LS-PrePost 4.8\lsprepost4.8_x64.exe"
103
104     # starting gmsh
105     gmsh.initialize()
106     # logging gmsh geometry and mesh creation process
107     if log_flag:
108         gmsh.logger.start()
109
110     gmsh.model.add("mymodel")
111     gmsh.option.setNumber('Geometry.CopyMeshingMethod',1);
112
113     #=====================================================================
114     #  Geometry creation
115     #=====================================================================
116
117     # Finding intersection point of a line passing through ellipse center and
118     # ellipse
119     a = Rx
120     b = Ry
121     if Rx>Ry:
122         m = Ry/Rx*(0.8)  # slope of the line
123     if Rx==Ry:
124         m = 1
125     else:
126         # m = 1
127         m = Ry/Rx*2     # slope of the line
128
129     c = Ly/2 - (m * Lx/2)
130     h = Lx/2
131     k = Ly/2
132     phi = c - k
133
134     e1x1 = (((b*b*h)-a*a*m*phi+
135             a*b*np.sqrt((b*b)+(a*a*m*m)-(2*m*phi*h)-(phi*phi)-(m*m*h*h)))/
136             (a*a*m*m + b*b))
137
138     e1y1 = m*e1x1 + c
139     e1x2 = (((b*b*h) - a*a*m*phi -
140              a*b*np.sqrt(b*b + a*a*m*m - 2*m*phi*h - phi*phi - m*m*h*h))/
141             (a*a*m*m + b*b))
142
143     e1y2 = m*e1x2 + c
144
145     e2x1 = 0 + (Lx/2-Ly/2)
146     e2y1 = 0
147     e2x2 = Ly+(Lx/2-Ly/2)
148     e2y2 = Ly
149
150     # Creating points in the space
151     gmsh.model.occ.addPoint(0,  0, 0, 1.0)          #1
152     gmsh.model.occ.addPoint(clp,  0, 0, 1.0)        #2
153     gmsh.model.occ.addPoint(Lx-clp,  0, 0, 1.0)     #3
154     gmsh.model.occ.addPoint(Lx,  0, 0, 1.0)         #4
155     gmsh.model.occ.addPoint(0, Ly, 0, 1.0)          #5
156     gmsh.model.occ.addPoint(clp, Ly, 0, 1.0)        #6
```

```
157     gmsh.model.occ.addPoint(Lx-clp, Ly, 0, 1.0)      #7
158     gmsh.model.occ.addPoint(Lx, Ly, 0, 1.0)          #8
159     gmsh.model.occ.addPoint(e1x2, e1y2, 0, 1.0)      #9
160     gmsh.model.occ.addPoint(e1x1, e1y2, 0, 1.0)      #10
161     gmsh.model.occ.addPoint(e1x1, e1y1, 0, 1.0)      #11
162     gmsh.model.occ.addPoint(e1x2, e1y1, 0, 1.0)      #12
163     gmsh.model.occ.addPoint(e2x2, e2y2, 0, 1.0)      #13
164     gmsh.model.occ.addPoint(e2x1, e2y2, 0, 1.0)      #14
165     gmsh.model.occ.addPoint(e2x1, e2y1, 0, 1.0)      #15
166     gmsh.model.occ.addPoint(e2x2, e2y1, 0, 1.0)      #16
167
168
169     if Rx>=Ry:
170         gmsh.model.occ.addPoint(Lx/2, Ly, 0, 1.0)    #17
171     else:
172         gmsh.model.occ.addPoint(0, Ly/2, 0, 1.0)     #18
173         gmsh.model.occ.addPoint(clp, Ly/2, 0, 1.0)   #19
174         gmsh.model.occ.addPoint(e2x1, Ly/2, 0, 1.0) #20
175
176     ClampXElem = []
177     MainbodyXElem = []
178     CSElem = []
179     CurveElem =  []
180     Curve2 = []
181
182     # Creating lines by joining points
183     if Rx>=Ry:
184         CurveElem.append(gmsh.model.occ.addLine(1,5))
185         CurveElem.append(gmsh.model.occ.addLine(2,6))
186         CurveElem.append(gmsh.model.occ.addLine(3,7))
187         CurveElem.append(gmsh.model.occ.addLine(4,8))
188         CurveElem.append(gmsh.model.occ.addLine(14,15))
189         CurveElem.append(gmsh.model.occ.addLine(13,16))
190         CurveElem.append(gmsh.model.occ.addLine(15,16))
191
192         ClampXElem.append(gmsh.model.occ.addLine(1,2))
193         ClampXElem.append(gmsh.model.occ.addLine(3,4))
194         ClampXElem.append(gmsh.model.occ.addLine(5,6))
195         ClampXElem.append(gmsh.model.occ.addLine(7,8))
196
197         MainbodyXElem.append(gmsh.model.occ.addLine(2,15))
198         MainbodyXElem.append(gmsh.model.occ.addLine(16,3))
199         MainbodyXElem.append(gmsh.model.occ.addLine(6,14))
200         MainbodyXElem.append(gmsh.model.occ.addLine(13,7))
201
202         CSElem.append(gmsh.model.occ.addLine(12,14))
203         CSElem.append(gmsh.model.occ.addLine(9,15))
204         CSElem.append(gmsh.model.occ.addLine(10,16))
205         CSElem.append(gmsh.model.occ.addLine(11,13))
206
207         Curve2.append(gmsh.model.occ.addLine(14,17))
208         Curve2.append(gmsh.model.occ.addLine(17,13))
209
210     if Rx<Ry:
211         ClampXElem.append(gmsh.model.occ.addLine(1,2))
```

```
212         ClampXElem.append(gmsh.model.occ.addLine(3,4))
213         ClampXElem.append(gmsh.model.occ.addLine(5,6))
214         ClampXElem.append(gmsh.model.occ.addLine(7,8))
215 
216         MainbodyXElem.append(gmsh.model.occ.addLine(2,15))
217         MainbodyXElem.append(gmsh.model.occ.addLine(16,3))
218         MainbodyXElem.append(gmsh.model.occ.addLine(6,14))
219         MainbodyXElem.append(gmsh.model.occ.addLine(13,7))
220 
221         CSElem.append(gmsh.model.occ.addLine(12,14))
222         CSElem.append(gmsh.model.occ.addLine(9,15))
223         CSElem.append(gmsh.model.occ.addLine(10,16))
224         CSElem.append(gmsh.model.occ.addLine(11,13))
225 
226         CurveElem.append(gmsh.model.occ.addLine(3,7))
227         CurveElem.append(gmsh.model.occ.addLine(4,8))
228         CurveElem.append(gmsh.model.occ.addLine(13,16))
229         CurveElem.append(gmsh.model.occ.addLine(13,14))
230         CurveElem.append(gmsh.model.occ.addLine(15,16))
231 
232         Curve2.append(gmsh.model.occ.addLine(1,17))
233         Curve2.append(gmsh.model.occ.addLine(17,5))
234         Curve2.append(gmsh.model.occ.addLine(2,18))
235         Curve2.append(gmsh.model.occ.addLine(18,6))
236         Curve2.append(gmsh.model.occ.addLine(15,19))
237         Curve2.append(gmsh.model.occ.addLine(19,14))
238 
239     gmsh.model.occ.synchronize()
240 
241     # Creating ellipse
242     mellipse = np.pi/2
243     if Rx>=Ry:
244         ellipse = gmsh.model.occ.addEllipse(Lx/2,Ly/2,0,Rx,Ry,angle1=mellipse,
245                                             angle2=2*np.pi+mellipse)
246     else:
247         ellipse = gmsh.model.occ.addEllipse(Lx/2,Ly/2,0,Ry,Rx,angle1=mellipse,
248                                             angle2=2*np.pi+mellipse)
249         gmsh.model.occ.rotate([(1,ellipse)], Lx/2, Ly/2, 0, 0, 0, 1, np.pi/2)
250     gmsh.model.occ.synchronize()
251 
252     # Splitting ellipse using lines across ellipse
253     cutOut = gmsh.model.occ.cut([(1,ellipse)],
254                                 [(1,CSElem[0]),(1,CSElem[1]),(1,CSElem[2]),
255                                  (1,CSElem[3])],removeTool=(False))
256 
257     for i in range(1,len(cutOut[0])-1):
258         CurveElem.append(cutOut[0][i][1])
259     Curve2.append(cutOut[0][0][1])
260     Curve2.append(cutOut[0][-1][1])
261 
262     gmsh.model.occ.synchronize()
263 
264     # Surface groups : Grouping different lines to form closed surface
265     # boundaries.
266     sTag_list = []
```

```
267
268     if Rx>=Ry:
269         linegroup = [
270             [1,8,2,10],[14,2,12,5],[6,13,3,15],[3,9,4,11],
271             [17,7,18,24],[25,18,6,19],[16,5,17,23]
272             ]
273     else:
274         linegroup = [
275             [6,13,8,15],[13,2,14,4],
276             [17,11,25,10],[11,15,12,26],[12,16,9,27]
277             ]
278     for surface in linegroup:
279         lTag = gmsh.model.occ.add_wire(surface)
280         sTag = gmsh.model.occ.add_plane_surface([lTag])
281         sTag_list.append(sTag)
282         gmsh.model.occ.synchronize()
283
284     # Setting transfinite curves for structured mesh
285     for i in ClampXElem:
286         gmsh.model.mesh.setTransfiniteCurve(i,numClampXElem)
287         gmsh.model.occ.synchronize()
288
289     for i in MainbodyXElem:
290         gmsh.model.mesh.setTransfiniteCurve(i,numMainbodyXElem)
291         gmsh.model.occ.synchronize()
292
293
294     for i in CSElem:
295         gmsh.model.mesh.setTransfiniteCurve(i,numCSElem)
296         gmsh.model.occ.synchronize()
297
298     for i in CurveElem:
299         gmsh.model.mesh.setTransfiniteCurve(i,numCurveElem)
300         gmsh.model.occ.synchronize()
301
302     for i in Curve2:
303         gmsh.model.mesh.setTransfiniteCurve(i,int(numCurveElem/2))
304         gmsh.model.occ.synchronize()
305
306     # Setting tranfinite surfaces for structured mesh
307     for i in sTag_list:
308         gmsh.model.mesh.setTransfiniteSurface(i)
309         gmsh.model.occ.synchronize()
310
311     # surface groups : Grouping different lines to form closed surface
312     # boundaries (with more than 4 points/lines)
313     if Rx>=Ry:
314         lTag = gmsh.model.occ.add_wire([19,21,20,16,22,26])
315         gmsh.model.occ.synchronize()
316         sTag = gmsh.model.occ.add_plane_surface([lTag])
317         gmsh.model.occ.synchronize()
318         gmsh.model.mesh.setTransfiniteSurface(tag=sTag,cornerTags=[13,14,12,11])
319         gmsh.model.occ.synchronize()
320
321     elif Rx<Ry:
```

```
322         lTag = gmsh.model.occ.add_wire([1,20,21,3,19,18])
323         sTag = gmsh.model.occ.add_plane_surface([lTag])
324         gmsh.model.occ.synchronize()
325         gmsh.model.mesh.setTransfiniteSurface(tag=sTag,cornerTags=[1,2,6,5])
326         gmsh.model.occ.synchronize()
327 
328         lTag = gmsh.model.occ.add_wire([5,22,23,7,21,20])
329         sTag = gmsh.model.occ.add_plane_surface([lTag])
330         gmsh.model.occ.synchronize()
331         gmsh.model.mesh.setTransfiniteSurface(tag=sTag,cornerTags=[2,15,14,6])
332         gmsh.model.occ.synchronize()
333 
334         lTag = gmsh.model.occ.add_wire([9,23,22,10,24,28])
335         sTag = gmsh.model.occ.add_plane_surface([lTag])
336         gmsh.model.occ.synchronize()
337         gmsh.model.mesh.setTransfiniteSurface(tag=sTag,cornerTags=[15,9,12,14])
338         gmsh.model.occ.synchronize()
339 
340     gmsh.model.mesh.recombine()
341 
342     # Extrude: Adding thickness to create a singly ply.
343     # Number of elements in thickness direction
344     numElemThickness = 3
345     model_ = gmsh.model.getEntities(2)
346     gmsh.model.occ.synchronize()
347     gmsh.model.occ.extrude(model_,0,0,LzPly,numElements=[numElemThickness],
348                            heights=[1],recombine=True)
349     gmsh.model.occ.synchronize()
350 
351     #=====================================================================
352     # Meshing
353     #=====================================================================
354     # Mesh options
355 
356     gmsh.option.setNumber("Mesh.Smoothing", 100)
357 
358     # 2D mesh algorithm (1: MeshAdapt, 2: Automatic, 3: Initial mesh only,
359     # 5: Delaunay, 6: Frontal-Delaunay, 7: BAMG, 8: Frontal-Delaunay for Quads,
360     # 9: Packing of Parallelograms)
361     meshalgo2d = 8
362     gmsh.option.setNumber("Mesh.Algorithm",meshalgo2d)
363 
364     # Recombine all triangular meshes? (yes:1, no:0)
365     RecombineTriMesh = 1
366     gmsh.option.setNumber("Mesh.RecombineAll",RecombineTriMesh)
367     gmsh.option.setNumber("Mesh.MeshSizeFactor",MeshSizeFactor)
368 
369     # Generating mesh
370     gmsh.model.mesh.clear()
371     # Meshing 2D
372     gmsh.model.mesh.generate(2)
373     # Meshing 3D
374     gmsh.model.mesh.generate(3)
375 
376     # gmsh mesh name without extension # save mesh (yes-1,no-0)
```

```
377     if save_mesh:
378         gmsh.write(f"{meshname}.msh")
379 
380     if onlypreview==0:
381         # Get nodes and their coordinates
382         nodeTags, nodeCoords, _ = gmsh.model.mesh.getNodes()
383         # Type,number of elements,
384         elementTypes, elementTags, elementNodeTags = gmsh.model.mesh.getElements(3)
385         elementTypes = elementTypes[0]
386         elementTags = elementTags[0]
387         elementNodeTags = elementNodeTags[0]
388 
389     # Launch the GUI to to preview geometry and mesh using gmsh applications.
390     # Script pauses untill gmsh application is closed manually
391     if preview_geom_mesh:
392         gmsh.fltk.run()
393 
394     if log_flag:
395         log = gmsh.logger.get()
396         gmsh.logger.stop()
397 
398     # close gmsh
399     gmsh.finalize()
400 
401     if onlypreview == 0:
402         #=====================================================================
403         # Data Extraction: processing data collected from gmsh
404         #=====================================================================
405         numElem = len(elementTags)
406         if meshalgo2d==8 or RecombineTriMesh==1:                                #
     processing based on shape of solid elements in mesh
407             elemtonodes = np.reshape(elementNodeTags,(-1,8))                    #
     quad/hex elements -> 8 unique coordinates
408         else:
409             elemtonodes = np.reshape(elementNodeTags,(-1,4))                    #
     tetra hedral elements -> 4 unique coordinates
410             last4nodes = np.transpose(np.asarray([elemtonodes[:,3]]))           #
     when represented in terms of hexahedral coordinates, the last node is repeated 4
     times -> resulting in 8 nodes
411             last4nodes = np.repeat(last4nodes,4,axis=1)
412             elemtonodes = np.concatenate([elemtonodes,last4nodes],axis=1)
413 
414         node_coord = np.round(np.transpose(np.vstack([nodeCoords[0::3],nodeCoords
    [1::3],nodeCoords[2::3]])),16)
415         numNode = len(node_coord)
416         assert(len(node_coord) == len(nodeTags))
417         assert(len(elemtonodes) == numElem)
418         print('Total number of elements in 1st ply:\t',numElem)
419         print('Total number of nodes in 1st ply:\t',numNode)
420         print('Mesh generation of 1st ply complete...')
421 
422         #=====================================================================
423         # Building full model
424         #=====================================================================
425         # Add 1st layer mesh data to the database dictionary 'mesh_data'
```

```
426         #-----------------------------------------------------------------------
427         mesh_data = {}
428         mesh_data[1]={}
429         mesh_data[1]['nodeCoord'] = np.copy(node_coord)
430         mesh_data[1]['elemNode'] = np.copy(elemtonodes)
431         mesh_data[1]['elemidx'] = np.array(list(range(1,numElem+1))).astype('int64')
432         mesh_data[1]['nodeidx'] = np.copy(nodeTags.astype('int64'))
433 
434         # To assert that there no duplicate elements sharing same nodes
435         assert(len(mesh_data[1]['elemNode']) ==
436                len(np.unique(mesh_data[1]['elemNode'],axis=0)))
437 
438         # Adding other layer mesh data to the database dictionary 'mesh_data'
439         #-----------------------------------------------------------------------
440         for ply in range(2,numPly+1):
441             mesh_data[ply]={}
442             # Adding thickness to all z coordinates
443             new_z = mesh_data[ply-1]['nodeCoord'][:,2] + LzPly
444             mesh_data[ply]['nodeCoord'] = np.transpose(np.vstack([mesh_data[ply-1]['
    nodeCoord'][:,0],mesh_data[ply-1]['nodeCoord'][:,1],new_z]))
445             mesh_data[ply]['nodeCoord'] = np.round(mesh_data[ply]['nodeCoord'],16)
446             mesh_data[ply]['elemidx'] = np.arange( max(mesh_data[ply-1]['elemidx'])
    +1, max(mesh_data[ply-1]['elemidx'])+1+numElem )
447             mesh_data[ply]['nodeidx'] = np.arange( max(mesh_data[ply-1]['nodeidx'])
    +1, max(mesh_data[ply-1]['nodeidx'])+1+numNode )
448             mesh_data[ply]['elemNode'] = np.copy(mesh_data[ply-1]['elemNode'] +
    numNode)
449 
450             # To assert that there no duplicate elements sharing same nodes
451             assert(len(mesh_data[ply]['elemNode']) == len(np.unique(mesh_data[ply]['
    elemNode'],axis=0)))
452         print('Mesh generation ply complete...')
453 
454         # Adding cohesive layers mesh data to the database dictionary 'mesh_data'
455         #-----------------------------------------------------------------------
456         mesh_data[numPly+1] = {}
457         elemnode_map = {}
458         for layer in range(1,numCohElem+1):
459             lower_ply = layer
460             upper_ply = layer+1
461 
462             # nodes on upper surface of lower ply
463             lower_nodes_lidx = np.where(mesh_data[lower_ply]['nodeCoord'][:,2] == max
    (mesh_data[lower_ply]['nodeCoord'][:,2]))[0]
464             # Elemetonode uses global node index. Hence converting them back
465             lower_node_gidx = mesh_data[lower_ply]['nodeidx'][lower_nodes_lidx]
466             LowerPlyElements_lidx = []
467 
468             temp1 = np.in1d(mesh_data[lower_ply]['elemNode'][:,4],lower_node_gidx)
469             temp2 = np.in1d(mesh_data[lower_ply]['elemNode'][:,5],lower_node_gidx)
470             temp3 = np.in1d(mesh_data[lower_ply]['elemNode'][:,6],lower_node_gidx)
471             temp4 = np.in1d(mesh_data[lower_ply]['elemNode'][:,7],lower_node_gidx)
472             temp = temp1*temp2*temp3*temp4
473             LowerPlyElements_lidx = np.where(temp)
```

```
474             lowerply_node_gidx = mesh_data[lower_ply]['elemNode'][
    LowerPlyElements_lidx,4:][0]
475 
476             vround = 10
477             if layer==1:
478                 upperply_elemnode_list = []
479                 for nodecoord_i in lowerply_node_gidx:
480                     for nodei in nodecoord_i:
481                         nodei_lidx = np.where(mesh_data[lower_ply]['nodeidx']==nodei)
    [0][0] # in local idx
482 
483                         # getting node with similar coordinate as "nodei" in local
    index
484                         node_x = np.where(np.round(mesh_data[upper_ply]['nodeCoord'
    ][:,0],vround)== np.round(mesh_data[lower_ply]['nodeCoord'][nodei_lidx][0],vround
    ))[0]
485                         node_y = node_x[np.where(np.round(mesh_data[upper_ply]['
    nodeCoord'][node_x,1],vround)== np.round(mesh_data[lower_ply]['nodeCoord'][
    nodei_lidx][1],vround))[0]]
486                         node_z = node_y[np.where(np.round(mesh_data[upper_ply]['
    nodeCoord'][node_y,2],vround)== np.round(mesh_data[lower_ply]['nodeCoord'][
    nodei_lidx,2],vround))[0]]
487 
488                         # getting global indices
489                         element_gidx = mesh_data[upper_ply]['nodeidx'][node_z][0]
490                         upperply_elemnode_list.append(element_gidx)
491 
492                 # 4 nodes of upper layer element for hex elements
493                 upperply_node_gidx = np.reshape(upperply_elemnode_list,(-1,4))
494 
495                 # Creating map to be used for other layers
496                 node_map = np.vstack((upperply_node_gidx[:,0] - lowerply_node_gidx
    [:,0],
497                                       upperply_node_gidx[:,1] - lowerply_node_gidx
    [:,1],
498                                       upperply_node_gidx[:,2] - lowerply_node_gidx
    [:,2],
499                                       upperply_node_gidx[:,3] - lowerply_node_gidx
    [:,3],
500                                       )).transpose()
501             else:
502                 # Using map to get nodes of eleemnts of other layers
503                 upperply_node_gidx = lowerply_node_gidx + node_map
504 
505             new_layer_elemnode = np.hstack((lowerply_node_gidx,upperply_node_gidx))
506             new_layer_elemnode = np.unique(new_layer_elemnode, axis=0)
507             elemnode_map[layer] = new_layer_elemnode
508 
509             if layer == 1:
510                 new_layer_elemidx = np.arange( max(mesh_data[numPly]['elemidx'])+1,
    max(mesh_data[numPly]['elemidx'])+1+len(new_layer_elemnode) )
511                 mesh_data[numPly+1]['elemidx'] = new_layer_elemidx
512                 mesh_data[numPly+1]['elemNode'] = new_layer_elemnode
513             else:
```

```
514                 new_layer_elemidx = np.arange( max(mesh_data[numPly+1]['elemidx'])+1,
     max(mesh_data[numPly+1]['elemidx'])+1+len(new_layer_elemnode) )
515                 mesh_data[numPly+1]['elemidx'] = np.hstack((mesh_data[numPly+1]['
    elemidx'],new_layer_elemidx))
516                 mesh_data[numPly+1]['elemNode'] = np.vstack((mesh_data[numPly+1]['
    elemNode'],new_layer_elemnode))
517
518         # To ensure that there are no duplicate elements sharing same nodes
519         assert(len(mesh_data[numPly+1]['elemNode']) == len(np.unique(mesh_data[numPly
    +1]['elemNode'],axis=0)))
520         print('Mesh generation cohesive layer complete...')
521
522         #=====================================================================
523         # Node Set: Creating node set for boundary conditions
524         #=====================================================================
525         node_list = {}
526         # all nodes with x-coord <= 0.05
527         kx1 = np.where(np.round(mesh_data[numPly]['nodeCoord'][:,0],15) <= clp)[0]
528         # all nodes with z-coord == Lz
529         kz1 = np.where(np.round(mesh_data[numPly]['nodeCoord'][:,2],15) >= round(Lz
    ,16))[0]
530         clamp_set_1_local_idx = np.intersect1d(kx1,kz1)
531         node_list[1] = mesh_data[numPly]['nodeidx'][clamp_set_1_local_idx]
532
533         # all nodes with x-coord <= 0.05
534         kx2 = np.where(np.round(mesh_data[1]['nodeCoord'][:,0],15) <= clp)[0]
535         # all nodes with z-coord == 0
536         kz2 = np.where(np.round(mesh_data[1]['nodeCoord'][:,2],15) <= 0)[0]
537         clamp_set_2_local_idx= np.intersect1d(kx2,kz2)
538         node_list[2] = mesh_data[1]['nodeidx'][clamp_set_2_local_idx]
539
540         # all nodes with x-coord >= Ly-0.05
541         kx3 = np.where(np.round(mesh_data[numPly]['nodeCoord'][:,0],15) >= Lx-clp)[0]
542         # all nodes with z-coord = Lz
543         kz3 = np.where(np.round(mesh_data[numPly]['nodeCoord'][:,2],15) >= round(Lz
    ,15))[0]
544         clamp_set_3_local_idx = np.intersect1d(kx3,kz3)
545         node_list[3] = mesh_data[numPly]['nodeidx'][clamp_set_3_local_idx]
546
547         # all nodes with x-coord >= Ly-0.05
548         kx4 = np.where(np.round(mesh_data[1]['nodeCoord'][:,0],15) >= Lx-clp)[0]
549         # all nodes with z-coord == 0
550         kz4 = np.where(np.round(mesh_data[1]['nodeCoord'][:,2],15) <= 0)[0]
551         clamp_set_4_local_idx = np.intersect1d(kx4,kz4)
552         node_list[4] = mesh_data[1]['nodeidx'][clamp_set_4_local_idx]
553
554         print("Node set complete ..")
555
556         #=====================================================================
557         # LS-Dyna file generation: creating database, in the form of string,
558         # to be added in the LS-Dyna keyword file
559         #=====================================================================
560
561         node_str = '*NODE\n'
562         elem_str = """*ELEMENT_SOLID
```

```
        """
        numEntities = len(mesh_data)
        for ply in range(1,numEntities+1):
            # Node data
            if ply != numPly+1:
                node_data = np.hstack((np.reshape(mesh_data[ply]['nodeidx'],(-1,1)),
                                       mesh_data[ply]['nodeCoord']))
                node_str += '\n'.join(', '.join(str(cell) for cell in row)
                                      for row in node_data)
                node_str += '\n'

            # solid elements
            elem_id = np.reshape(mesh_data[ply]['elemidx'],(-1,1))
            part_id = np.reshape(np.repeat(ply,len(elem_id)),(-1,1))
            node_coord =  mesh_data[ply]['elemNode']
            elem_data = np.hstack((elem_id,part_id,node_coord))
            elem_data.astype('int64')
            elem_str += '\n'.join(', '.join(str(int(cell)) for cell in row)
                                  for row in elem_data)
            elem_str += "\n"

        # Node set
        node_list_str_all = ''
        for node_list_id in node_list:
            node_list_str = f"""*SET_NODE_LIST
        {node_list_id}, 0.0, 0.0, 0.0, 0.0, 0.0, MECH
        """
            for i,node_i in enumerate(node_list[node_list_id]):
                node_list_str += str(node_i)
                if node_i != node_list[4][-1]:
                    if ((i+1)%8 == 0 or node_i==node_list[node_list_id][-1]):
                        node_list_str += "\n"
                    else:
                        node_list_str += ", "

            node_list_str_all += node_list_str

        geo_str = node_str + '$#\n' + elem_str + '$#\n' + node_list_str_all

        # Adding generated string to the LS-Dyna keyword files and getting
        # final string
        main_str = mainFile(geo_str)            # *END inserted here

        print("String generation complete...")

        # Writing LS-Dyna keyword file
        with open(f'{file_dst}{filename}.k','w+') as mygeo:
            mygeo.write(main_str)
        print("Writing file complete...")

        # creating macro for keyword style conversion to new keyword style
        geo_macro_str = f"""*lsprepost macro command file
*macro begin macro_post
openc keyword "{file_dst}{filename}.k"

```

```
$# Deleteing unreferenced nodes
elemedit createnode accept
genselect target node
elemedit delenode delete
elemedit delenode accept 1
Build Rendering data
genselect clear

$# Saving keyword files as per new format
save keywordabsolute 0
save keywordbylongfmt 0
save keywordbyi10fmt 0
save outversion 10
save keyword specialsubsystem "{file_dst}{filename}.k" 1
*macro end
"""
        macro_filename = f"{filename}"
        print("Writing macro to location: ",file_dst)
        with open(f"{file_dst}{macro_filename}.cfile","w+") as my_geo_macro:
            my_geo_macro.write(geo_macro_str)

        # Converting the old keyword style to new keyword style using lsprepost
        print("Saving newly fomatted keyword to location: ",file_dst)
        cmd=f'"{lspp_loc}" -nographics c="{file_dst}{macro_filename}.cfile"'

        subprocess.Popen(cmd, shell=True, stdin=subprocess.PIPE,
                         stdout=subprocess.PIPE)

        time.sleep(20)
    print('elapsed time', process_time() - start)

#%% Example
# loc = "D:\\Academics\\MasterThesisData\\DataAnalysis\\"
# create_geo(Rx              = 0.0024,
#             filename        = 'geo',
#             file_dst        = loc,
#             numCurveElem    = 10,#35,
#             numClampXElem   = 1,#4,
#             numMainbodyXElem= 5,#12,
#             numCSElem       = 10,#20,
#             )
```

A.4.10 LS-Prepost Sample Macro

```
*lsprepost macro command file
*macro begin macro_post

open d3plot "/scratch/ws/0/apku868a-mt/mainProject/0/d3plot"
splane dep0 0.12 0.042 0.00093  1.000  0.000  0.000
splane setbasept 0.12 0.042 0.00093
splane drawcut
$# Extracting: resultant force, x-force,y-force,crossection area
splane xsection 3 4 5 13
xyplot 1 savefile ms_csv "/home/apku868a/main_project/output_csv/0/gForce.csv" 1 all
xyplot 1 donemenu
```

```
deletewin 1
splane done

gtime 2
xyplot 1 savefile ms_csv_multiple "/home/apku868a/main_project/output_csv/0/
    gIntEnergy.csv" 1 all

genselect clear all
genselect part add part 12/0
genselect part add part 10/0
genselect part add part 8/0
genselect part add part 5/0
genselect part add part 3/0
genselect part add part 1/0
maxvalue 1
xyplot 1 select all
xyplot 1 operation max_curve all
xyplot 1 operation save "/home/apku868a/main_project/output_csv/0/0_max_x_stress.csv"
     MSoft_CSV
deletewin 1
genselect clear all

genselect clear all
genselect part add part 12/0
genselect part add part 10/0
genselect part add part 8/0
genselect part add part 5/0
genselect part add part 3/0
genselect part add part 1/0
maxvalue 2
xyplot 1 select all
xyplot 1 operation max_curve all
xyplot 1 operation save "/home/apku868a/main_project/output_csv/0/0_max_y_stress.csv"
     MSoft_CSV
deletewin 1
genselect clear all

genselect clear all
genselect part add part 12/0
genselect part add part 10/0
genselect part add part 8/0
genselect part add part 5/0
genselect part add part 3/0
genselect part add part 1/0
maxvalue 4
xyplot 1 select all
xyplot 1 operation max_curve all
xyplot 1 operation save "/home/apku868a/main_project/output_csv/0/0_max_xy_stress.csv
    " MSoft_CSV
deletewin 1
genselect clear all

genselect clear all
genselect part add part 12/0
genselect part add part 10/0
```

```
genselect part add part 8/0
genselect part add part 5/0
genselect part add part 3/0
genselect part add part 1/0
maxvalue 13
xyplot 1 select all
xyplot 1 operation max_curve all
xyplot 1 operation save "/home/apku868a/main_project/output_csv/0/0_max_shear_stress.csv" MSoft_CSV
deletewin 1
genselect clear all

genselect clear all
genselect part add part 12/0
genselect part add part 10/0
genselect part add part 8/0
genselect part add part 5/0
genselect part add part 3/0
genselect part add part 1/0
maxvalue 14
xyplot 1 select all
xyplot 1 operation max_curve all
xyplot 1 operation save "/home/apku868a/main_project/output_csv/0/0_max_princ_stress.csv" MSoft_CSV
deletewin 1
genselect clear all

genselect clear all
genselect part add part 11/0
genselect part add part 9/0
genselect part add part 7/0
genselect part add part 6/0
genselect part add part 4/0
genselect part add part 2/0
maxvalue 1
xyplot 1 select all
xyplot 1 operation max_curve all
xyplot 1 operation save "/home/apku868a/main_project/output_csv/0/90_max_x_stress.csv" MSoft_CSV
deletewin 1
genselect clear all

genselect clear all
genselect part add part 11/0
genselect part add part 9/0
genselect part add part 7/0
genselect part add part 6/0
genselect part add part 4/0
genselect part add part 2/0
maxvalue 2
xyplot 1 select all
xyplot 1 operation max_curve all
xyplot 1 operation save "/home/apku868a/main_project/output_csv/0/90_max_y_stress.csv" MSoft_CSV
deletewin 1
```

```
genselect clear all

genselect clear all
genselect part add part 11/0
genselect part add part 9/0
genselect part add part 7/0
genselect part add part 6/0
genselect part add part 4/0
genselect part add part 2/0
maxvalue 4
xyplot 1 select all
xyplot 1 operation max_curve all
xyplot 1 operation save "/home/apku868a/main_project/output_csv/0/90_max_xy_stress.csv" MSoft_CSV
deletewin 1
genselect clear all

genselect clear all
genselect part add part 11/0
genselect part add part 9/0
genselect part add part 7/0
genselect part add part 6/0
genselect part add part 4/0
genselect part add part 2/0
maxvalue 13
xyplot 1 select all
xyplot 1 operation max_curve all
xyplot 1 operation save "/home/apku868a/main_project/output_csv/0/90_max_shear_stress.csv" MSoft_CSV
deletewin 1
genselect clear all

genselect clear all
genselect part add part 11/0
genselect part add part 9/0
genselect part add part 7/0
genselect part add part 6/0
genselect part add part 4/0
genselect part add part 2/0
maxvalue 14
xyplot 1 select all
xyplot 1 operation max_curve all
xyplot 1 operation save "/home/apku868a/main_project/output_csv/0/90_max_princ_stress.csv" MSoft_CSV
deletewin 1
genselect clear all

genselect clear all
genselect part add part 13/0
maxvalue 1
xyplot 1 select all
xyplot 1 operation max_curve all
xyplot 1 operation save "/home/apku868a/main_project/output_csv/0/c_max_x_stress.csv" MSoft_CSV
deletewin 1
```

```
genselect clear all

genselect clear all
genselect part add part 13/0
maxvalue 2
xyplot 1 select all
xyplot 1 operation max_curve all
xyplot 1 operation save "/home/apku868a/main_project/output_csv/0/c_max_y_stress.csv" MSoft_CSV
deletewin 1
genselect clear all

genselect clear all
genselect part add part 13/0
maxvalue 13
xyplot 1 select all
xyplot 1 operation max_curve all
xyplot 1 operation save "/home/apku868a/main_project/output_csv/0/c_max_shear_stress.csv" MSoft_CSV
deletewin 1
genselect clear all

genselect clear all
genselect part add part 13/0
maxvalue 14
xyplot 1 select all
xyplot 1 operation max_curve all
xyplot 1 operation save "/home/apku868a/main_project/output_csv/0/c_max_princ_stress.csv" MSoft_CSV
deletewin 1
genselect clear all

genselect node add node 56138/0
ntime 5
addplot
ntime 6
addplot
ntime 7
addplot
ntime 8
xyplot 1 select all
xyplot 1 savefile ms_csv "/home/apku868a/main_project/output_csv/0/node_middle.csv" 1 all
deletewin 1
clearpick


postmodel off
+M 1

splane setbasept 0.12 0.0144 7.91666e-05
splane dep0 0.12 0.0144 7.91666e-05  1.000  0.000  0.000
splane drawcut
splane xsection 4 13
xyplot 1 operation divide_curves 1/2
```

```
xyplot 1 savefile ms_csv "/home/apku868a/main_project/output_csv/0/nStress.csv" 1 all
xyplot 1 donemenu
deletewin 1
splane done

genselect target element solid
switch2selection
clearpick
genselect target element solid
genselect element add plane in 0.119865 0.0105 7.91667e-05 0 0 1 0.00002 0
genselect element remove plane in 0.12 0.011 0 0 -1 0 1 1
genselect element remove plane in 0.12 0.0175 0 0 1 0 1 1
genselect element remove plane in 0.119125 0.0144 0 -1 0 0 1 1
genselect element remove plane in 0.120875 0.0144 0 1 0 0 1 1

etime 14
xyplot 1 operation max_curve all
xyplot 1 savefile ms_csv "/home/apku868a/main_project/output_csv/0/maxPrinc.csv" 1
    all
deletewin 1

etime 81
xyplot 1 operation max_curve all
xyplot 1 savefile ms_csv "/home/apku868a/main_project/output_csv/0/fa.csv" 1 all
deletewin 1

etime 83
xyplot 1 operation max_curve all
xyplot 1 savefile ms_csv "/home/apku868a/main_project/output_csv/0/fmat.csv" 1 all
deletewin 1

etime 85
xyplot 1 operation max_curve all
xyplot 1 savefile ms_csv "/home/apku868a/main_project/output_csv/0/da.csv" 1 all
deletewin 1

etime 87
xyplot 1 operation max_curve all
xyplot 1 savefile ms_csv "/home/apku868a/main_project/output_csv/0/dmat.csv" 1 all
deletewin 1

etime 87
xyplot 1 operation average_curves all
xyplot 1 savefile ms_csv "/home/apku868a/main_project/output_csv/0/dmat_avg.csv" 1
    all
deletewin 1

clearpick
postmodel on

*macro end
```

Statement of Originality

This thesis has been performed independently with support by the candidate's supervisors. It contains no material that has been accepted for the award of a degree in this or any other university. To the best of the candidate's knowledge and belief, this thesis contains no material previously published or written by another person except where due reference is made.

Apurv Kulkarni

Dresden, 04 May 2021