A Fast and Efficient Discrete Model for Composites (FastDM4C)

Antonio Alessandro Deleo

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2023

Reading Committee:

Marco Salviato, Chair

Jinkyu Yang

Richard Wiebe

Program Authorized to Offer Degree:
Aeronautics & Astronautics

University of Washington

Abstract

A Fast and Efficient
Discrete Model for Composites
(FastDM4C)

Antonio Alessandro Deleo

Chair of the Supervisory Committee:
Professor Marco Salviato
Aeronautics & Astronautics

The adoption of composite materials in the aerospace industry has enabled the achievement of structural performance levels and weight savings unimaginable even just twenty years ago. Yet, only a fraction of the true potential of these materials has been expressed to date due to lack of high-fidelity models which has resulted in the adoption of extremely conservative designs compared to metallic counterparts. In part this is because, in contrast to high performance metallic alloys, the fracturing behaviors of composites are way more complex and difficult to simulate. Fiber reinforced composites feature many interacting mechanisms spanning several material and structural length scales, from the fiber scale of few microns to the structural scale of a composite wing with a span of several meters. This makes the development of computational models for the design and optimization of composite structures extremely challenging owed to the conflicting need of being able to capture microdamage events at the fiber scale while still be efficient enough to simulate structures that are at least six orders of magnitude larger.

This work attempts to address this challenging problem by formulating a novel discrete, sub-lamina-scale model aimed at providing an effective description of damage at the microscale while maintaining computational costs comparable to continuum, homogenized formulations. The proposed model is also based on the Discrete Model for Composites

(DM4C)[1] where the constitutive relationship of the discrete members representing the matrix are edge-based instead of 3D-based. Both DM4C and Fast Discrete Model for Composites (FastDM4C) have been proven successful and are providing tools to both academic and industry partners to pave the way for the full exploitation of the advantages of composites in aerospace structure design.

In this new approach, composites are simulated as an assembly of Representative Unit Cells (RUC) of roughly the same dimensions of the average distance between splitting cracks. In contrast to traditional models, which homogenize the mechanical behavior of the fibers and matrix into an equivalent continuum, the new model simulates explicitly groups of fibers and surrounding matrix material leveraging a proper configuration of one-dimensional Finite Elements. The arrangement of the fiber- and matrix-elements within the RUC is designed to replicate the transversely-isotropic behavior of the lamina. One distinct regularized strain-softening constitutive law is utilized to describe the behavior of the fibers using a new element called Discrete Fiber Model (DFM). The matrix is instead modeled using three different implementations of the same edge-based constitutive formulation called Discrete Matrix Model (DM2) which is meant to capture pure matrix behavior, in-plane shear behavior, and interface behavior between different plies.

A multiple stage optimization algorithm is developed to calibrate both elastic and fracture parameters of the model, both at the small scale (for elastic behavior) and at the large structure scale (for fracturing behavior) levying the use of a Machine Learning/Artificial Intelligence algorithm coupled with a relational database to store and process large number of simulations. Then, several simulations of the composite structures under highly non-linear behavior are used to validate the model and showcase its capability of capturing the inherent damage and fracture mechanisms of composite laminates.

It will be shown that the proposed FastDM4C is capable of capturing the inherently complex damaging behavior of composites by comparing it to experimental results, while at

[1]Discrete Model for Composites is currently under-development by the author and his research group composed by Prof. Marco Salviato, Sean Phenisee from University of Washington Seattle, and Dr. Daniele Pelessone from ES3, San Diego.

the same time showing its numerical efficiency capable of running real engineering structures.

This dissertation is split into six main chapters:

1. **Introduction and Research Objective**: clear statement of goals to be achieved by the completion of the project. List all the included topics in the deliverable package that will be available to any reader.

2. **Review of Computational Fracture Mechanics Models**: overview of already existing and established computational models and their thorough explanation, highlighting pros and cons and where the proposed model would fit in the overall taxonomy of computational fracture mechanics.

3. **Theoretical Framework**: cover the fundamental theories used in the proposed model, starting from governing equations, constitutive relationships, finite element implementations, damage mechanics and optimization algorithms.

4. **Computational Implementation**: detailed explanation of the steps of the computational model.

5. **Results**: showcases of the completed tests that showed the feasibility of the model, comparing the computational results to real fracture experiments.

6. **Conclusions**: overall summary of the whole model with its advantages, disadvantages, and modes of use. The proposed model is not trying to solve all the problems, but rather tackle some of them in a unique way. It stands with the user the understanding of why using this model can be beneficial to computational studies.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Number Page

LIST OF TABLES

Table Number Page

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to my advisor, Professor Marco Salviato, for his unwavering guidance, invaluable support, and exceptional mentorship throughout the journey of completing these doctorate studies since my undergrad. His profound knowledge, dedication, and passion for the subject matter have been instrumental in shaping, first me as a better person, and secondly, me as a better researcher. He and his exquisite wife Rossella has always supported me both in good and bad times and without them I would have never been able to start and finish this tortuous, nevertheless wonderful, path. I will always be grateful for them and I hope I will be able to continue perform top-notch research with everything Professor Salviato taught me, making him as proud as possible.

I would also like to extend my sincere appreciation to all the members of my graduate committee members. Professor Jinkyu Yang for his constant availability and willingness to guide me through the challenges encountered along the way, both in my undergrad studies and in graduate research projects we were able to work together. Professor Eli Livne for his commitment to fostering a nurturing and intellectually stimulating environment for all the projects I have been working on. His classes on aerolasticity, structural optimization, and aircraft capstone were crucial and paramount to my academic studies and I will be forever grateful to have had such high quality teachings. Professor Richard Wiebe from the Civil Engineering Department who I first met taking his structural stability class. His passion for teaching was so vibrant it was so refreshing to take a class after years of completing my academic required classes and he was the reason why I went back taking at least a class per quarter for fun until the end of my studies. The Civil Engineering Department is very lucky to have him. I am also deeply grateful for the countless hours Dr. Daniele Pelessone spent with me in discussing and refining research objectives, methodology and findings. What is more unbelievable is that he found all this time while working full time in the company

his wife and he founded. His expertise and meticulous attention to detail have played a pivotal role in ensuring the rigor and coherence in both the theoretical and computational implementation of the model I have been working on. Last but not least Professor Dana Dabiri for his patience in teaching me Fluid Mechanics, which has been by far my Achilles's heel.

Furthermore, I would like to acknowledge Professor Peter Mackenzie-Helnwein for his object oriented advanced python class, Professor Kuen-Yuan Lin for his incredible teachings of mechanics of composite materials and for been able to teach three generations of my family as well, and Professor Duane Storti for introducing me to the CUDA world which I intend to spend a lot of my career on for the computational advancements of fracture mechanics.

Moving to family, there are no more important people to thanks than my parents, my dad Roberto and my mom Angelica. For obvious reasons I would not be here if it were not for them and I will always be forever grateful because they never let me miss anything while growing up. Their endless support is the reason why I became who I am right now. My brothers Riccardo and Federico for supporting me during the hardest times and more importantly for enduring my constant peskiness. My grandma Emilia for loving me to the end and beyond. My grandpa Antonio and uncle Francesco to whom I owe most of my academic career and to whom I have always looked while growing up to become an aerospace engineer. My parents-in-law for also supporting me from the other part of the world and to all the rest of my family scattered as well all around the globe.

I cannot miss to acknowledge also the infinite support from my friends and peers in GUG307. Gustavo Eidji Camarinha Fujiwara for all the sleepless nights spent in the office working coupled with the most technical discussion about research; Pablo Trefftz-Posada for sharing pretty much all the hobbies I ever had; Abhiram Balachandra Aithal for his wisdom in both academic and every-day life; Sean Eunsik Phenisee for always being there for me and support me for 8 years straight; Dr. Seunghyun (Seunge) Ko for his patience and academic advises; Dr. Yao Qiao (together with his wife Wen Wen and son Evan) for his tremendous help, support, and coaching; Dr. Federico Fonte for his international support

from all around the world, all the outdoor trips he organized, and his immense hospitality whenever I get to visit him; and Troy Nakagawa, Dr. Minh Hoang Nguyen, Dawei Lu, Ryan Howe, Shiva Goutham Pattapu.

I would like to thanks also to all my friends from Italy, particularly from the BevanoRacingTeam: Pietro Berloni, Alessandro Nava, Alessandro Marchetti, Riccardo Paci, Riccardo Painelli, Alessandro Mariotti, Nicola Furiassi, Luca Battistoni, Nicola Manna, Nicola Bartocetti, Niccolo' Bartoletti; and the best for last: Anna Riccomi and the new arrived Olimpia, Cecilia Papalini, Domitilla Ravot-Licheri, Camilla Serafini and Serena Monticchio. A big thanks also to all my friends from the Conservatory of Music, particularly to my Professor Fiorenzo Di Tommaso; and all my high school friends from the class 5°C; and from the Tiro a Segno Nazionale TSN of Pesaro, particularly to Michele Candalia and Alberto Cardinali.

A huge thanks goes also to all my friends in Seattle which kept me sane during these hard working years. To my undergraduate friends: Maxine Tan, Tarik Ruydemir, Nick Dominski, Molly Zhang, Reed Hawkins, Jamie Lambie, Leila Asfari and Tim Brummer. To my climbing friends: Adriana Medrado Austgen, Wyatt Lee Moore, Vidur Vij, Alice Liu, Sierra Schatz, Cerwyn Chiew, Tashi Sherpa, Anais Capik, Tomek Fraczek, Diala Sleek, Ian Culhane, Jackie Makdah, Ari Athair, Nicole Zhao, and all the rest of the climbing club @ UW, past and present staff and officers.

I would also like to honor the memory of all the people who, although no longer with us, have made a profound impact on my academic career and personal growth. They all remain an eternal source of inspiration.

DEDICATION

to my dear wife, Crystal

Chapter 1

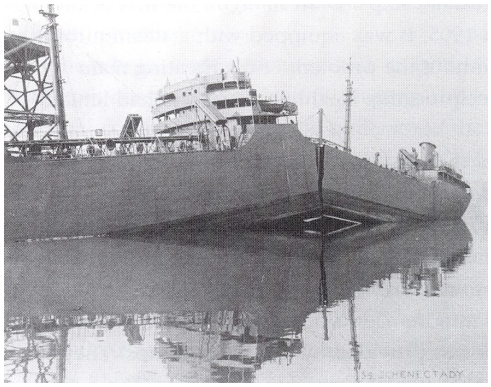
INTRODUCTION AND RESEARCH STATEMENT

1.1 Motivation for Computational Fracture Mechanics in Aerostructural Design

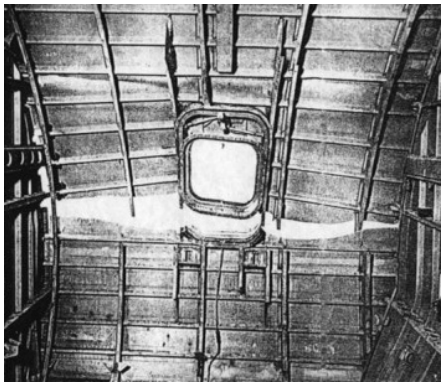
Fracture mechanics is a branch field of mechanics that is concerned with the study of the propagation of cracks and defects in materials and structures. It is a fairly recent discipline that was initially formally introduced to the academic world by Inglis in 1913 [1] and Griffith right after in 1921 and 1924 [2, 3]. The initial interest in this discipline was fed by historical accidents, as cracks and defects were considered insignificant inconveniences at the time and designers and engineers did not think they posed major threats. It was during the Second World War that scientists started realizing many ships and air-crafts started to fail unexpectedly and in inexplicable ways. Later on, it was indeed determined that these failures occurred due to cracks in the metal structures. The most classical example with great educational purpose is the brittle fracture in the *Liberty Ships* [4], where many ships operating in cold environment experienced an embrittlement of the hull causing the ships to crack almost in half as shown below in Figure 1.1a.

Another example worth citing is the fatigue failure of the *de Havilland Comet* [5], Figure 1.1b, where oversight in the design process led to cracks developing and propagating from stress concentration areas located in the squared fuselage windows. During pressurization tests, a higher value of yielding stress for the aluminum fuselage was achieved to prove the worthy design, but at the time the engineers did know very little about fatigue crack growths and did not know cracks would eventually develop in those higher stress concentration regions which would result in catastrophic failure. By the time the redesigned Comet entered service, the aviation marked moved away and chose the newly designed Boeing 707, marking a new era for the American commercial aviation market.

The last example of interest is the Aloha Airline Flight 243 [6], Figure 1.1c, where simi-



(a) Liberty Ship cracking in half.



(b) Catastrophic failure of the *de Havilland* Comet.

(c) Extended Fuselage Damage of Aloha 243.

Figure 1.1: Famous Historical Accidents

larly to the previous accident, an oversight in the fatigue and damage tolerance certification led to an extensive mid-flight fuselage damage where a third of the top fuselage detached from the aircraft itself. Fracture mechanics was already known then; however, due to the much shorter than average flight cycles (islands in Hawaii are very close to each other, a take-off from one island and a landing to another would still result in a flight cycle even if shorter than an hour flight), multiple fatigue cracks developed and propagated from skin panel to skin panel through lap joints. Luckily, the accident caused only one fatality, a crew member who was standing in aisle, while all the passengers remained unscathed.

These are just three of (unfortunately) many and many examples where the engineers learned the hard way how to properly and safely design aircraft, and any other type of machinery. This emphasises the need to have at disposal the most advanced and state of the art tools to do a safer job as a single loss of life would be one too many no matter the circumstances.

In the past 50 years, with the advent of modern computational technologies and better computing machines, scientists and engineers were able to develop better and better models, revolutionizing the aerospace sector as more and more tools started to become available to produce, test, and validate complex aerostructural designs, on par with the development of the Finite Element Method (FEM) by Turner *et al.* in 1956 [7] which also revolutionized almost all engineering sectors. In less that a decade after the introduction of the FEM method, the first fracture computational models started to emerged, capable of capturing some of the complex behavior of the mechanics that lead to tragic accidents as the ones described in Figure 1.1. This only highlights and re-iterates again the importance and the need of more advanced, reliable, and accurate computational models to better describe the complex modern designs.

1.2 Challenges in the Formulation of Computational Fracture Mechanics Approaches for Composite Structures

The tremendous progress in computational modeling and advanced elasto-plastic fracture mechanics have enabled unprecedented accuracy and efficiency in the simulation of metallic aerostructures. Establishing similar computational tools is probably even more important

for composites than for metals due to the broader parameter space available for optimization on one side and the complex multi-scale, multi-mechanism failure behavior on the other. However, addressing this important problem is far from easy and a computational model that can take on this challenge is still elusive.

What makes the simulation of composites so challenging is their complex damaging behavior stemming from the presence of several constituents of vastly different characteristic length scales and mechanical behaviors. Single carbon fibers have a diameter between 5 to 10 micrometers [8]; therefore, in a single laminate, tens or even hundred of thousands of single fibers are present. These fibers are impregnated by a resin, a material of completely opposite mechanical behavior, and then each fiber or bundle of fibers develops a complex interface where both elastic and fracture mechanisms enact. It would be impossible to characterize the behavior of each single fiber given its diameter variability and distribution even with state of the art computational power; therefore, engineers and scientists rely on different scales modeling. Figure 1.2 shows some of the typical damage behaviors that composite structure experience: splitting cracks, fiber breakage, delamination, matrix cracking, and fiber kinking, from an X-ray scanning of a composite laminate. Also, Figure 1.3 shows the random distribution of fibers in a single composite ply and the complex matrix fracturing behavior between each single fiber. It is clear then it would be critical and absolutely necessary being able to characterize such damages if very accurate and reliable results are expected. On the other hand, a computational model needs to be computationally feasible to run in order to be used in real engineering applications and design workflows.

1.3 Research Statement

Following the points highlighted in Section 1.1 and Section 1.2, a novel discrete computational model is presented capable of capturing complex fracturing behavior but still being able to run efficiently with run times comparable almost to continuum approaches. The scale strategy lies exactly in between microscale modeling (ply scale) and mesoscale modeling (laminate modeling), where a unit representative cell of size comparable to the size of the distance between splitting cracks is modeled similarly to a transversely isotropic composite substructure where fibers dominate one direction, matrix dominates the perpen-

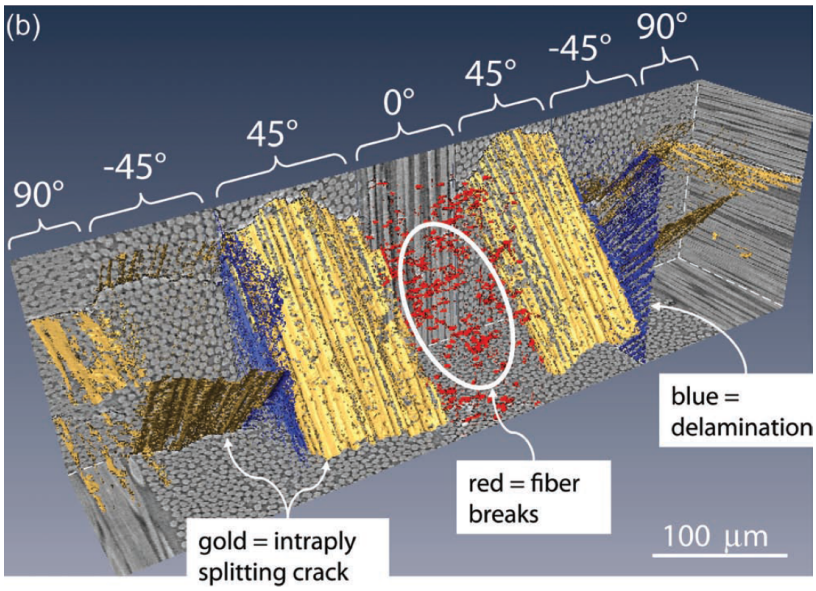


Figure 1.2: Schematics of the different failure micromechanisms in Composites using an X-ray computed tomography of a quasi-isotropic laminate loaded in tension parallel to the plies with fibers oriented at 0°. Adapted from [9].

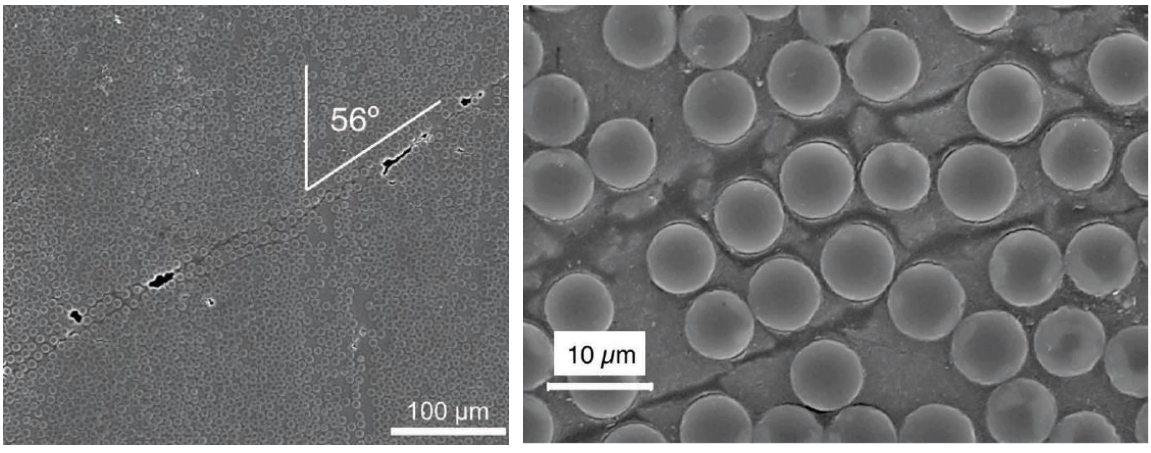


Figure 1.3: Another example of complex distribution of fibers in a single lamina and a complex matrix cracking between each single fiber. Adapted from [9].

dicular direction and diagonal elements are added to mimic the interaction between the two completely opposite behaviors. With the use of advanced optimization techniques split into two main stages using gradient descent for small structures and global surrogate modeling for larger ones, the unit cell is calibrated to represent an elastic transversely isotropic material and the whole structure is used to calibrate fracture parameters. The result is a complete discrete model described by 1D elements which is extremely computationally efficient and still capable of describing complex damage behaviors. Another advantage of this approach is the possibility to model the damage within the different members of the unit cells in different ways, yielding to a more flexible way to easily match experimental tests. One example is the constitutive behavior of the diagonal members of the unit cell. While yet not implemented at the presentation of this written portion of the general exam, some of the initial preliminary issues with splitting crack location and propagation is due to the fact that once the unit cell is undergoing a condition similar to simple shear, one diagonal element is in tension (therefore experience damage), while the other is in compression (therefore remain elastic as damage is not modeled in compression). The solution is to model the diagonal members as elastic-perfectly-plastic in compression while retain all the damaging constitutive behavior in tension. But the fiber and matrix components do not experience such behavior; therefore, not needing special behavior in compression.

To recapitulate, the goal of the author is to develop a novel tool for structural engineers to have a better understanding of the damage characteristics and behaviors of complex composite structures both as a preliminary design tool and as a general analysis tool by providing a model that tries to fill the gaps between the modern microscale and mesoscale modeling of composites. Figure 1.4 provides a broad overview of modern scale modeling techniques with emphases of where the FastDM4C will fit in.

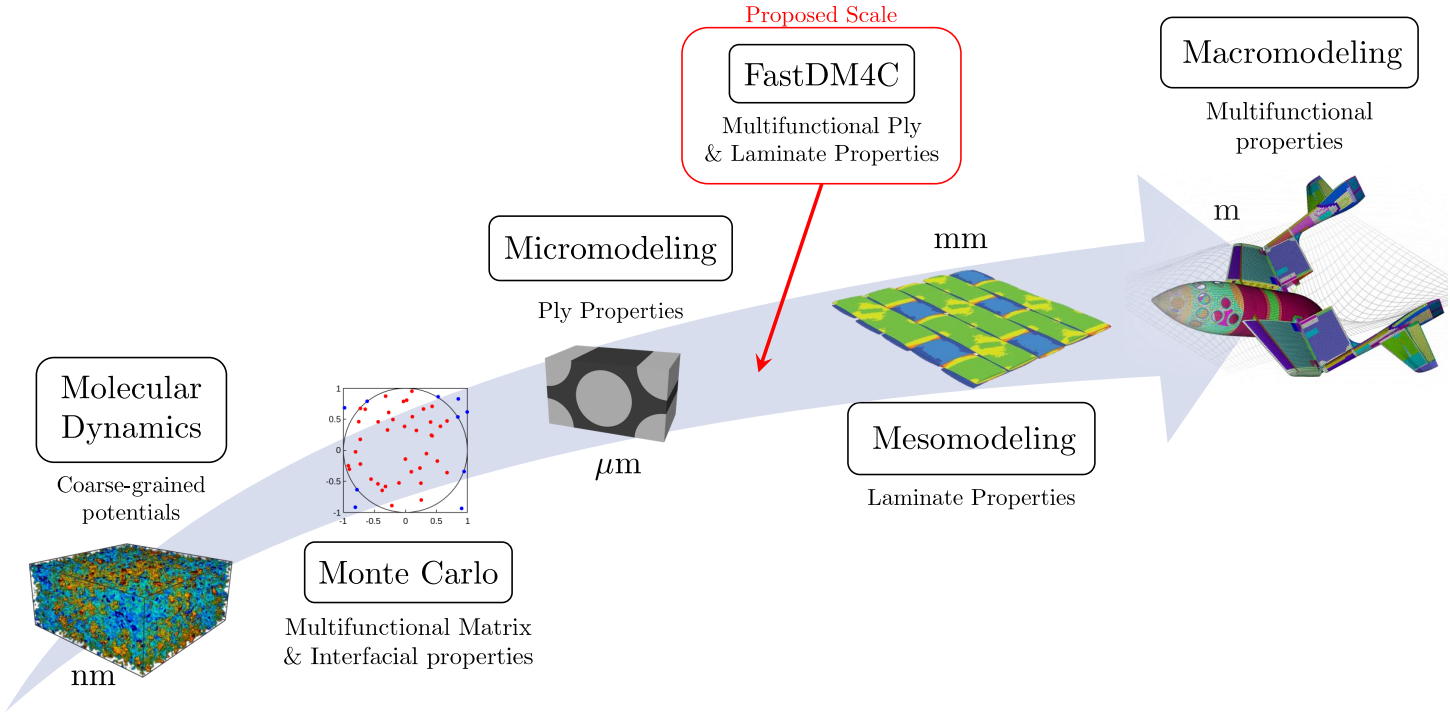


Figure 1.4: FastDM4C is a model proposed to fill the gaps between the microscale and the mesoscale modeling of composites.

Chapter 2

COMPUTATIONAL MODELS OF COMPOSITES: STATE OF THE ART

2.1 Importance of Better Damage Modeling in Composites

Although modern composite materials have been used since the early 1900s, they are still an important and currently active area of academic and commercial research. This is due to the inherent benefits of much higher weight to strength ratio over classical metal alloy structures and increase tailorability. Moreover, characteristics such as fatigue life, thermal expansion, and corrosion resistance are also improved. [10, 11, 12, 13, 14, 15, 16, 17, 18]. However, the damage and failure mechanisms are still posing the biggest challenge in the design workflow of any composite structure, as they can be fairly complicated and up to now there is no a comprehensive model that is able to capture everything that is going on in a damaged laminate given the demanding computational cost and complexity of such modeling [19].

Academic community has been developing three main types of computational model for modeling damage in composites as shown in Figure 2.1 fully on page 9. Both most classical and most recent state of the art models will be presented to give an overview of the computational status in the academic field up until now.

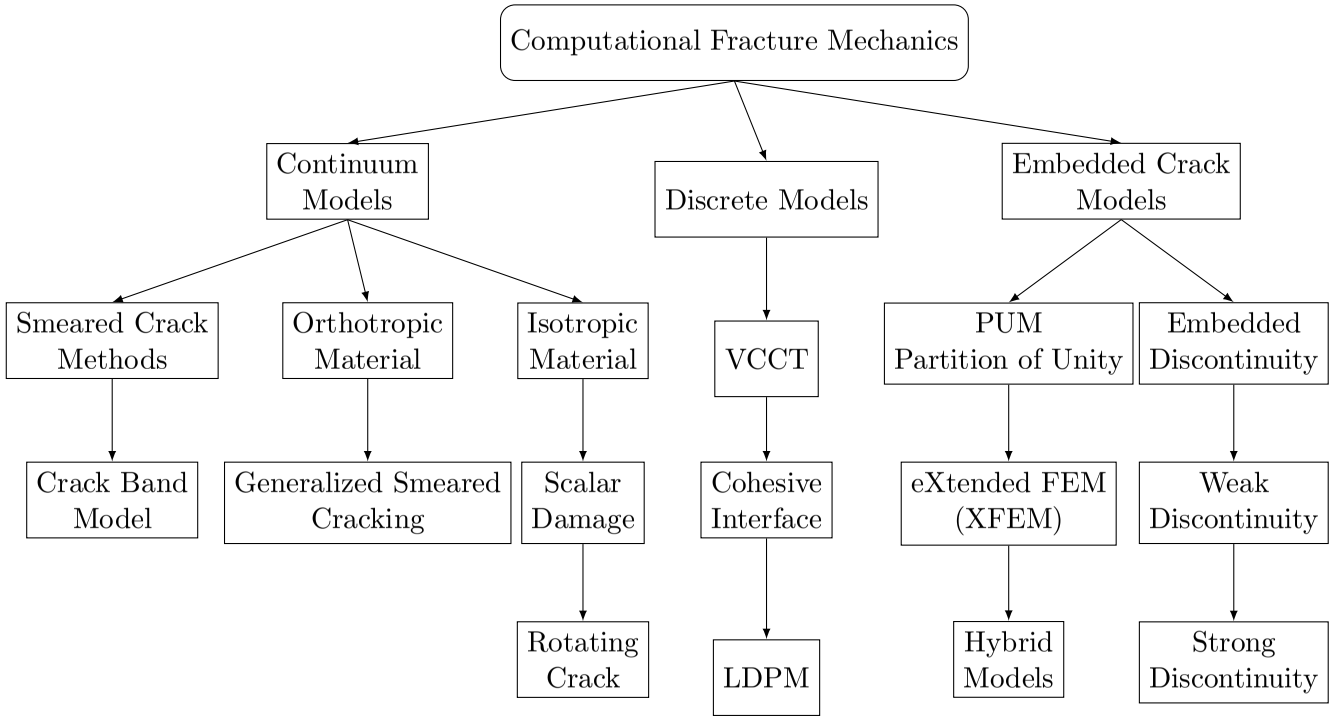


Figure 2.1: Taxonomy of Major Computational Fracture Mechanics Models

2.2 Continuum Models

In continuum models, cracks and defects are modeled in a *smeared* approach. Firstly introduced by Rashid in 1968 [20], the nucleation of one or multiple cracks translates into a deterioration of the material properties at the element level (at the Gaussian integration points for example in the Finite Element (FE) Framework). Therefore, the damage is spread on the overall element and usually it is not possible to easily distinguish different modes of failure and elements need to be enhanced in order to capture the damage kinematics constraints. On the other hand, remeshing and restrictions on the orientation of crack planes are not needed and generally the computational cost is low enough for non extremely complex models that the use of supercomputers is generally not warranted.

2.2.1 Rotating Smeared Crack Model

The first model introduced by Rashid [20] used a simple implementation: when the combination of stresses satisfies a specified criterion (for example major principal stress or strain), the crack is initiated and the orthotropic stress-strain relation is written in accordance to normal and tangential direction of the crack. If the direction of the crack changes, the new change in the direction is implemented using a rotational/transformation matrix. However, due to the inherent ill-condition of the orthotropic stress-strain relation at crack initiation, convergence issues usually arise. Smartly, Rashid introduced a reduced shear modulus and a gradual decrease of the tensile carrying capacity, conceiving the very first *Tension-Softening* model, used anywise in almost every future continuum model. The introduction of the softening part was explained by the fact that such drop to zero would put at disadvantage the part of the material that has not been damaged yet. While this approach represented one of the first attempts in modeling fracture-induced strain localization, it suffered of significant mesh-dependence. In fact, simulating the same material with different meshes would generally provide very differently fracturing behaviors. The reason is that this model did not feature any strain localization limiter. A solution to this important problem was proposed with the Crack Band model which is presented next.

2.2.2 Crack Band Model

To overcome localization instabilities and spurious mesh sensitivity issues, Bažant introduced his *Crack Band Model* in 1983 [21] (initially in 1982, but he published shortly after multiple articles correcting and refining certain arguments). Both issues could be competently dealt with the association of a certain width of the crack band to the crack constitutive relation, which represents a major reference width that can be treated as a material property. As initially formulated by Bažant, the width of the crack band would require the mesh to have exactly the same size of the crack width. The brilliant solution, however, was to uniformly distribute the fracturing strain over the strain and re-scaling the energy of the softening part in order to maintain the total fracture energy constant. This solution yielded extremely accurate results to where the crack path was aligned with the crack band; however, when that was not the case, the model suffered of slightly higher crack propagation resistance. One way to solve the problem was to implement a Voronoi-based mesh to introduced more uniform randomness. Ultimately, the Crack Band Model is a very powerful tool if used properly as it circumvents many issues that were not addressed until it was proposed, and even though it was proposed almost 40 years ago, it is still widely used by modern scientific community even for composite laminates as shown in Figure 2.2. For more information, the reader is referred to [22].

2.3 Discrete Models

Differently from the Continuum models described in Section 2.2, discrete models' approach is to simulate the initiation and propagation of dominant cracks by treating them as a purely geometric discontinuity and as such, cracks are forced to propagate along element boundaries. Early works were done by Scordalis in 1967 and Rashid in 1968 [20]. Discrete models are usually more intuitive approaches, easy to implement, and fairly accurate; however, they suffer of mesh dependencies, continuous change in topology (remeshing) and for large complex structures, they become extremely expensive.

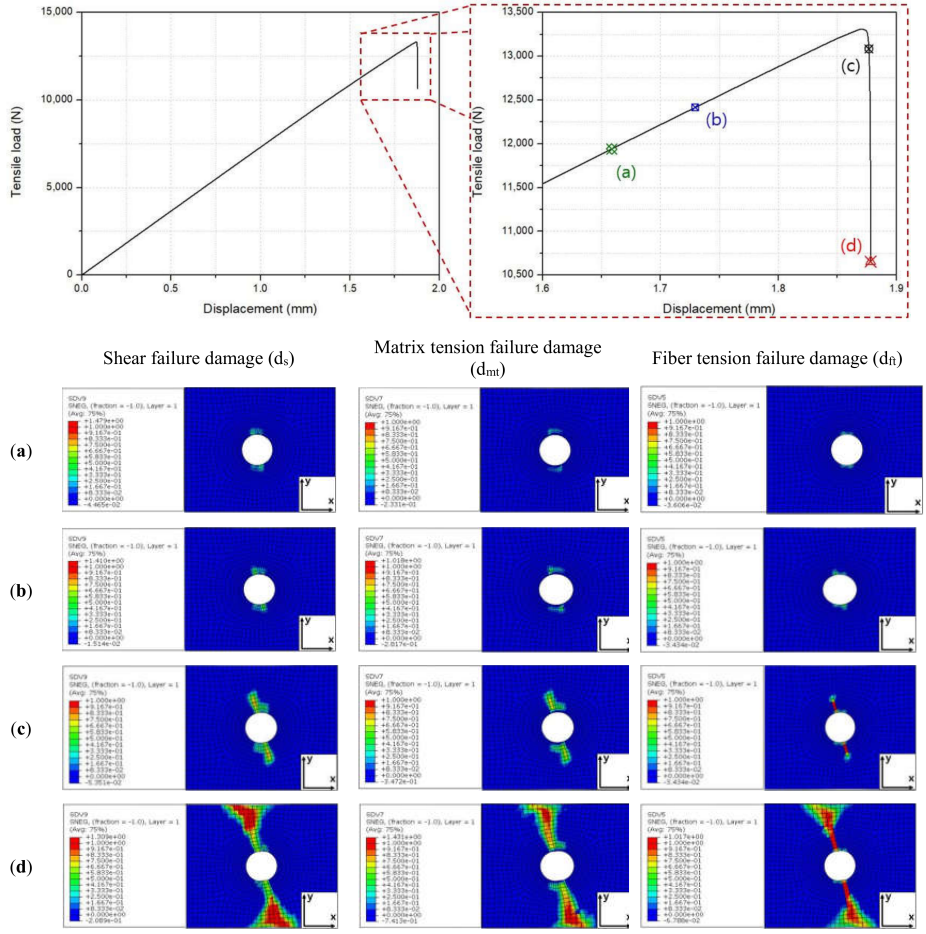


Figure 2.2: Numerical results and validation of the Crack Band Model applied to notched composite laminates. Extracted from [23].

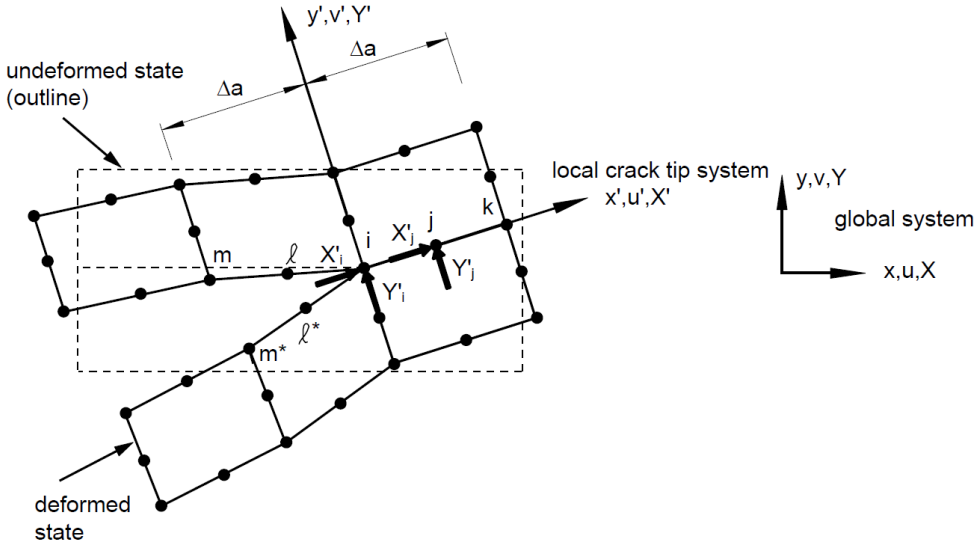


Figure 2.3: Graphical description of the Virtual Crack Closure Technique (VCCT) method. Extracted from [27].

2.3.1 Virtual Crack Closure Technique (VCCT)

One of the first discrete models ever published with enormous success was the Virtual Crack Closure Technique (VCCT), presented by Rybicki and Kanninen in 1977 [24]. A quick graphical description is provided in Figure 2.3. It became widely used to compute the energy release rate because the fracture mode separation was determined explicitly. Based on Irwin's Crack Closure Integral [25], the model is based on the assumption that when the crack propagates, the energy release rate is equal to the energy required to open the crack. While the model worked fairly well with linear elements, initially node-wise opening or closing operations were kinematically incompatible with quadratic elements. However, Raju in 1987 [26] provided an element-wise opening operation which resulted compatible and greatly improved the accuracy of the VCCT model. One critical drawback was the instability associated to a crack opening between two different material interfaces as stress oscillations rendered results unreliable. The solution to this problem gave birth to a completely new discrete approach: Cohesive-Surface Models.

2.3.2 Cohesive-Surface Models

One main disadvantage of Linear Elastic Fracture Mechanics (LEFM) is that the assumption of the Fracture Process Zone (FPZ)'s size being comparable to the structural dimension is not satisfied all the time [22]. Cohesive-Surface models, firstly introduced by Barenblatt in 1959 [28], differently for example from the Virtual Crack Closure Technique, allow to analyze the damaging process even before a dominant crack or defect is created. Although this perk was not included in the original papers, it was quickly discovered right after and amply used. Moreover, the cohesive constitutive relationships encompass all the failure characteristics of the materials as well as its separation process. Fracture, if it happens, is a natural outcome of the deformation and damaging process, and no more initial fracturing conditions must be added to such modeling. However, from dimensional analysis consideration of the fracture energy, a new characteristic length is introduced (similarly to the Crack Band Model in Section 2.2.2) and all the disadvantages of mesh dependency, crack path that must be known a priori, and dynamic instabilities resurface back, as noted by Schellekens and de Borst in 1993 [29, 30]. Although the cohesive surface model is essentially a discrete approach, it could be transformed into a continuum approach by re-distributing and smearing the fracture energy over the width of the elements.

2.3.3 Lattice Discrete Particle Model (LDPM)

The Lattice Discrete Particle Model (LDPM) is a fairly recent discrete model developed by Cusatis *et al.* in 2011 [31] which is extremely suitable for simulation of the failure behavior of concrete. LDPM simulates the concrete structure at the mesoscale level, which is considered to be the length scale of coarse aggregate pieces. Formally, it is the union of the Confinement Shear Lattice (CSL) model [32] and the Discrete Particle Model (DPM). The particle generation is carried out by assuming that each aggregate piece can be approximated as a sphere using a distribution function. Subsequently, the aggregate volume fraction is calculated, where all components such as air, water, and cement are taken into consideration. The lattice system is then defined by using Delaunay Tetrehedralization [33] and the potential material failure is characterized between edges, nodes, and facets. The

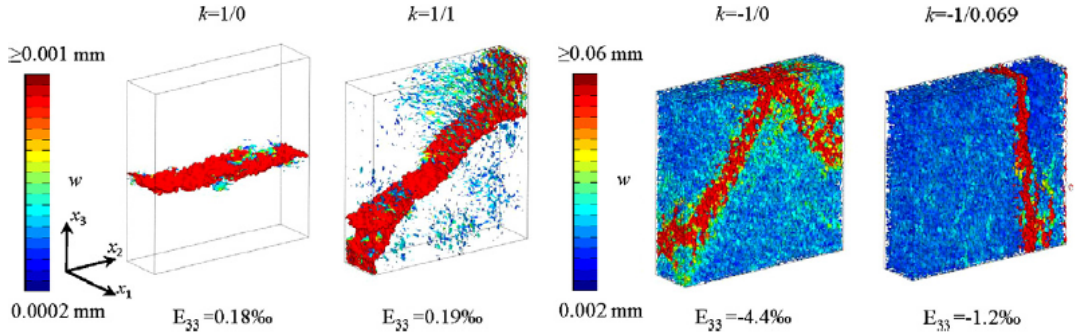


Figure 2.4: Example of biaxial calibration using the LDPM model for a concrete structure. Extracted from [34].

lengthy calibration [34] is carried out to fully characterized the fracture behavior of composites and the computational results agree fairly well with the experimental results. An example is shown in Figure 2.4 where a concrete material was calibrated for biaxial behavior and the complex fracturing behavior captured fairly well.

2.4 Crack-Embedded Models

Embedded Crack Models have been considered as a way to bridge the gaps between discrete and continuum models by integrating (embedding) the crack discontinuity into the element formulation and model the constitutive relationships within the framework of damage theory. The pioneer in this recent field was Nayroles in 1992 [35] with his diffuse element method, which became the precursor of the two most famous Crack-Embedded methods, namely GFEM, developed in 1995 in Texas University [36], and XFEM, developed in 1999 in Northwestern University [37].

2.4.1 eXtended Finite Element Method (XFEM)

Differently from the Element-Free Galerkin method, the eXtended Finite Element Method (XFEM) method relies on the introduction of additional enriched basis functions which satisfy partition of unity and can tackle discontinuous displacement fields along the crack surface. For cracks in elastic materials, the crack tip enrichment functions are based on the

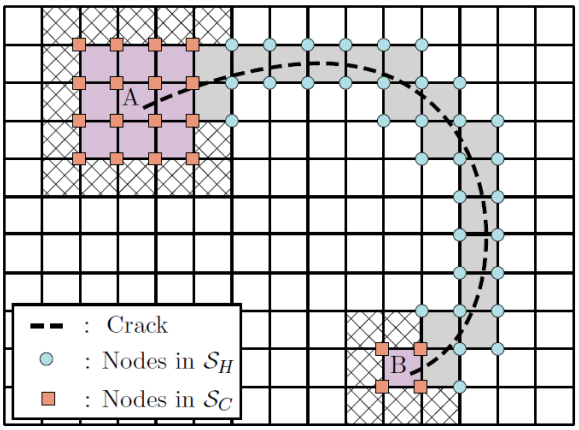


Figure 2.5: Graphical description of the XFEM. The crack can be described via discontinuous enrichment functions which differ from crack tip and crack surface. Extracted from [39].

asymptotic solutions of Williams (1952) [38]. The biggest advantage is that these enrichment functions (called *ansatz*) can be activated and deactivated as needed; therefore, taking care of any crack plane bias and major mesh dependency. While initially these enriched functions were composed by discontinuous or even singular functions which were incompatible with the classical FEM polynomial shape functions, Ventura in 2003 [39] was able to successfully map the enriched functions to equivalent polynomials that could be easily integrated by a standard Gauss rule as used in classical FEM. Figure 2.5 shows the distinction between the crack tip and the general crack surface. Overall, this new method which is still under development, yielded extremely good results and posed as a valid alternative to classical FEM despite quadrature and blending issues, considering that the crack initiation and propagation does not need remeshing at all.

As a side note, XFEM is very similar to Generalized Finite Element Method (GFEM). The term GFEM was coined by Strouboulis, Copps and Babuska [40], which similarly to XFEM was to describe the approach of incorporating enrichment functions to the FE framework. The main difference is that XFEM specifically targets problems with discontinuities by introducing additional functions that represent the displacement jump across the cracks and interfaces, while GFEM introduces functions to capture general localized behaviors.

2.5 Note on Quasicontinuum Models

Given the distinct differences between discrete and continuum models, researchers have also tried to implement models that use both discrete and continuum models *distinctively* within the same structure in order to achieve better results and lower run times. Emphasis on the word *distinctively*, as the reader might confuse them with the Crack-Embedded Models. In the Crack-Embedded Models, the geometrical discontinuity (inherently discrete) of the crack is merged within the constitutive behavior of a continuum element. In Quasicontinuum Models instead, both models are used in distinct parts of the structure. Generally, continuum elements allow faster simulations at the cost of lower resolution, so in regions of higher damage or where a crack might start or propagate, discrete elements are used instead to better capture the fracturing behavior of the structure. Quasicontinuum Models have been used in many different fields, as early as in 1989 by Chou [41], where DNA filaments were modeled using the aforementioned approach to study low-frequency motion. In structures, Shenoy in 1998 [42] used it to better model interfacial behavior and deformation at microscopic scale for metals. Recent researchers [43] used a Quasicontinuum approach as it is a suitable multiscale approach that reduces the computational cost of lattice models and allows the incorporation of local lattice defects in large-scale problems.

2.6 Final Motivations

Now that a broad overview of all the current state-of-the-art and past methods was given to the reader, a final note can be presented on why the need to use a discrete approach to model the fracturing of composites and ease the transition to the next chapter. As extensively explained in section 1.1, the fracturing of composites is complex and multi-faceted and many failure mechanisms may occur at the same time. Continuum approaches are very powerful and flexible as they can smoothly represent stress and strain fields and yield accurate results even for large scales. However, when they try to represent cracks, defects, or notches, they suffer of mesh dependency, numerical instabilities and they can only represent discontinuities in a limited manner. These disadvantaged are instead the bread and butter of discrete approaches, which while they might need a more complex

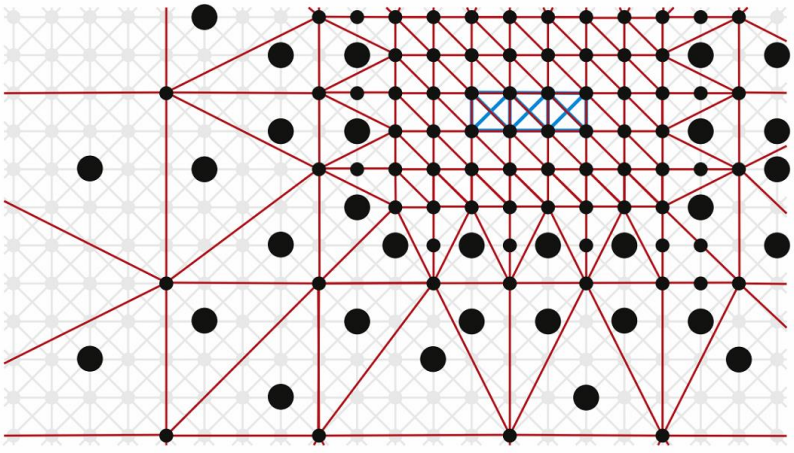


Figure 2.6: Example of discrete lattice structure which is modified in such a way on the region of interest relies on the discrete lattice members while far away an homogenized approach is used. Extracted from [43].

geometry generator and might suffer from higher computational costs, they are much more flexible in modeling discontinuities and capturing local effects. And again, given the variety of damage that needs to be captured in the damaging of composites, a discrete approach such as the proposed FastDM4C is suitable and warranted for these sort of problems.

Chapter 3

THEORETICAL FRAMEWORK

In the presented FastDM4C model, a composite laminate is decomposed into single plies, where a multitude of arrays of rectangular or square unit cells composed by beam elements and new elements called Facet Elements (FE) as shown in Figure 3.1 are aligned with the direction of the lamina. The connection and kinematics constraints of the unit cell allow its members to behave macroscopically as a transversely isotropic material (composite ply). The connection between each layer are the edges resulting from a constrained Delaunay triangulation using the nodes of each ply where instead of keeping the tetrahedrons elements, only the edges whose nodes are on different plies are retained. At that point optimization routines are used to defined the best geometric, elastic, and fracture material parameter that will match a given set of macro-scale properties via an explicit Finite Element solver.

3.1 Finite Element Framework

3.1.1 Introduction

The Finite Element Method (FEM) is a widely used numerical method firstly developed at the University of Washington in 1956 [44]. It is used to solve complex physical problems by discretizing the problem domain into smaller and simpler domains which are indeed called finite elements where the application of constitutive equations and boundaries constraints are easy to implement and otherwise difficult if not impossible to perform with a closed-form solution. There are two main categories of solvers within the Finite Element framework: implicit and explicit methods. Implicit methods involve solving a system of equations by inverting the reduced global stiffness matrix to find the new nodal displacements. The displacements calculated at the next time step are dependent on the nodal velocities and acceleration of the next time step as well and that is the reason why this method is called implicit. On the other hand, explicit methods involve solving the equations sequentially for

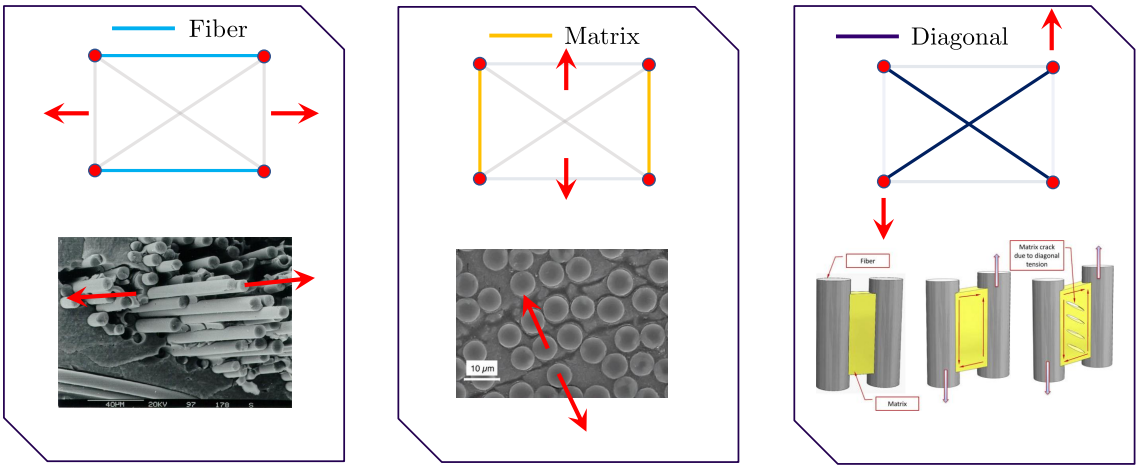


Figure 3.1: Description of the three members that forms the representative unit cell used in the model. The fiber elements represent the fibers found in CFRP, stiff but brittle. The normal elements represent the matrix in CFRP, which is compliant and resilient, while the diagonal elements are oblique in order to mimic the interaction between the two other types of elements, with properties that range in between the fiber and matrix properties.

each element and they are called as such because the nodal displacements of the next time step are only dependent on the nodal velocities and acceleration of the current time steps. The objective of this chapter is to give the reader a basic understanding of the numerical framework used, highlighting some differences between different methods and categorize them appropriately for their different uses.

3.1.2 Implicit Methods

Implicit methods refer to numerical techniques that solve governing differential equations of a mathematical problem using time integration schemes where all the solution at a given time steps depends on the solutions at previous time steps. The most common methods used in Finite Element analysis include [45]:

1. Backward Euler Method: not to confuse with the standard Euler method which is an explicit method, this method has error of order one in time and it computes a solution using $y_{k+1} = y_k + h(t_{k+1}, y_{k+1})$. The new approximation y_{k+1} appears on both sides

Table 3.1: Common parameters used for the Newmark's generalized Beta method.

Parameters	Description
$\beta = 1/4,\ \gamma = 1/2$	Implicit and Unconditionally Stable
$\beta = 1/6,\ \gamma = 1/2$	Linear Acceleration Method
$\beta = 0,\ \gamma = 1/2$	Central Difference Method
$2\beta \geq \gamma \geq 1/2$	Stable regardless of time step h
$\gamma < 1/2$	Conditionally Stable
$\gamma = 1/2$	At least second order accurate

of the equation. It is unconditionally stable, so it can tackle large time steps; however, it tends to be very stiff and it might not capture high-frequency behaviors.

2. Crank-Nicolson Method: This is a second-order implicit method that calculates the solution at half of the current time step using the solutions at the beginning and at the end of the same time step. It can be written as an implicit Runge-Kutta method and it is also unconditionally stable and can capture high-frequency behaviors. The approximation is given by:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{1}{2}\left[F_i^{n+1}\left(u, x, t, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}\right) + F_i^n\left(u, x, t, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}\right)\right]$$

3. Newmark's Methods: These are a family of multiple implicit methods, which include the linear acceleration method, the average acceleration method and the more commonly used generalized-alpha method [46]. All these methods are second-order accurate and can handle also highly non-linear problems; however, they are conditionally stable, meaning that the time steps must be carefully chosen to ensure numerical stability. The approximation is given by:

$$u_i^{n+1} = u_i^n + hv_i^n + h^2\left[(/frac12 - \beta)\, a_i^n + \beta a_i^{n+1}\right]$$
$$v_i^{n+1} = v_i^n + h[(1-\gamma)a_i^n + \gamma v_i^{n+1}]$$

and depending on the values of β and γ, different algorithms can be used, as shown in table 3.1.

Many more algorithms are available [45, 46], but the ones aforementioned are the most commonly used. While the final implementation of the thesis is done via an explicit method, during the initial development of the model, most of the linear elastic calibration simulations were performed using standard implicit methods as the one offer in Abaqus [47] but also using codes developed in the MAMS Research Lab at University of Washington using Matlab [48]. The codes developed in Matlab were implemented using an implicit method with fast sparse-matrices implementation.

3.1.3 Explicit Methods

Differently from the implicit methods listed in section 3.1.2, explicit methods refer to the numerical techniques where the solutions at the next time step only depend on the solutions of the current time step. Explicit methods are suitable for problems where deformations are generally large in short-duration events, such as crashes, explosions, impacts, and high material non-linearities. These methods are conditionally stable, which means they need to satisfy the Courant-Friedrichs-Lewy (CFL) condition. As for the implicit methods, there are a multitude of explicit methods. For example, for certain parameters of β and γ in the Newmark's method in section 3.1.2, such method becomes explicit. In most academic and industry finite element explicit solvers, the method of choice is the central difference method.

This method is graphically summarized in fig. 3.2 and as shown in the equations below, it is also extremely straightforward. Given a time step t^n, the half steps can be calculated using:

$$\begin{aligned} t^{n+\frac{1}{2}} &= \frac{t^{n+1} + t^n}{2} \\ t^{n-\frac{1}{2}} &= \frac{t^n + t^{n-1}}{2} \end{aligned} \tag{3.1}$$

With these definitions, from the definition of half velocities, it is possible to find an explicit equation for the displacements at the next time step.

$$v^{n+\frac{1}{2}} = \frac{\left(d^{n+1} - d^n\right)}{\Delta t^{n+\frac{1}{2}}} \quad \Longrightarrow \quad d^{n+1} = v^{n+\frac{1}{2}} \Delta t^{n+\frac{1}{2}} + d^n \tag{3.2}$$

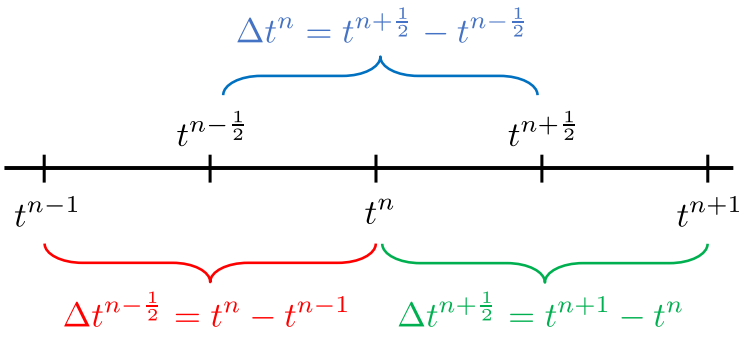


Figure 3.2: Graphical explanation of the Central Difference Scheme used in general Finite Element Solvers.

However, the half velocities are still unknown, but they can be easily extracted from the definition of the current time step acceleration:

$$a^n = \frac{v^{n+\frac{1}{2}} - v^{n-\frac{1}{2}}}{\Delta t^n} \quad \Longrightarrow \quad v^{n+\frac{1}{2}} = a^n \Delta t^n + v^{n-\frac{1}{2}} \tag{3.3}$$

Now, given the basic definition of Newton's second law:

$$[M]\{a\} + [C]\{v\} + [K]\{d\} = \{f\}^{ext} - \{f\}^{int} \tag{3.4}$$

the accelerations are found:

$$a^n = [M]^{-1}\{f\}^{ext} - \{f\}^{int} \tag{3.5}$$

Plugging eqs. (3.1), (3.3) and (3.5) into eq. (3.2) yields to equation for the displacements at the next time step:

$$d^{n+1} = d^n + \Delta t^{n+\frac{1}{2}} \left\{ v^{n-\frac{1}{2}} + \Delta t^n [M]^{-1} \left(\{f\}^{ext} - \{f\}^{int} \right) \right\} \tag{3.6}$$

Since this is an explicit method, d^{n+1} depends only on values at the current or previous time steps.

As mentioned, this method is conditionally stable; therefore, it has to satisfy the CFL condition. The critical time step is based on the ratio between the smallest element size and the longitudinal wave velocity:

$$\Delta t^{crit} = \min\left(\frac{l^{crit}}{\sqrt{E/\rho}}\right)$$
$$\Delta t = \alpha \Delta t^{crit} \tag{3.7}$$

where α is defined as the Courant's number and it is generally a value between 0.2 and 0.95 [46]. Moreover, conservation of energy needs to be checked as well. Given the definitions of internal and external work as:

$$W_{int}^{n+1} = W_{int}^{n} + (d^{n+1} - d^{n})\left(\frac{f_{int}^{n+1} + f_{int}^{n}}{2}\right)$$
$$W_{ext}^{n+1} = W_{ext}^{n} + (d^{n+1} - d^{n})\left(\frac{f_{ext}^{n+1} + f_{ext}^{n}}{2}\right) \tag{3.8}$$
$$W_{kin} = \frac{1}{2}\{v^{n}\}^{T}[M]\{v^{n}\}$$

the following condition needs to be satisfied for small values of ϵ depending on the type of simulation.

$$|W_{kin} + W_{int} - W_{ext}| < \epsilon \cdot \max\left(W_{kin}, W_{int}, W_{ext}\right) \tag{3.9}$$

3.1.4 Mass Scaling and Loading Speed Rate

Implicit methods allow for much larger time steps compared to explicit methods; however, they are not guarantee to converge and therefore are not used very often for fracture simulations as variables might unexpectedly reach instability. Explicit methods tackle this problem by reducing the time step sometimes even to values on the order of 10^{-7}s. This allows for great numerical stability at the cost of running time. Even a simulation with a total running time of 1s might take hours, days, even weeks to run if the time step is 10^{-7}s depending on the size of the problem. This can be solved by the use of two different techniques:

1. **Mass-Scaling**: in this technique non-physical mass is added to a structure in order

to achieve larger explicit time steps. Users might use a global mass scaling value by changing for example the whole density value of the model, or they might opt for a better approach such as an elemental-based mass scaling. In the latter, given a target time step, each element is properly scaled (by adding mass to its nodes) so that the overall time step of the model is indeed the target one. This is particularly advantageous because if smaller elements are caused by an ill meshed, they are not driving the entire time simulation down.

2. **Loading Speed Rate**: in this technique, the speed of the load at which the structure is loaded is numerically increased many order of magnitudes to lower the total running time. This works particularly well if the load applied is either quasi-static or not as fast as the wave propagation speed in the material.

Coupling the mass-scaling with the increase of loading speed rate, even with explicit simulations it is possible to achieve manageable running times. However, there is not a closed-form or recipe to know exactly a priori which values of mass-scaling or how fast to load the specimen to achieve the best combination possible. It is usually done by a lot of trial and error, together with personal knowledge of the material system and structure properties.

3.2 Discrete Fiber Model (DFM)

3.2.1 Introduction

The Discrete Fiber Model is a formulation in the class of the beam finite elements. In the following description, the term fiber is used both for individual fibers at the micro-scale and for tows or group of fibers at the meso-scale. Fibers are modeled as one-dimensional beam elements capable of axial forces, shear, bending and torque. Individual fibers are discretized using a string of beam finite elements. To properly simulate the local fracturing of fibers, it is assumed that fracture localizes at the mid section of an element. This is achieved by postulating the displacement field as follows. Each beam is split into two segments of equal length. The two segments are separately tied to each of the nodes that define the beam. Two segments from adjacent beam elements that share the same node form a rigid cell that

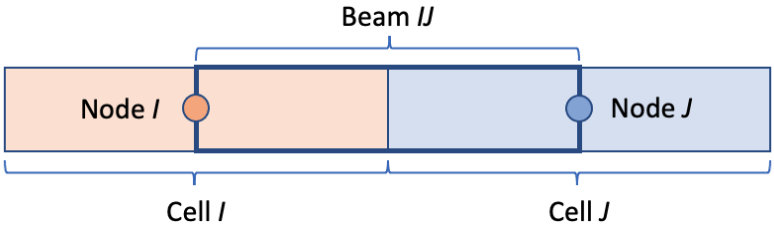


Figure 3.3: Cells consists of segments of beams that share a common node.

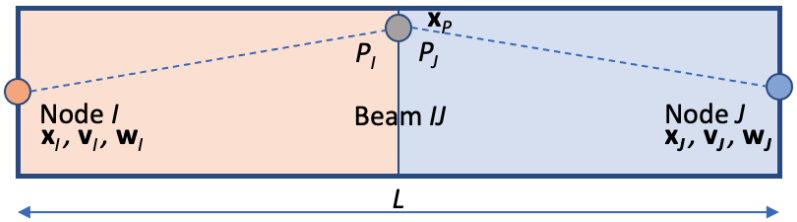


Figure 3.4: Rigid body motion is used to compute the velocity jump at the interface between two cell. This is the cross-section at the midpoint of the beam.

is tied to the six Degrees of Freedom (DOF)'s the shared node as shown in fig. 3.3. Thus, the velocity field of cell I can be defined using rigid body equations as:

$$\mathbf{v}(P) = \mathbf{v}_I + \mathbf{w}_I \left(\mathbf{x}(P) - \mathbf{x}_I\right) \tag{3.10}$$

where $\mathbf{x}_I$, $\mathbf{v}_I$ and $\mathbf{w}_I$ are the coordinate, velocity and rotation rate vectors of node I at the center of the cell, $\mathbf{x}(P)$ and $\mathbf{v}(P)$ are the coordinate and velocity vector of an arbitrary point inside the cell.

The vectorial stress-strain state in a beam is characterized at the points on the beam middle cross section. Let P_I and P_I be two material points on the mid-section that at time zero share the same location, see fig. 3.4. The velocity step $\Delta\mathbf{v}_P$ at point P is defined as:

$$\Delta\mathbf{v}_P = \mathbf{v}(P_J) - \mathbf{v}(P_I) \tag{3.11}$$

where P_J is the material point associated to cell J and P_I is the material point associated

to cell I.

The strain rate vector $\dot{\boldsymbol{\varepsilon}}_P$ at P is defined as

$$\dot{\boldsymbol{\varepsilon}}_P = \frac{\Delta \mathbf{v}_P}{L} \tag{3.12}$$

where L is the distance between N_I and N_J. Note that this expression for $\dot{\boldsymbol{\varepsilon}}_P$ is the same equation that results from the Timoshenko beam theory. The strain rate vector can be directly related to the velocity vector $\mathbf{v}_{IJ}$ through the B-matrix $\mathbf{B}_{IJP}$:

$$\dot{\boldsymbol{\varepsilon}}_P = \mathbf{B}_{IJP}\ \mathbf{v}_{IJ} \qquad \text{where} \qquad \mathbf{v}_{IJ}^T = |\mathbf{v}_I^T,\ \mathbf{w}_I^T,\ \mathbf{v}_J^T,\ \mathbf{w}_j^T| \tag{3.13}$$

The $\dot{\boldsymbol{\varepsilon}}_P$ vector can be decomposed into three components: a normal component $\dot{\varepsilon}_{Pn}$ aligned with the direction $N_I - N_J$, and two shear components $\dot{\varepsilon}_{Pl}$ and $\dot{\varepsilon}_{Pm}$, perpendicular to $\dot{\varepsilon}_{Pn}$ and to each other. The strain rates are used in the constitutive equations to compute the evolution of the stress vector:

$$\boldsymbol{\sigma}_P = \Phi(\dot{\boldsymbol{\varepsilon}}_P) \tag{3.14}$$

The DFM formulation uses a very general integration scheme, which relies on a two-dimensional finite element mesh of triangular or quadrilateral elements. Figure 3.5 shows the cross-section discretization of a circular cross section. The center points of the N elements are the integration points P_k which operate on the corresponding areas A_k. Thus, the cross section integration scheme computes N strain rate vectors $\dot{\boldsymbol{\varepsilon}}_k$ and stress vectors $\boldsymbol{\sigma}_k$. The resulting forces and moments at the nodes are computed using the principle of virtual power:

$$\delta P = \mathbf{f}_{IJ}\ \delta \mathbf{v}_{IJ} = \sum_{k=1}^{N} \boldsymbol{\sigma}_k\ \delta \dot{\boldsymbol{\varepsilon}}_k\ A_k\ L = \sum_{k=1}^{N} \boldsymbol{\sigma}_k\ \mathbf{B}_{IJk}\ \delta \mathbf{v}_{IJ}\ A_k\ L \tag{3.15}$$

Hence,

$$\mathbf{f}_{IJ} = \sum_{k=1}^{N} \boldsymbol{\sigma}_k\ \mathbf{B}_{IJk}\ A_k\ L \qquad \text{where} \qquad \mathbf{f}_{IJ}^T = |\ \mathbf{f}_I^T,\ \mathbf{m}_I^T,\ \mathbf{f}_I^T,\ \mathbf{m}_I^T\ | \tag{3.16}$$

where $\mathbf{f}_I$ is the force vector at node I, $\mathbf{m}_I$ is the moment vector at node I.

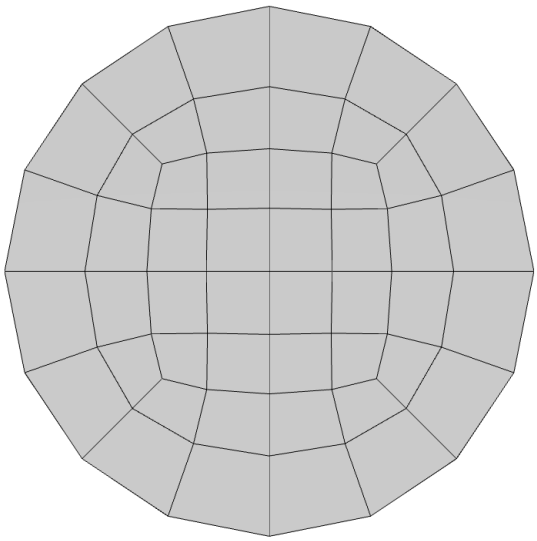


Figure 3.5: Beam Cross Section Integration Scheme.

This formulation assumes that large fracture strains localize at the mid-section of a beam and approximates the small deformation behavior of a cell as rigid. As such it is possible to capture the macroscopic failure of a fiber in tension, without resorting to element erosion. Furthermore, this formulation is able to capture the kink fracture of a fiber in compression.

3.2.2 FDM Constitutive Relationships

The material model for the fiber material is described in this section. The axial stresses are evaluated at the integration points in the cross-section of a beam element. In the elastic regime, the stress vector ($\sigma_n,\ \sigma_{s1},\ \sigma_{s2}$ is linearly proportional to the strain vector ($\varepsilon_n,\ \varepsilon_{s1}, \varepsilon_{s2}$) as follows:

$$\begin{aligned} \sigma_n &= E_n \varepsilon_n \\ \sigma_{s1} &= E_s \varepsilon_{s1} \\ \sigma_{s2} &= E_s \varepsilon_{s2} \end{aligned} \tag{3.17}$$

where E_n is the normal modulus and E_s is the shear modulus of the fiber bundle. E_s is actually defined as:

$$E_s = \frac{E_n}{1-\nu} \tag{3.18}$$

The parameters E_n and ν are calibrated to match the elastic properties of the composite material under consideration.

When the axial tensile stress reaches the tensile strength limit f_t, the stresses in eq. (3.17) start to follow a linear or bilinear softening law. Equation (3.17) can be rewritten in more general terms as:

$$\begin{aligned} \sigma_n &= E_n(1-D)\varepsilon_n \\ \sigma_{s1} &= E_s(1-D)\varepsilon_{s1} \\ \sigma_{s2} &= E_s(1-D)\varepsilon_{s2} \end{aligned} \tag{3.19}$$

Equation (3.19) shows that the damage can be implemented as a scalar variable that deteriorates the stiffness of the element in order to have it follow a particular curve. While the softening curve is simply linear, the damage variable D will be highly non-linear and will differ for both linear and bi-linear case. Both cases can be written as a function of strain (ε) or Crack Mouth Opening Displacement (CMOD). CMOD is defined as the measure on the loading line or on the surface of the specimen as the difference between the original and final crack opening and it will be zero until the strain will pass the elastic limit (or the stress passes the peak strength). For a simple linear law, the damage variable D is defined as a function of CMOD:

$$D = \frac{w_f(w - w_{\text{elastic}})}{w(w_f - w_{\text{elastic}})} \tag{3.20}$$

while for a bilinear law, using strain definition instead of CMOD (the only difference is that at the value of strain, it needs to be subtracted the value of strain at strength peak), for each branch of the traction separation law it is possible to write the following equations:

$$\begin{aligned} D^1(\varepsilon) &= 1 - \frac{\left(\varepsilon - \frac{f_t}{E}\right) s_1(\varepsilon) + f_t}{E\varepsilon} \\ D^2(\varepsilon) &= 1 - \frac{(\varepsilon - \varepsilon_k) s_2(\varepsilon) + f_k}{E\varepsilon} \end{aligned} \tag{3.21}$$

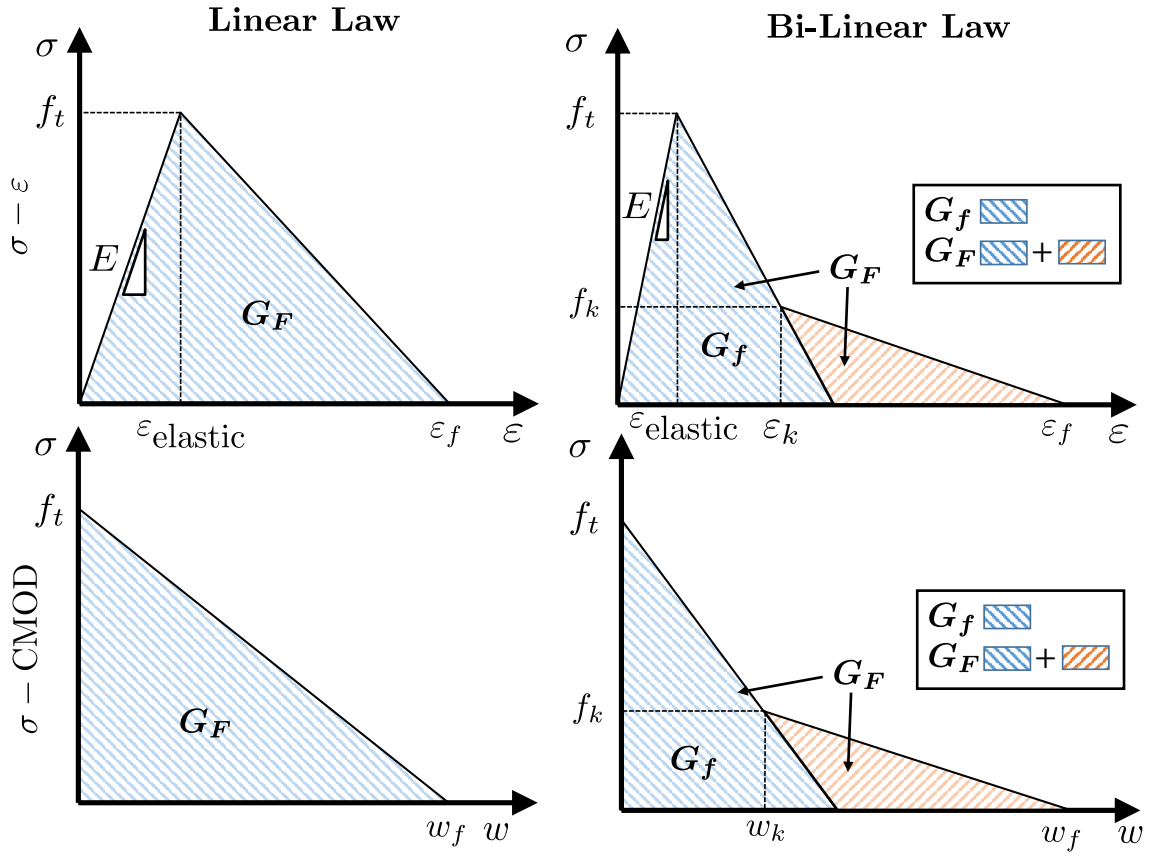


Figure 3.6: Schematic representation of a Linear Traction Separation Law (left column) and of Bi-Linear Traction Separation Law (right column), both expressed in terms of Strain (top row) and Opening Displacement (bottom row).

Where $E = E_n$ and $D^1(\varepsilon)$ and $D^2(\varepsilon)$ describe the damage scalar value of each softening branch. For sake of completeness, also the slope of the two branches can be found analytically:

$$\begin{aligned} s_1(\varepsilon) &= \frac{f_k - f_t}{\varepsilon_k - \frac{f_t}{E}} \\ s_2(\varepsilon) &= \frac{f_k}{\varepsilon_f - \varepsilon_k} \end{aligned} \tag{3.22}$$

where ε_k, ε_f, graphically shown in Figure 3.6 can be again described as a function of kinking

strain ε_k and failure strain ε_f:

$$\begin{aligned} \varepsilon_k &= \frac{f_k}{E} + 2G_F^* \left(\frac{1}{f_t} - 1\right) \\ \varepsilon_f &= f_t \left(\frac{\varepsilon_k}{f_k} + \frac{1}{E}\right) + \frac{2G_F^*}{f_k} \end{aligned} \tag{3.23}$$

To eliminate mesh dependency using cohesive elements [49], the fracture energies must be scaled accordingly to the size of the element chosen:

$$\begin{aligned} G_F^* &= G_F^{\text{total}} h^{\text{element}} \\ G_f^* &= G_f^{\text{total}} h^{\text{element}} \end{aligned} \tag{3.24}$$

Equations (3.21) to (3.24) can fully described any bilinear behavior for any given energy ratio γ ($\gamma = Gf/GF$), but something to keep in mind is that ε_k must be always greater than $\varepsilon_{\text{elastic}} = f_t/E$ in order to avoid positive slope for the first softening branch. If this is satisfied, re-arranging eq. (3.23) yields to:

$$\gamma > \frac{f_t^2}{2EG_F^*} \quad \text{for} \quad \varepsilon_k > \frac{f_t}{E} \tag{3.25}$$

Similarly, assuming that ε_k has to always be less ε_f, it is possible to find the limit for the second branch:

$$\gamma < \frac{f_t^2[2EG_F^* + f_k(2f_t - f_k)]}{2EG_F^*(f_t - f_k)^2} \quad \text{for} \quad \varepsilon_k < \varepsilon_f \tag{3.26}$$

A neat example which takes advantage of the cross section implementation and the damage is shown in fig. 3.7. In this way it is possible to capture explicitly the buckling of a single fiber. Materials don't fail in compression, but rather due to the shearing effect and instability, some parts will snap into tension and then eventually break, and this is very clear in fig. 3.7.

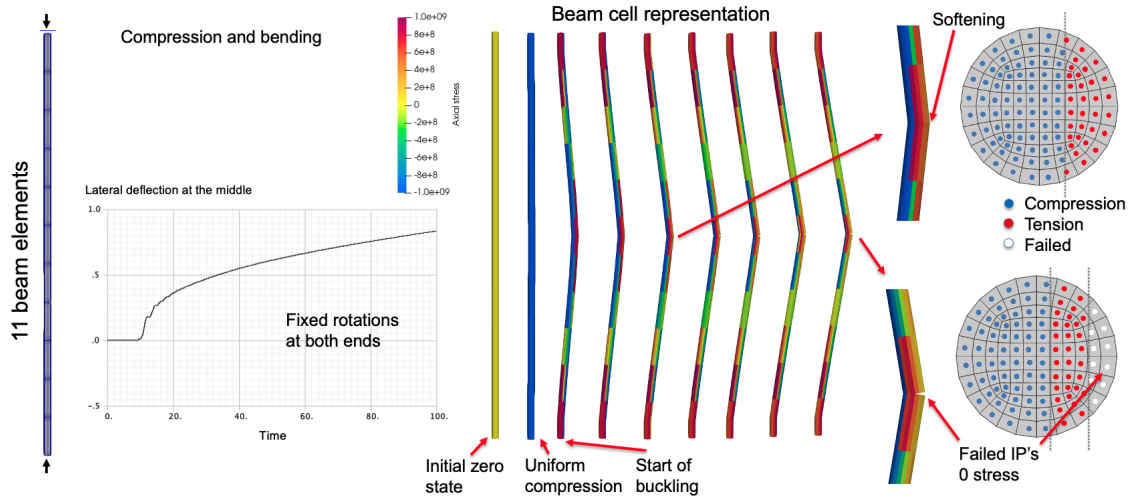


Figure 3.7: Example of a single beam in buckling. The formulation allows to explicitly and nicely capture the buckling behavior in a single element. At some time, some integration points will go in tension and initiate the damage while the others will remain in compression.

3.3 Discrete Matrix Model (DM2)

3.3.1 Introduction

The Discrete Matrix Model is used for modeling the structural contribution of the matrix material to the overall behavior of the composite. DM2 employs a solid and discrete representation of the matrix. This formulation has some vague similarities to LDPM which was specifically intended for modeling concrete materials; nonetheless, it has quite many differences aimed to model only composite materials and therefore it should be considered an independent formulation.

In DM4C the matrix is modeled as an ensemble of tetrahedral elements whose edges are used to create quadrilateral facets where DM2 is applied. But in FastDM4C, only the edges of the tetrahedra that go from one ply to another are used to generate the DM2 facets. In DM4C the six facets (one per edge of the tetrahedron) is oriented in specific directions to capture the most probable failure pattern; however, in FastDM4C the facet is oriented specifically in one of the three directions, either matrix (parallel to the fibers), diagonal

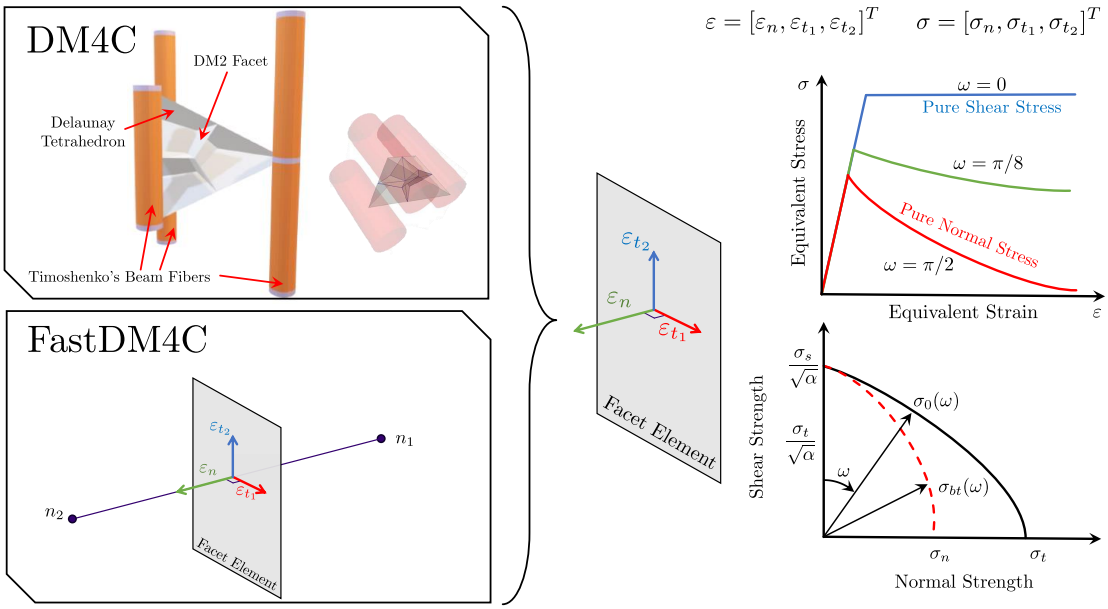


Figure 3.8: Graphical explanation of how the facets are modeled both in DM4C and FastDM4C. While their generation is different, the constitutive equations are exactly the same for both.

(cross-membered to the fibers), and interface (parallel to the layup). Apart from these aforementioned differences, the DM2 becomes exactly the same for DM4C and FastDM4C, and the following equations hold true for both models.

3.3.2 DM2 Constitutive Relationships

The material model which governs the vectorial equations at the facets is specifically formulated for simulating the elasto-plastic behavior of the matrix materials commonly employed in composites and a rigorous treatment of the volumetric strain that makes it possible to achieve Poisson's ratios above 0.3 which are not achievable with LDPM. A graphical visualization of the facets and their constitutive modeling, which will be covered right after, is shown in

The displacement field $\boldsymbol{u}$ inside each element is defined following the kinematic relation

of nodes as follows:

$$\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{u}_i + \boldsymbol{\theta}_i \times (\boldsymbol{x} - \boldsymbol{x}_i) = \boldsymbol{A}_i(\boldsymbol{x})\boldsymbol{Q}_i \tag{3.27}$$

where

$$\boldsymbol{A}_i(\boldsymbol{x}) = \begin{bmatrix} 1 & 0 & 0 & 0 & x_3 - x_{3i} & x_{2i} - x_3 \\ 0 & 1 & 0 & x_{3i} - x_3 & 0 & x_1 - x_{1i} \\ 0 & 0 & 1 & x_2 - x_{2i} & x_{1i} - x_1 & 0 \end{bmatrix} \tag{3.28}$$

The vector $\boldsymbol{x}$ is the Cartesian coordinate inside the element, and the vector $\boldsymbol{x}_i$ represents the coordinate of node i. The vector $\boldsymbol{Q}_i^T = \left[\boldsymbol{u}_i^T, \boldsymbol{\theta}_i^T\right]$ is the ordered pair of the vector $\boldsymbol{u}_i^T = [u_{1i}, u_{2i}, u_{3i}]$ and the vector $\boldsymbol{\theta}_i^T = [\theta_{1i}, \theta_{2i}, \theta_{3i}]$, which are the translational and rotational DOF of node i.

The strain components in each projected facet are computed following:

$$\begin{aligned} \varepsilon_{Nk} &= \boldsymbol{B}_N^{jk}\boldsymbol{Q}_j - \boldsymbol{B}_N^{ik}\boldsymbol{Q}_i \\ \varepsilon_{Mk} &= \boldsymbol{B}_M^{jk}\boldsymbol{Q}_j - \boldsymbol{B}_M^{ik}\boldsymbol{Q}_i \\ \varepsilon_{Lk} &= \boldsymbol{B}_L^{jk}\boldsymbol{Q}_j - \boldsymbol{B}_L^{ik}\boldsymbol{Q}_i \end{aligned} \tag{3.29}$$

where:

$$\begin{aligned} \boldsymbol{B}_N^{pk} &= (1/\ell_e)\, \boldsymbol{n}_k^T \boldsymbol{A}_p(\boldsymbol{x}_{Ck}) \\ \boldsymbol{B}_M^{pk} &= (1/\ell_e)\, \boldsymbol{m}_k^T \boldsymbol{A}_p(\boldsymbol{x}_{Ck}) \\ \boldsymbol{B}_L^{pk} &= (1/\ell_e)\, \boldsymbol{l}_k^T \boldsymbol{A}_p(\boldsymbol{x}_{Ck}) \end{aligned} \tag{3.30}$$

for $p = i, j$. The vectors $\boldsymbol{n}_k = (\boldsymbol{x}_j - \boldsymbol{x}_i)/\ell_e$, $\boldsymbol{m}_k$, and $\boldsymbol{l}_k$ compose a set of mutually orthogonal axes for the plane of the projected facet.

Once the stress vector for each facet is obtained through the constitutive relation, the nodal forces can be computed via the Principle of Virtual Work, which imposes that the total external work applied should be identical to the sum of internal work from all the facets. This leads to the following equation defining the internal energy associated with the facet k:

$$\begin{aligned}\delta W_k &= \ell_k A_k \boldsymbol{\sigma}_k^T \delta \boldsymbol{\varepsilon}_k \\ &= \ell_k A_k \left(\sigma_{Nk}\delta\varepsilon_{Nk} + \sigma_{Mk}\delta\varepsilon_{Mk} + \sigma_{Lk}\delta\varepsilon_{Lk}\right)\end{aligned} \tag{3.31}$$

where A_k is the area of a projected facet k.

By substituting the relations between strain components and the nodal DOF to eq. (3.31), the nodal force vectors, $\boldsymbol{F}_{ik}$ and $\boldsymbol{F}_{ik}^T$, for nodes i and j associated to the facet k can be computed as follows:

$$\begin{aligned}\delta W_k &= \boldsymbol{F}_{ik}^T \delta \boldsymbol{Q}_i + \boldsymbol{F}_{jk}^T \delta \boldsymbol{Q}_j, \quad \text{where} \\ \boldsymbol{F}_{ik}^T &= -\ell_k A_k \left(\sigma_{Nk}\boldsymbol{B}_N^{ik} + \sigma_{Mk}\boldsymbol{B}_M^{ik} + \sigma_{Lk}\boldsymbol{B}_L^{ik}\right) \\ \boldsymbol{F}_{jk}^T &= \ell_k A_k \left(\sigma_{Nk}\boldsymbol{B}_N^{jk} + \sigma_{Mk}\boldsymbol{B}_M^{jk} + \sigma_{Lk}\boldsymbol{B}_L^{jk}\right)\end{aligned} \tag{3.32}$$

Elastic Behavior

The elastic relationships for each component of strain and stress vectors resembles the Hooke's law in one-dimension:

$$\begin{aligned}\sigma_N &= E_0 \varepsilon_N \\ \sigma_M &= \alpha E_0 \varepsilon_M \\ \sigma_L &= \alpha E_0 \varepsilon_L\end{aligned} \tag{3.33}$$

where E_0 is effective mesoscale normal modulus, and α is a shear-normal coupling parameter. E_0 and α are the DM4C matrix properties defined at the facet level that are calibrated to match the macroscopic elastic properties of a composite material in the transverse direction which is dominated by the properties of the matrix.

Tensile Fracture Behavior

When a facet deforms under positive normal strain, fracturing behavior determines the softening of the stress components. Following the LDPM formulation, this behavior is characterized with the effect strain, ε, and the effective stress, σ:

$$\begin{aligned} \varepsilon &= \sqrt{\varepsilon_N^2 + \alpha\left(\varepsilon_M^2 + \varepsilon_L^2\right)} \\ \sigma &= \sqrt{\sigma_N^2 + \left(\sigma_M^2 + \sigma_L^2\right)/\alpha} \end{aligned} \tag{3.34}$$

The effective stress incrementally increases proportional to the effective strain until it reaches the tensile boundary σ_{bt}. As expressed in eq. (3.35), the tensile boundary σ_{bt} has an exponential relationship with the normal-shear ratio variable ω and the history-dependent maximum effective strain ε_{max} following the Macaulay's notion: $\langle x \rangle = \max(x, 0)$.

$$\sigma_{bt}(\varepsilon, \omega) = \sigma_0(\omega) \exp\left(-H_0(\omega) \frac{\langle \varepsilon_{max} - \varepsilon_0(\omega) \rangle}{\sigma_0(\omega)}\right) \tag{3.35}$$

The normal-shear ratio variable ω is computed as

$$\tan\omega = \frac{\varepsilon_N}{\sqrt{\alpha}\varepsilon_T} \tag{3.36}$$

where $\varepsilon_T = \sqrt{\varepsilon_M^2 + \varepsilon_L^2}$. This variable indicates the degree of interaction between shear and normal load in a facet.

In eq. (3.35), $\sigma_0(\omega)$ represents the strength limit of the effective stress and expressed as:

$$\sigma_0(\omega) = \sigma_t \frac{-\sin(\omega) + \sqrt{\sin^2(\omega) + 4\alpha\cos^2(\omega)/r_{st}^2}}{2\alpha\cos^2(\omega)/r_{st}^2} \tag{3.37}$$

where $r_{st} = \sigma_s/\sigma_t$ is the ratio between the shear strength σ_s and the tensile strength σ_t. The elastic limit $\varepsilon_0(\omega)$ is obtained by $\sigma_0(\omega)/E_0$.

The rate of decay of the tensile boundary $H_0(\omega)$ is also dependent on ω and is defined as

$$H_0(\omega) = H_t \left(\frac{2\omega}{\pi}\right)^{n_t} \tag{3.38}$$

where $H_t = 2E_0/(\ell_t/\ell - 1)$ and $\ell_t = 2E_0 G_t/\sigma_t^2$. G_t is associated to the fracture energy of the matrix, and ℓ is the edge length of the tetrahedron element (in the case of DM4C) or beam element (in the case of FastDM4C) related to the facet in the evaluation.

In summary, there are four parameters to be determined besides the elastic parameters (α, E_0) for the complete description of the matrix tensile failure: σ_s, σ_t, ℓ_t, and n_t.

Hardening Behavior under Compression

A facet under compression exhibits hardening behavior dependent on the interaction between the volumetric strain ε_V and the deviatoric strain $\varepsilon_D = \varepsilon_N - \varepsilon_V$ as the LDPM formulation proposed in [31]. The normal stress component increases incrementally until it reaches the compressive boundary $\sigma_{bc}(\varepsilon_D, \varepsilon_V)$, which is defined as

$$\sigma_{bc}(\varepsilon_D, \varepsilon_V) = \begin{cases} \sigma_{c0} \\ \qquad \text{for} \quad -\varepsilon_{DV} \leq 0 \\ \sigma_{c0} + \langle -\varepsilon_{DV} - \varepsilon_{c0} \rangle H_c(r_{DV}) \\ \qquad \text{for} \quad 0 \leq -\varepsilon_{DV} \leq \varepsilon_{c1} \\ \sigma_{c1}(r_{DV}) \exp((-\varepsilon_{DV} - \varepsilon_{c1}) H_c(r_{DV}) / \sigma_{c1}(r_{DV})) \\ \qquad \text{otherwise} \end{cases} \tag{3.39}$$

where σ_{c0} is the compressive yielding strength of a facet, H_c is the initial hardening modulus, and σ_{c1} is the normal stress value at the onset of the rehardening. In the compressive boundary, $\varepsilon_D V = \varepsilon_V + \beta \varepsilon_D$, and $\varepsilon_{c0} = \sigma_{c0}/E_0$. In the current model, β is assumed to be zero. The onset of the rehardening is determined by $\varepsilon_{c1} = \kappa_{c0}\varepsilon_{c0}$, which can be used to compute the rehardening stress $\sigma_{c1}(r_{DV}) = \sigma_{c0} + (\varepsilon_{c1} - \varepsilon_{c0}) H_c(r_{DV})$.

The initial hardening modulus $H_c(r_{DV})$ is expressed as

$$H_c(r_{DV}) = \frac{H_{c0} - H_{c1}}{1 + \kappa_{c2} \langle r_{DV} - \kappa_{c1} \rangle} + H_{c1} \tag{3.40}$$

where H_{c0}, H_{c0}, and H_{c0} are matrix properties in facet level and $H_{c1} = \kappa_{c3} E_0$. In the current DM4C model, $\kappa_{c3} = 0.1$.

Both σ_{c1} and H_c depends on the deviatoric and volumetric dependent variable, r_{DV},

which has the expression of

$$r_{DV} = \begin{cases} -\dfrac{|\varepsilon_D|}{\varepsilon_V - \varepsilon_{V0}} & \text{for} \quad \varepsilon_V \leq 0 \\ \dfrac{|\varepsilon_D|}{\varepsilon_{V0}} & \text{for} \quad \varepsilon_V > 0 \end{cases} \tag{3.41}$$

where $\varepsilon_{V0} = \kappa_{c3}\varepsilon_{c0}$.

In summary, σ_{c0}, H_{c0}, ε_{c1}, κ_{c1} and , κ_{c2} are calibrated to complete the description of the hardening behavior of a facet under compression.

Frictional Behavior

Facets with compressive normal stress exhibit frictional behavior which increases the shear strength. The effective shear stress:

$$\sigma_T = \sqrt{\sigma_M^2 + \sigma_L^2} \tag{3.42}$$

increases incrementally until it reaches the shear boundary σ_{bs} defined as follows:

$$\sigma_{bs} = \sigma_s + \alpha\left(\mu_0 - \mu_\infty\right)\sigma_{N0} - \alpha\mu_\infty\sigma_N - \alpha\left(\mu_0 - \mu_\infty\right)\sigma_{N0}\exp\left(\frac{\sigma_N}{\sigma_{N0}}\right) \tag{3.43}$$

where σ_s is the shear strength of a facet under pure shear loading condition, and σ_{N0} is the parameter related to the σ_N at which the initial friction coefficient μ_0 transitions to the asymptotic friction coefficient μ_∞.

3.4 Constitutive Modeling of Lattice Members and Composite Laminates

This section should give the reader a basic understanding of the constitutive modeling of composite laminae even if these equations *per se* are not utilized in the model, rather, they offer a way to compare macroscopic results of the model with the ones of the composites via the use of basic mechanics of composites principles [50].

3.4.1 Transversely-Isotropic Elastic Behavior at the Lamina Scale

At the lamina scale, the elastic behavior is generally described leveraging an homogenized continuum approximation. This approximation is typically very accurate and used broadly in academia and industry. Even though the discrete model proposed in this study implements the elastic behavior starting from an inherently lower scale, it is important to make sure that the model is capable of reproduce the transversely isotropic elastic properties measured at the lamina scale. For an homogenized continuum the generalized Hooke's Law that relates stresses to strains can be written in contracted form as:

$$\sigma_i = C_{ij}\varepsilon_j \qquad i,j = 1,...,6 \tag{3.44}$$

where σ_i are the stresses components on a three dimensional cube, C_{ij} is the stiffness matrix and ε_j the strain components. By initial definition the stiffness matrix C_{ij} has 36 constants, however, less than 36 can be used to characterize an elastic material. By writing incremental work per unit volume of an elastic material and upon integration for all strains, it can be shown that the stiffness matrix is actually symmetric [50], reducing the number of independent variables from 36 to 21. A material with 21 independent material constants is called *anisotropic* has it has no planes of symmetry. With one plane of symmetry, the number of variables further reduces to 13 and the material is called *monoclinic*. If a material has two planes of symmetry, it will automatically be normal to the third one has well and such material is called *orthotropic* and has 9 material constants. If at every point on a plane of the material the material properties are all the same, the material is called *transversely isotropic* and it has 5 material constants as shown in the following stiffness matrix:

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_{23} \\ \tau_{31} \\ \tau_{12} \end{Bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{12} & C_{11} & C_{13} & 0 & 0 & 0 \\ C_{13} & C_{13} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & (C_{11}-C_{12})/2 \end{bmatrix} \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_{23} \\ \gamma_{31} \\ \gamma_{12} \end{Bmatrix} \tag{3.45}$$

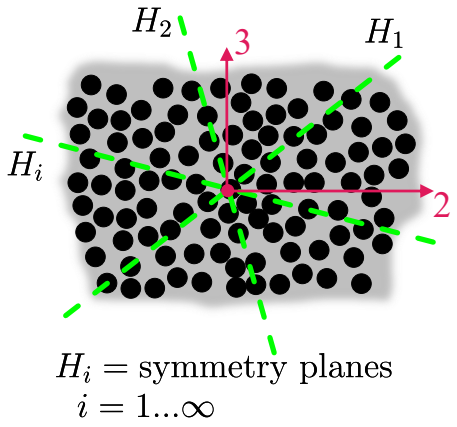


Figure 3.9: Schematic representation the infinite symmetry planes underpinning the transversely-isotropic behavior of unidirectional fiber composites

This case is the result of having an infinite number of symmetry planes in the plane orthogonal to the polar axis of the material as shown in Figure 3.9. Last case, if there are infinite number of planes to which a material is symmetric, it is called *isotropic* and it only requires two material constants as shown in eq. (3.46).

Composite materials behave as *transversely isotropic* materials and the macroscopic model will be calibrated to mimic such properties. However, the single elements in the unit cell behave singularly as *isotropic* materials. Therefore, the orthotropic behavior of the ply is achieved by kinematics constraints and relations of different members of a unit cell.

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_{23} \\ \tau_{31} \\ \tau_{12} \end{Bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{11} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{12} & C_{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & (C_{11}-C_{12})/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & (C_{11}-C_{12})/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & (C_{11}-C_{12})/2 \end{bmatrix} \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_{23} \\ \gamma_{31} \\ \gamma_{12} \end{Bmatrix} \tag{3.46}$$

Unidirectional Case

While eqs. (3.45) to (3.46) are generalized for three dimensional materials, for a unidirectionally reinforced lamina it can be simplified to plane stress state by imposing the following conditions:

$$\begin{aligned} \sigma_3 = 0 \qquad \tau_{23} = 0 \qquad \tau_{31} = 0 \\ \sigma_1 \neq 0 \qquad \sigma_2 \neq 0 \qquad \tau_{12} \neq 0 \end{aligned} \tag{3.47}$$

And the strain-stress relations is simply reduced to:

$$\begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \gamma_{12} \end{Bmatrix} = \begin{bmatrix} S_{11} & S_{12} & 0 \\ S_{12} & S_{22} & 0 \\ 0 & 0 & S_{66} \end{bmatrix} \begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_{12} \end{Bmatrix} \tag{3.48}$$

where by simple unidirectional tests it is possible to obtain:

$$S_{11} = \frac{1}{E_1} \qquad S_{12} = -\frac{\nu_{12}}{E_1} = -\frac{\nu_{21}}{E_2} \qquad S_{22} = \frac{1}{E_2} \qquad S_{66} = \frac{1}{G_{12}} \tag{3.49}$$

Equation (3.48) can be easily inverted to obtain the stress-strain relations and it can be written as:

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_{12} \end{Bmatrix} = \begin{bmatrix} \dfrac{S_{11}}{S_{11}S_{22} - S_{12}^2} & \dfrac{S_{12}}{S_{11}S_{22} - S_{12}^2} & 0 \\ S_{12} & \dfrac{S_{22}}{S_{11}S_{22} - S_{12}^2} & 0 \\ 0 & 0 & \dfrac{1}{S_{66}} \end{bmatrix} \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \gamma_{12} \end{Bmatrix} \tag{3.50}$$

with all the components already found in eq. (3.49). At this point is easy to see that to describe a lamina in plane stress condition only 4 material parameters are needed, the Young modulus in the fibers' direction E_1, the Young modulus normal to the fibers' direction E_2, the in-plane shear modulus G_{12}, and finally the in-plane Poisson's ratio ν_{12}.

Ply Arbitrarily Oriented in a Plane

Equation (3.50) are for a unidirectional ply oriented the first axis in the direction of the fibers; however, in laminates, very often different orientations are seen. While the main

behavior does not change, eq. (3.50) needs to be generalized for any orientation.

This is done by using elementary mechanics [50, 51], where stresses in a 1-2 (unidirectional case for example) reference frame can be written in a x-y reference frame:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = \begin{bmatrix} \cos^2\theta & \sin^2\theta & -2\sin\theta\cos\theta \\ \sin^2\theta & \cos^2\theta & 2\sin\theta\cos\theta \\ \sin\theta\cos\theta & -\sin\theta\cos\theta & \cos^2\theta - \sin^2\theta \end{bmatrix} \begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_{12} \end{Bmatrix} \tag{3.51}$$

However, the transformations are commonly written as:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = [T]^{-1} \begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_{12} \end{Bmatrix} \quad \text{and} \quad \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \frac{\gamma_{xy}}{2} \end{Bmatrix} = [T]^{-1} \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \frac{\gamma_{12}}{2} \end{Bmatrix} \tag{3.52}$$

where $[T]$ is defined as the rotational matrix:

$$[T] = \begin{bmatrix} \cos^2\theta & \sin^2\theta & 2\sin\theta\cos\theta \\ \sin^2\theta & \cos^2\theta & -2\sin\theta\cos\theta \\ -\sin\theta\cos\theta & \sin\theta\cos\theta & \cos^2\theta - \sin^2\theta \end{bmatrix} \tag{3.53}$$

and θ is the angle *from* the x-axis to the 1-axis. At this point eq. (3.50) can be rotated:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = [T]^{-1}[Q][T]^{-T} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} \tag{3.54}$$

and the general for of the stress-strain relationship in a x-y reference system of a lamina is written as:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = [\overline{Q}] \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \begin{bmatrix} \overline{Q}_{11} & \overline{Q}_{12} & \overline{Q}_{16} \\ \overline{Q}_{12} & \overline{Q}_{22} & \overline{Q}_{26} \\ \overline{Q}_{16} & \overline{Q}_{26} & \overline{Q}_{66} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} \tag{3.55}$$

with the following parameters:

$$\begin{aligned}
\overline{Q}_{11} &= Q_{11}\cos^4\theta + 2(Q_{12}+2Q_{66})\sin^2\theta\cos^2\theta + Q_{22}\sin^4\theta \\
\overline{Q}_{12} &= (Q_{11}+Q_{22}-4Q_{66})\sin^2\theta\cos^2\theta + Q_{12}(\sin^4\theta+\cos^4\theta) \\
\overline{Q}_{22} &= Q_{11}\sin^4\theta + 2(Q_{12}+2Q_{66})\sin^2\theta\cos^2\theta + Q_{22}\cos^4\theta \\
\overline{Q}_{16} &= (Q_{11}-Q_{12}-2Q_{66})\sin\theta\cos^3\theta + (Q_{12}-Q_{22}+2Q_{66})\sin^3\theta\cos\theta \\
\overline{Q}_{26} &= (Q_{11}-Q_{12}-2Q_{66})\sin^3\theta\cos\theta + (Q_{12}-Q_{22}+2Q_{66})\sin\theta\cos^3\theta \\
\overline{Q}_{66} &= (Q_{11}+Q_{22}-2Q_{12}-2Q_{66})\sin^2\theta\cos^2\theta + Q_{66}(\sin^4\theta+\cos^4\theta)
\end{aligned} \tag{3.56}$$

where the Q_i components are the components of the matrix in eq. (3.50).

3.4.2 Elastic Isotropic Materials

As mentioned in the previous sections, the proposed modeling framework captures the mechanical behavior of composites by explicitly modeling groups of fibers and surrounding matrix via one-dimensional elements. While at the lamina scale the 1D elements are arranged so to reproduce a transversely-isotropic behavior, the material behavior of each member is assumed elastic isotropic before the material strength is reached. This hypothesis is confirmed by a large number of tests which show that the behavior of the matrix in composites is well approximated by this assumption. In regards to the fibers, the behavior is known to be anisotropic, with elastic moduli and strength being significantly higher in the longitudinal direction. However, the approximation of isotropic behavior is still very reasonable to reproduce the mechanical behavior at the lamina scale. In fact, the elastic behavior of the fibers in the transverse direction has generally a negligible effect on the macroscopic behavior. For this reason, the fibers are modeled as isotropic materials with elastic constants corresponding to the longitudinal moduli as already explained in section 3.2.

For plane stress (assumptions listed in eq. (3.47)) in isotropic materials, the strain-stress relation becomes:

$$\begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \gamma_{12} \end{Bmatrix} = \begin{bmatrix} S_{11} & S_{12} & 0 \\ S_{12} & S_{11} & 0 \\ 0 & 0 & 2(S_{11}-S_{12}) \end{bmatrix} \begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_{12} \end{Bmatrix} \tag{3.57}$$

where similarly the components can be found by unidirectional testing:

$$S_{11} = \frac{1}{E} \qquad S_{12} = -\frac{\nu}{E} \tag{3.58}$$

and by inverting eq. (3.57) the stress-strain relationship can be easily found:

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_{12} \end{Bmatrix} = \begin{bmatrix} \frac{E}{1-\nu^2} & \frac{\nu E}{1-\nu^2} & 0 \\ \frac{\nu E}{1-\nu^2} & \frac{E}{1-\nu^2} & 0 \\ 0 & 0 & \frac{E}{2(1+\nu)} (= G) \end{bmatrix} \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \gamma_{12} \end{Bmatrix} \tag{3.59}$$

Finally, for isotropic materials, only two materials need to be defined, the Young modulus E and the Poisson's ratio ν_{12}.

3.5 Notes on Damage

[1]Recent experiments and studies clearly showed that while Linear Elastic Fracture Mechanics yields accurate result for large structures, it underestimates the fracture energy when the Fracture Process Zone size is not negligible compared to the size of the structure [52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]. This can be shown using Figure 3.10.

For brittle materials, where the size of the FPZ and Plastic Zone (PZ) is small compared to the structure, LEFM works fairly well and computational model results generally agree with experiments. For ductile materials, while LEFM cannot be used due to the large plastic zone, the overall behavior is accurately captured by Elasto-Plastic Fracture Mechanics (EPFM) and nowadays it is well-known and widely used. However, for quasibrittle materials instead, the majority of the nonlinear zone undergoes microdamage and softening and therefore LEFM becomes inapplicable since FPZ is inherently neglected and doing so will yield to completely wrong results.

A clear example would be in the application of Micro Electro-Mechanical Systems

[1]From *Characterization of the Bi-Linear Cohesive Law in Quasibrittle Media via Size Effect and Dimensional Analysis* paper yet not published by author and Prof. Marco Salviato.

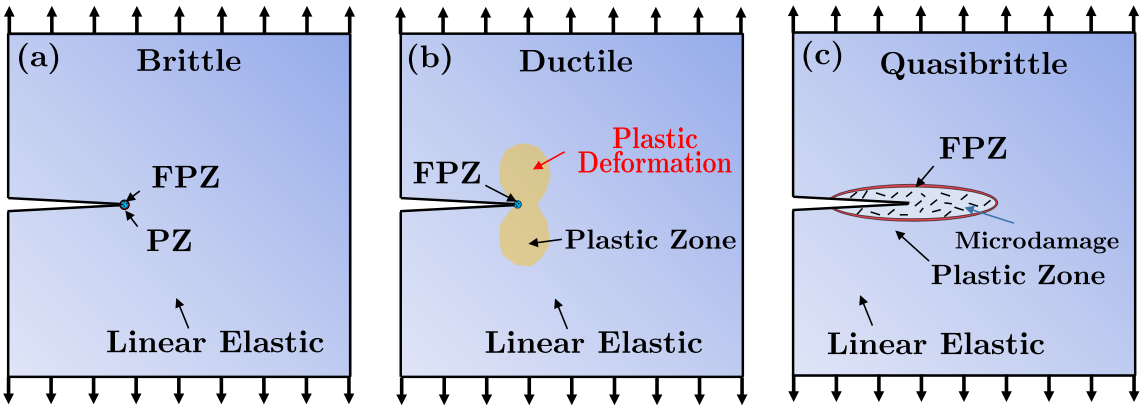


Figure 3.10: Damage in Brittle, Ductile, and Quasibrittle Materials.

(MEMS) devices. While for very large specimen sizes the FPZ might be fully developed, meaning that the shape and type of the cohesive traction separation law would not matter at all if not just the total value of the fracture energy, for smaller specimens that might not be true. The damage and fracturing mechanisms are almost completely driven by the development of the traction separation law, which can assume very different shapes. In the case of a bi-linear law for the same base material properties for example of a Carbon Fiber Reinforced Plastic (CFRP) laminate E, f_t, G_F (enough to characterize LEFM behavior) multiple configurations are possible within the variables f_k/f_t (ratio between peak strength and kinking stress) and G_f/G_F (ratio between initial fracture energy and total fracture energy). Therefore, one can only rely on size effect tests and cohesive modeling to properly characterize fracturing behavior of structures.

Another source of explanation can be drawn from Figure 3.11.

If the structural strength is plotted as a function of structure size, for very large structures the FPZ will eventually fully develop letting LEFM theory matching fairly well experimental results; however, for smaller structures this might not be true. If the FPZwhat is not fully developed, if LEFM theory is used, it might lead to wrong results. The work done by Bažant [49] provided a guideline to approach this problem, known as Size Effect Law (SEL). Testing geometrically-scaled specimens provides a very easy and effective way to explore the traction separation law at different values of opening displacements and relative

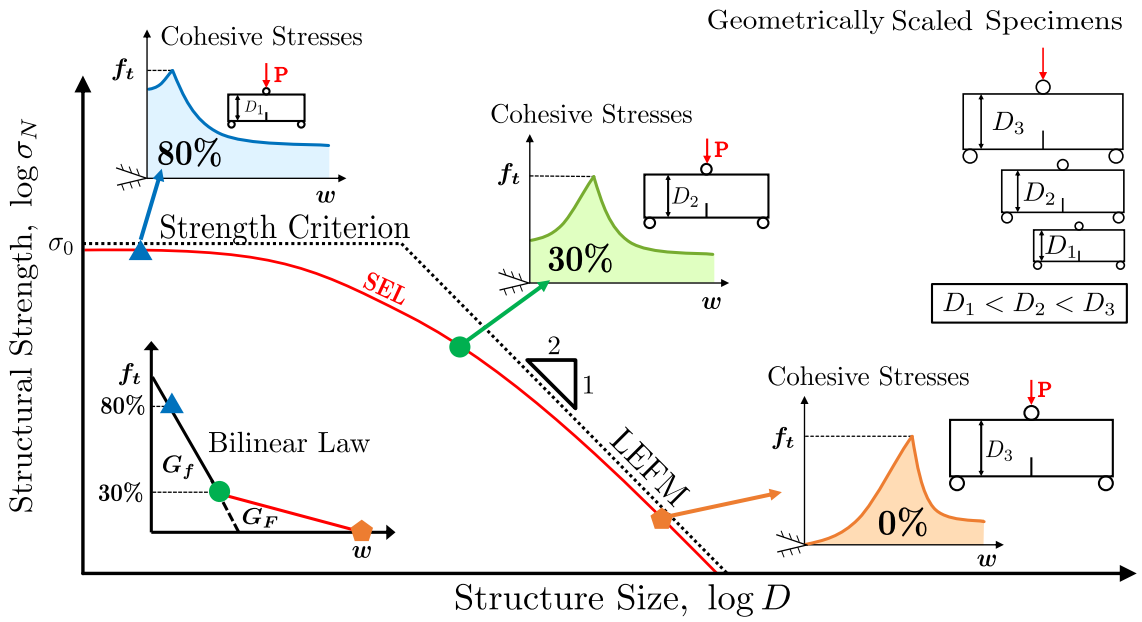


Figure 3.11: Graphical Explanation of Size Effect Law.

stresses. From Figure 3.11 above, for the smallest size specimen in the traction cohesive law, the stress in front of the crack tip might be 80% of the strength of the material; therefore, for such size the first branch of the cohesive law might be the most important factor while the second branch might not have any effect at all.

3.6 Optimization

The proposed model utilizes many optimization routines that are used to find the best parameters to use for either geometric entities, elastic material parameters, and finally nonlinear material parameters which will be the most complicated and time consume to find. MATLAB 2023a [67] is used as many optimization functions are already implemented and ready to use. Gradient Descent algorithms suited the best for the initial calibration of the geometric and elastic material parameters, particularly the interior-point, but other methods such as Sequence of Quadratic Programming and Active-Set have been used and they all yielded acceptable results. For larger simulations where the evaluation of the objective function is too time consuming and complex, a surrogate model has been used

with also acceptable results. Gradient-free algorithms have been tested, but did not provided much better results. Therefore, this section should give the reader a basic understanding on how the model works in an optimization framework.

3.6.1 Interior-Point Algorithm

Interior-point methods are one type of optimization algorithms that solve both linear and non-linear convex optimization problems. First discovered by Dikin in 1967 [68] and then extended by Karmarkar in 1984 [69], these type of algorithms were trying to tackle the problems that the Simplex algorithms were not able to solve at the time. From a geometric standpoint, interior point methods approach a solution from the interior or exterior of the feasible regions, but are never on the boundary, and the objective function is modified by a term that either increases or decreases depending on how well the constraints are satisfied.

The Matlab Interior Point Algorithm utilizes the so called *Barrier* function. The original problem is:

$$\min_x f(x), \text{ subjected to } h(x) = 0 \text{ and } g(x) \leq 0 \tag{3.60}$$

and for each $\mu > 0$, the approximate problem is:

$$\min_{x,s} f_\mu(x,s) = \min_{x,s} f(x) - \mu \sum_i \ln(s_i), \text{ subjected to } h(x) = 0 \text{ and } g(x) + s = 0 \tag{3.61}$$

where there are as many slack variables s_i as many inequalities constraints g are present. In order to keep $\ln(s_i)$ bounded, the slack variables s_i must be positive. As μ reaches zero, the minimum of f_μ should also approach the minimum of f. The logarithmic term is called indeed a *barrier* function. Equation (3.61) is in reality a sequence of equality constraint problems; therefore, they are easier to solve than the original problem in eq. (3.60).

The Matlab algorithm uses one of two other main types of algorithms at each iteration:

1. *Direct Step* in (x, s) in order to explicitly solve the Karush-Kuhn-Tucker (KKT) equations via linear approximation. This steps is often referred as well as the *Newton Step*.
2. *Conjugate Gradient Step* using a trust region.

By default, the algorithm always tries first to take the direct step and if it fail, the conjugate gradient one. At each iteration, the algorithm decreases a *cost* function:

$$f_\mu(x,s) + \nu\|(h(x), g(x)+s)\| \tag{3.62}$$

where the ν parameter can increase with the iteration number to force the solution towards a feasibility region. If an attempted step does not reduce the cost function or produces a complex, not-a-number, or infinite number, the algorithm rejects the step and tries a new one.

Direct Step

The following equation defines the direct step $(\Delta x, \Delta s)$:

$$\begin{bmatrix} H & 0 & J_h^T & J_g^T \\ 0 & S\Lambda & 0 & -S \\ J_h & 0 & I & 0 \\ J_g & -S & 0 & I \end{bmatrix} \begin{Bmatrix} \Delta x \\ \Delta s \\ -\Delta y \\ -\Delta\lambda \end{Bmatrix} = - \begin{Bmatrix} \nabla f - J_h^T y - J_g^T \lambda \\ S\lambda - \mu e \\ h \\ g + s \end{Bmatrix} \tag{3.63}$$

where all the variables are defined as:

- H is the Hessian of the Lagrangian of f_μ:

$$H = \nabla^2 f(x) + \sum_i \lambda_i \nabla^2 g_i(x) + \sum_j \lambda_j \nabla^2 h_j(x)$$

- J_g is the Jacobian of the constrain function g
- J_h is the Jacobian of the constrain function h
- S is the diagonal of s
- λ is the Lagrange Multiplier vector associated with constraints g
- Λ is the diagonal of λ
- y is the Lagrange Multiplier vector associated with constraints h
- e is the vector of ones with the same size of constraints g

Matlab performs a LDL factorization of the matrix, and if the result is not positive definite, it performs the conjugate gradient step instead.

Conjugate Gradient Step

In order to solve the approximation problem given by eq. (3.61) if the Cholesky decomposition does not work in the direct step, a conjugate gradient step is performed, similar to other conjugate gradient methods. In this step, the MATLAB algorithm adjusts both x and s keeping all the slack variables s_i positive. The goal is to minimize a quadratic approximation in a trust region subjected to linearized constraints. By denoting by R the radius of the trust region and using the same definition variables listed in the direct step approach, initially the algorithm calculates the Lagrange multipliers by solving the KKT equations in the least-square sense, subject to positive λ:

$$\nabla_x L = \nabla_x f(x) + \sum_i \lambda_i \nabla g_i(x) + \sum_j y_j \nabla h_j(x) = 0 \tag{3.64}$$

then, it takes a step $(\Delta x, \Delta s)$ to approximately solve:

$$\min_{\Delta x, \Delta s} \nabla f^T \Delta x + \frac{1}{2} \Delta x^T \nabla^2_{xx} L \Delta x + \mu e^T S^{-1} \Delta s + \frac{1}{2} \Delta s^T S^{-1} \Lambda \Delta s \tag{3.65}$$

subjected to the linearized constraints:

$$g(x) + J_g \Delta x + \Delta s = 0, \quad h(x) + J_h \Delta x = 0 \tag{3.66}$$

To solve the linearized constraints in eq. (3.66), the MATLAB algorithm tries to minimize the norm of the linearized constraints within the trust region with radius R. Then the Conjugate Gradient approximation in eq. (3.65) is solved with the constraints matching the residual from eq. (3.66) and checking that the solution lies within R and that the slack variables s_i are strictly positive. More equations and derivations can be found in [70, 71, 72].

3.6.2 Surrogate Modeling

Surrogate optimization is a technique used to optimize expensive or computationally intensive objective functions by building a surrogate model which resembles the original objective function where calculating optimal points is much simpler. The algorithm alternates stages

where first some points are ran with the original objective function and subsequently, using those results, the model optimizes the surrogate and check backs with the original optimization function if the results were good or need to be iterated more. This is of particular interest in the development, testing, and calibration of DM4C and FastDM4C because optimizing all the material properties in different tests is computationally prohibitive, particularly if gradients are to be found as well. For example, to capture the longitudinal strength in the fiber direction, a simulation needs to be ran up to failure and the size has to be large enough to capture both the elastic behavior but possibly even matrix and interfacial behavior. This simulation might require sometime even hours, making this approach unfeasible. By using the surrogate model as basis to change design variables accordingly to the output behavior, this process can be rendered feasible.

The surrogate optimization algorithm generally alternates between two phases:

1. Construction of the Surrogate
2. Search for the Minimum

Construction of the Surrogate

To construct the surrogate, quasi-random points are chosen within the given bounds to be ran with the full model. Once evaluated, the algorithm uses Radial Basis Function (RBF) to build the surrogate model. Many types of RBF functions are possible; however, it has been shown that a cubic RBF with linear tail works the best for the problems similarly posed by the calibration and optimization of FastDM4C [73, 48].

Search for the Minimum

Once the model is approximated using the surrogate, the search for the minimum phase consists of find the input parameters that minimize the surrogate model, not the real model. There are many different techniques that can be used in this phase, but being the surrogate model a simpler model, even first order algorithms can work pretty well [74]. At this point the algorithm generates a set of candidate solutions for evaluations which are then evaluated in the real model. The results are then incorporated again into the surrogate model and these

steps repeated until final convergence. If the surrogate model is not capable of improving the solution, it is usually reset and restarted following again all the steps outlined in section 3.6.2.

3.6.3 Parallel Quadratic Regression

Another method developed and used to calibrate the properties of the FastDM4C was quadratic regression [75] in a parallel setting. Simulations can be ran in parallel and the results can be fitted to a parabola where the minimum can be found easily knowing the coefficients of the parabola. If the best optimal approximated value by the regression is found outside of the parabola bounds, then the best result is simply switched to the point whose objective function was minimized. Then the bounds are diminished by a percentage and the steps repeated until final convergence occurs.

3.7 Weibull Distribution for Strength

Engineering structures are often designed in such a way the failure of probability is in the order of one in a million ($< 10^{-6}$). To satisfy these requirements, often empirical tools and methodologies have been used [76]. Perfectly ductile and brittle materials have a Cumulative Distribution Function (CDF) of random strength historically known. However, this approach does not fit well with quasibrittle materials because the FPZ's size is comparable to the size of the structure itself [77, 22]. Moreover, as the structure size increases, the strength CDF starts changing from a Gaussian distribution to a Weibull distribution. With the aforementioned reasons, to inherently capture all the damage mechanisms of composites, it is necessary to implement a Weibull distribution of the strength.

The Probability Density Function of a Weibullian random strength variable σ is given by the following equation:

$$\text{PDF} \Longrightarrow p^{\text{PDF}}(\sigma) = \frac{\alpha}{\sigma_0}\left(\frac{\sigma}{\sigma_0}\right)^{\alpha-1} \exp\left(-\left(\frac{\sigma}{\sigma_0}\right)^{\alpha}\right) \tag{3.67}$$

where $\sigma_0 > 0$ is the shape parameter and $\alpha > 0$ the scale parameter. As the names imply, the shape parameter affects simply the shape of the distribution while the scale parameter affects its stretching or shrinking. This is often referred as the 2-parameter

Weibull Distribution curve. In literature it is often possible to find a 3-parameter Weibull curve with an added location parameter which affects the shifting of the curve; however, it has been profoundly discarded by the academic community when used in probabilistic fracture mechanics [22, 78, 79].

Another curve already mentioned, is the CDF, which is the integral of the PDF. It is given by the following equation:

$$\text{CDF} \Longrightarrow p^{\text{CDF}}(\sigma) = 1 - \exp\left(-\left(\frac{\sigma}{\sigma_0}\right)^{\alpha}\right) \tag{3.68}$$

where the parameters $\sigma_0 > 0$ and $\alpha > 0$ are the same for eq. (3.67). This curve describes the probability of having an equal or less value of σ. Therefore, as expected, the curve ranges from 0 to 1. The inverse of the CDF eq. (3.68) is called the Quantile Function:

$$\sigma(p) = \sigma_0 \sqrt[\alpha]{-\ln(1-p)} \tag{3.69}$$

where given a certain probability $0 < p < 1$, it gives back the value of the maximum σ allowable.

3.7.1 Weibull Parameters Calibration

The Weibull strength calibration is out of the scope of this thesis; however, extensive work has already been done and the parameters σ_0 and α will be used by the available academic sources [80, 79]. To give the reader an idea of the order of the magnitude of the values used, the shape parameter α is found in the literature to be around $56 \sim 60$ with a variance of $5 \sim 6\%$.

However, if starting from the average value of the material strength $\bar{\sigma}$ and its variance $\text{Var}(\sigma)$, it is possible to also calculate and find σ_0 and α. From the definition of average and variance of a Weibull parameter [81]:

$$\begin{aligned} \bar{\sigma} &= \sigma_0 \, \Gamma\left(1 + \frac{1}{\alpha}\right) \\ \text{Var}(\sigma) &= \sigma_0^2 \left[\Gamma\left(1 + \frac{2}{\alpha}\right) - \Gamma^2\left(1 + \frac{1}{\alpha}\right)\right] \end{aligned} \tag{3.70}$$

Equation (3.70) is a system of two equations for which σ_0 and α can be found numerically given an average material strength $\bar{\sigma}$ and variance $\mathrm{Var}(\sigma)$. The gamma in eq. (3.70) refers to the gamma function [82] which is defined as:

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt \tag{3.71}$$

3.7.2 Mesh Independent Weibull Distribution

Of particular importance, the strength distribution used in any numerical model must be mesh independent. For example, if a beam is discretized into two elements and for each element a strength is found via eqs. (3.68) to (3.70), the mean strength should be the same for a beam discretized with order of magnitudes higher elements. By simply using eqs. (3.68) to (3.70), this is not captured, as if more values of σ are used, the higher probability there is to get values from the lower spectrum of the distribution which are the ones driving the failure behavior of the structure. This can be tackled by adding an extra parameter for which the ratio of the volume is taken into consideration. In the proposed model where beam elements with constant cross sections are used, the parameter is proportional simply to the length. This extra parameter is defined as l_0, therefore the characteristic length applied to the strength value σ. This parameter shows as an extra ratio for which each element's length must be divided for. The CDF eq. (3.68) becomes:

$$\mathrm{CDF} \Longrightarrow p^{\mathrm{CDF}}(\sigma) = 1 - \exp\left(-\left(\frac{l}{l_0}\right)\left(\frac{\sigma}{\sigma_0}\right)^{\alpha}\right) \tag{3.72}$$

where l is the length of the element in the mesh and l_0 the fixed parameter. Accordingly, the average strength eq. (3.70) needs to be updated as:

$$\bar{\sigma} = \sigma_0 \left(\frac{l}{l_0}\right)^{-\frac{1}{\alpha}} \Gamma\left(1 + \frac{1}{\alpha}\right) \tag{3.73}$$

and the Quantile Function (Inverse of the CDF) as:

$$\sigma(p) = \sigma_0 \sqrt[\alpha]{\frac{\ln(1-p)}{l/l_0}} \tag{3.74}$$

Using eqs. (3.72) to (3.74) it is possible to fully implement a mesh independent Weibullian strength distribution.

3.7.3 Latin Hypercube Sampling

Given a certain number of discretized beam elements in a model, each element needs to be assigned a value of strength based on the theoretical framework laid out in section 3.7.2. However, a finite number of segments might not be evenly distributed throughout the spectrum, and this is critical as the lower tail of the CDF is the one that will yield fracture and failure. By using a statistical sampling technique called Latin Hypercube Sampling (LHS), it is possible to reduce the risk of overlooking critical regions of the parameter space, particularly the lower and upper tails. Without this sampling technique, particularly using a fewer number of elements, it can lead to inaccurate or biased results as the number of elements do not guarantee the overall capture of the distribution.

This is shown graphically in fig. 3.12. If ten elements were used to discretize a beam with ten random probability values selected between 0 and 1, there might be chance that not enough elements will have either very low or very high tail values of the probability distribution. If however, the space is divided in 10 divisions, representing each element, and the probability is picked within each of those subdivisions, the whole beam will be sure to have values ranging the whole distribution spectrum. While the assignment is done in order in the computational code, after creating an array of values of strength, the values of strength are shuffled and then assigned to each element in the beam.

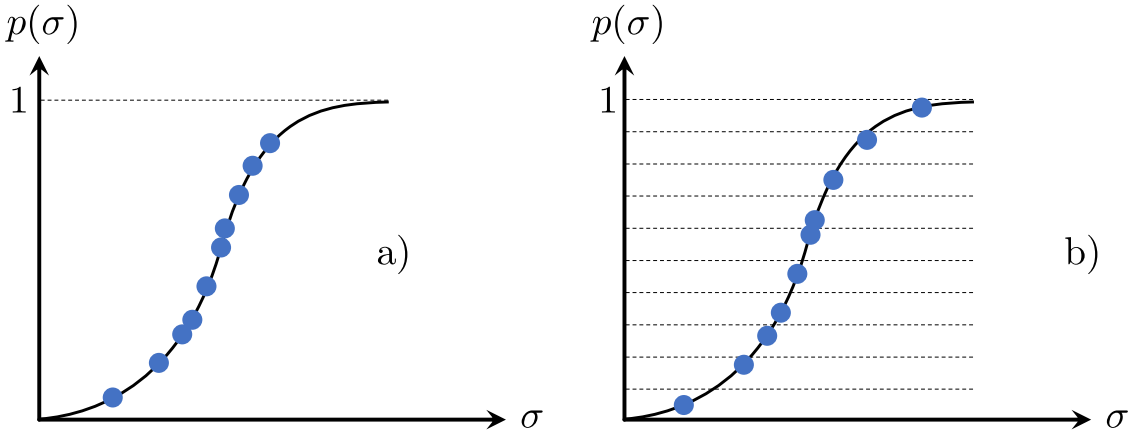


Figure 3.12: Quick representation of the difference between sampling ten different values of strength by using the Latin Hypercube Sampling Method (b) or not (a).

Chapter 4

IMPLEMENTATION WITHIN THE FINITE ELEMENT FRAMEWORK

In this chapter, all the steps needed to create the finite element model of any structure will be given. The goal is to represent with discrete elements a layered composite structure of any kind, where each ply is characterized by oriented repetitions of a basic RUC. Each ply is created independently from the other plies and once all the plies are created a constrained Delaunay triangulation in 3D is performed between two different plies. Of this triangulation which yields as a result solid tetrahedrons, only the edges are retained, and of all the edges, only the ones that have both nodes on different plies are kept. In this way the final model will consist of only 1D edge elements, where the elements representing the fibers will be implemented with the Discrete Fiber Model while the in-plane matrix and the interface matrix will be implemented with the Discrete Matrix Model. A dual optimization framework is also developed and used to calibrate and test the model. To match a single material system, both gradient descent and surrogate models are utilized, but to be able to match multiple material system, a database of simulation results are compiled by (i) running parallel simulations onto a High Performance Computing (HPC) cluster and (ii) the results saved into an efficient Structured Query Language (SQL) database where (iii) a Neural Network (NN) is trained to provide the user particular model parameters to better describe a macroscopic material system.

4.1 Programming Language and Software

The whole model was developed both using Matlab and C++ [83, 67]. Matlab is a high-level programming language specifically designed to perform numerical computation with easy tools also in data analysis and visualization. On the other hand, C++ is a low-level general purpose programming language aimed for efficiency and high performance. Moreover, Matlab is an interpreted language, which means debugging and on-the-fly changes

are usually allowed. This is of particular interest during development and testing of a computational model because the provided libraries are capable of display, plotting, analysis and comparing changes rather quickly and painlessly. Once the logic and workflow are figured out, the code can be easily changed into C++ which allows order of magnitude faster performance at the cost of harder debugging and required memory management. Recent upgrades to Matlab also allowed easy implementation of Object Oriented Programming (OOP) handle classes. These are classes which under the hood make the use of pointers (reference in memory instead of local memory copies), similarly to the already established C++, and in doing so, complicated models can be built upon the use of smaller classes.

While most of the generation algorithms were developed using Matlab, the software MARS was used as the computational solver [84]. MARS is being developed by Dr. Daniele Pelessone at ES3 in San Diego, and the code is written in modern C++. It is based on an explicit central difference algorithm and it includes most of the capabilities and versatility of other commercially available finite element codes. Such solver was chosen mainly for ease of integration. Dr. Pelessone granted almost unlimited access to the author to develop and integrate new codes. Other commercially available software such as Abaqus, Nastran, and Ansys allow users to write specific subroutines, but hiding most of the logic of the solver which cannot be access by any means.

4.1.1 Remarks on the Code's Object Oriented Capabilities

Numerous benefits make OOP a popular choice for software development. The following are some significant benefits of Object Oriented Programming:

1. Modularity: OOP encourages modularity by dividing complicated issues into more manageable chunks known as objects. Each object contains data and associated behaviors in a single unit, simplifying the maintenance and modification of the code.
2. Reusability: Code reuse is made possible by OOP thanks to the inheritance notion. It is unnecessary to rewrite shared code when new classes can inherit attributes and methods from older ones thanks to inheritance. As a result, development time is cut down, code duplication is minimized, and maintainability is improved.

3. Encapsulation: Encapsulation is a fundamental OOP paradigm that shields data and methods from outside intervention by grouping them together into an object. It improves security and maintainability by enabling information hiding and offering a clear user interface for dealing with objects.
4. Abstraction: OOP encourages abstraction by enabling the creation of abstract classes and interfaces by developers. By concentrating on important traits and behaviors and obscuring unimportant aspects, abstraction enables programmers to create complex systems by using streamlined models. It improves the organization, readability, and simplicity of the code.
5. Polymorphism: The ability to consider objects of many classes as members of a single superclass is known as polymorphism. This adaptability permits the use of replaceable parts and permits the writing of more generic and adaptable code. Due to the fact that new classes may be added without altering existing code, polymorphism makes code expansion and maintenance easier.
6. Maintainability: By offering a clear and modular structure, OOP increases code maintainability. Since objects are self-contained components, it is simpler to comprehend and adjust particular sections of the code without changing the behavior of the entire system. This makes testing, refactoring, and debugging easier, which makes the product easier to manage and more scalable.
7. Code Organization: OOP promotes a methodical approach to code organization. Code may be naturally organized using objects and classes, which makes it simpler to explore and find certain functionality. This enhances readability and comprehension of the code, particularly in bigger projects.
8. Code Extensibility: OOP makes it simple to extend code through polymorphism and inheritance. By deriving from pre-existing classes, new classes can be constructed by adding or replacing particular behaviors. This feature makes it easier to add new features or modify current code, which lowers the likelihood of making mistakes.

The capacity of Object Oriented Programming to encourage code modularity, reuse, maintainability, collaboration, and scalability, leading to more durable and adaptable software systems, is the main benefit of this approach which has been picked for the development

and deployment of the Discrete Model for Composites (DM4C) the Fast Discrete Model for Composites (FastDM4C).

4.2 Implementation Workflow

To better explain all the steps of the computational implementation, fig. 4.1 is used to aid the reader. The first step is create the Representative Unit Cells. As explained in the previous section this can be done for any size and shape, but for composites the shape is recommended square to avoid directional bias and the size should be similar to the average distance between splitting cracks, anywhere between values of 0.05 mm and 0.4 mm. Once the shape and size of the RUC are fixed, an array is created and a wanted geometry is overlaid on top of it. A trimming algorithm is used to cut the array matching the same size of the geometry and the elements that are cut are removed from the model. This step can be done in parallel as each ply is independent from each other. Once all the plies are constructed, they need to be connected to each other and this is the most critical step. Previous iteration of the model explored the possibility of using both cohesive interaction technique and cohesive elements; however, they were proven to be numerically unstable in most of the cases. It was chosen to perform a node-to-node connection using the same modeling used for the Discrete Matrix Model and FacetElement as described in section 3.3. One the plies are connected to each other, both elastic and fracture calibration are performed in order to make sure the constituents mechanical behavior mimics exactly the behavior of a composite material system. Finally, predictive simulations can be performed of much larger structures, both for example laboratory-sized specimens and real engineered structures such as joints and frames.

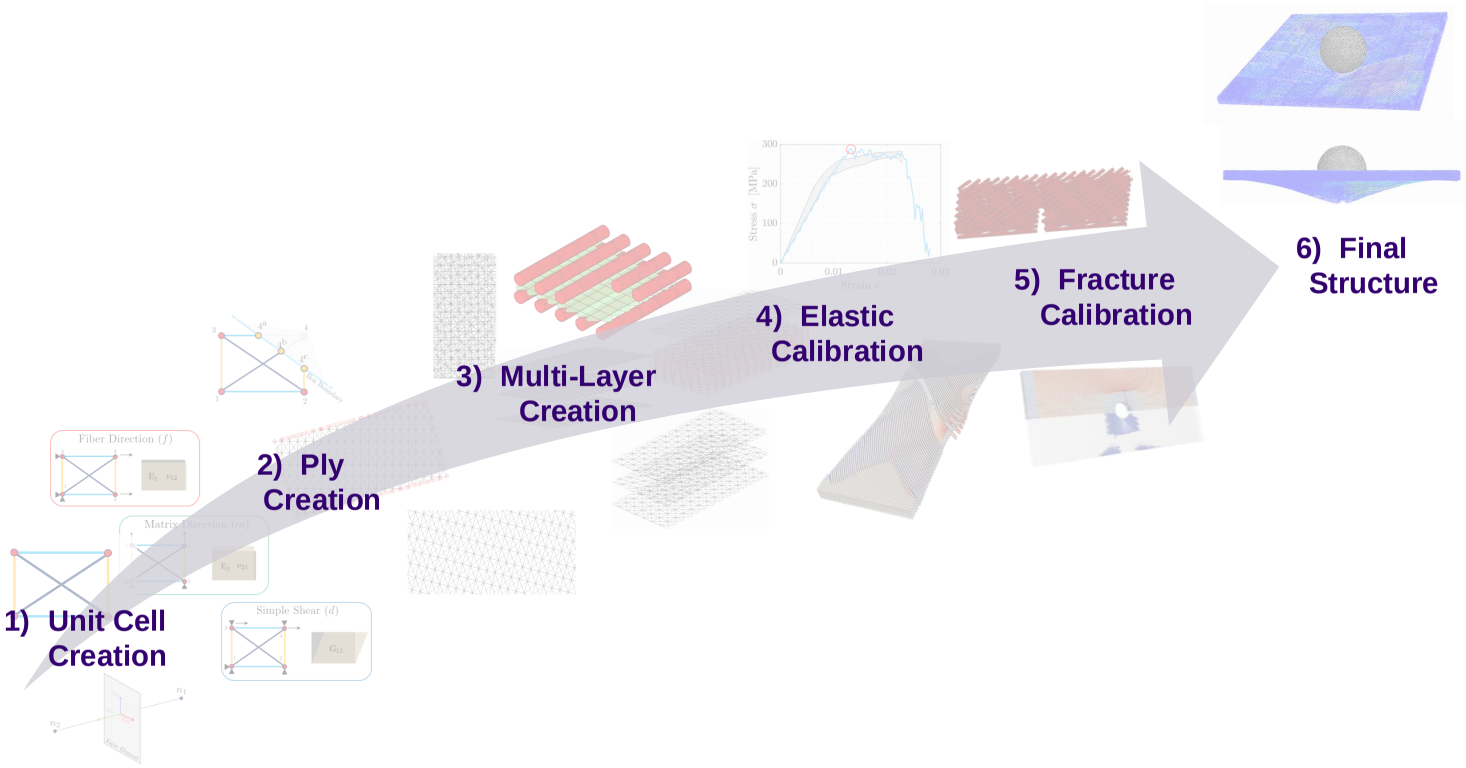


Figure 4.1: Implementation Workflow of the FastDM4C.

4.3 Unit Cell Creation

The unit cell was modeled to mimic as much as possible the orthotropic behavior of a composite laminate. This can be achieved by using three different sets of 1D elements. In reference to Figure 4.2, the top and bottom horizontal elements (in light blue) can be referred as *Fiber* elements, the left and right vertical elements (in yellow) as *Matrix* elements and finally the diagonal elements as *Diagonal* elements (in dark purple). In this way, even if each element does not behave like a transversely isotropic material, the interaction between the three sets provides a behaviour more similar to orthotropic materials. In comparison to real composite laminates this makes sense, there are fibers which are much stiffer, orthogonal to the fiber elements there is resin, which usually has a much lower stiffness, and the interface between the two is a mix of both behaviors. The size of the unit cell can be arbitrary, both in length and height, and initial test showed that many possible configurations can be achieved by changing both geometric and material parameters, even if, preliminary elastic calibration showed that usually square unit cell work much better. However, the physical size of the RUC should be close the average size of the distance between splitting cracks commonly found in composites as this will guarantee the capture of physical cracking phenomena. But again, the other advantage of the proposed model is the capability to arbitraraily change the shape and size of the RUC to adapt to a multitude of different problems. Firstly, the nodes array is creating following the steps outlined in Algorithm 1 and then the three element members of the RUC are created following Algorithm 2. A small flexibility was added where there is an actual distinction between the positive and negative orientation of the diagonal members; however, in all aspects they are considered the same.

4.4 Ply Trimming Algorithm

Once the elastic calibration of the unit cell is finished, the unit cells are repeated into an array large enough to be able to fit a ply of any given size (Figure 4.3a) and orientation. In this way, information of each unit cell's members is retained and properly stored.

Once the box representing the ply is overlapped, a trimming algorithm (Figure 4.3b) with pseudo-code provided in Algorithm 3 is used to cut all the elements which have intersection

ALGORITHM 1: Creation of Node Array.

Input : L_{UC} – Length of the Unit Cell
H_{UC} – Height of the Unit Cell
L_{ply} – Length of the Ply
H_{ply} – Height of the Ply
Output: N – 2D Node Matrix

x_{rep} = `floor` $((L_{\mathrm{ply}}/L_{\mathrm{UC}}) \cdot 1.10)$ `/* UC in x-dir */`
y_{rep} = `floor` $((H_{\mathrm{ply}}/H_{\mathrm{UC}}) \cdot 1.10)$ `/* UC in y-dir */`
N = `zeros` $((x_{\mathrm{rep}}+1)\cdot(y_{\mathrm{rep}}+1), 2)$ `/* Initialization of the Node Matrix */`
for i $= 1 : x_{\mathrm{rep}} + 1$
  for j $= 1 : y_{\mathrm{rep}} + 1$
    counter = counter +1
    $N(\mathrm{counter}, :) = [(\mathrm{i}-1)\cdot L, (\mathrm{j}-1)\cdot H]$
  end
end

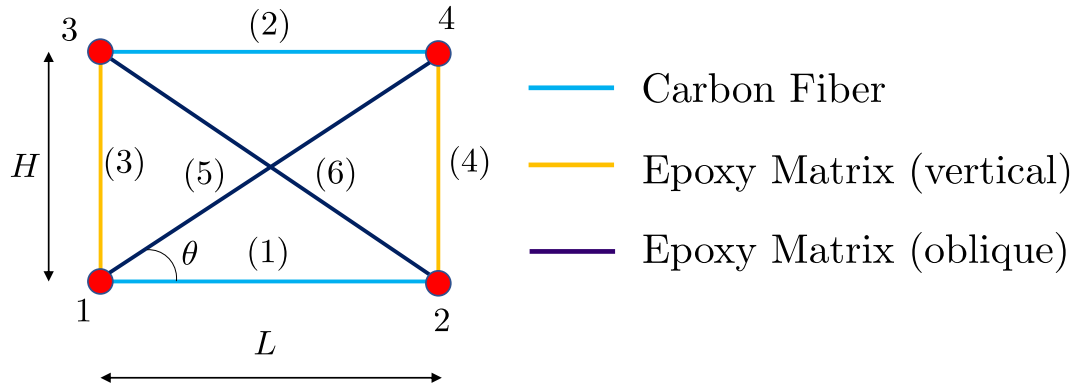
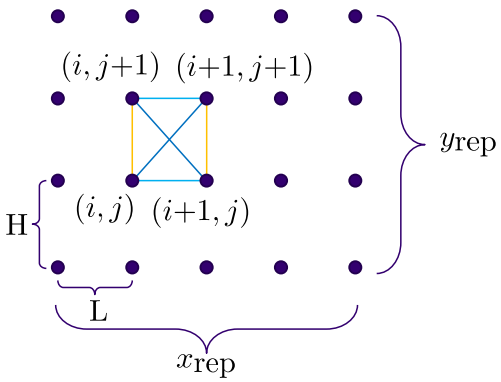


Figure 4.2: Unit Cell Description.

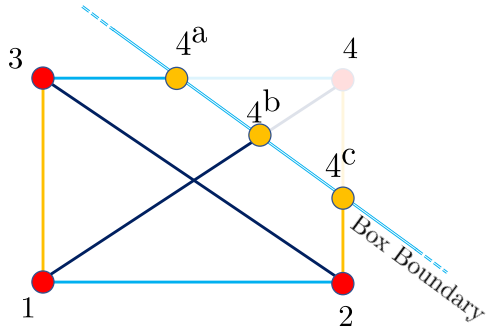
ALGORITHM 2: Creation of Element Connectivity.

Input : x_{rep} – X-repetitions UC
y_{rep} – Y-repetitions UC
Output: E_0 – Fiber Elements
E_{90} – Matrix Elements
E_{45}^{pos} – Positive Diagonal Elements
E_{45}^{neg} – Negative Diagonal Elements

E_0 = **zeros** $((x_{\text{rep}}) \cdot (y_{\text{rep}}), 3)$ /* Initialization of Fiber Elements */
E_{90} = **zeros** $((x_{\text{rep}} - 1) \cdot (y_{\text{rep}}), 3)$ /* Initialization of Matrix Elements */
E_{45}^{pos} = **zeros** $((x_{\text{rep}}) \cdot (y_{\text{rep}} - 1), 3)$ /* Initialization of Positive Diagonal Elements */
E_{45}^{neg} = **zeros** $((x_{\text{rep}}) \cdot (y_{\text{rep}} - 1), 3)$ /* Initialization of Negative Diagonal Elements */
counter = 0 /* First temporary iteration counter */
for i = $1 : x_{\text{rep}}$
  for j = $1 : y_{\text{rep}} + 1$
    counter = counter +1
    E_0(counter, :) = [counter, counter + y_{rep} + 1, 1]
  end
end
counter = 0
counterb = 0 /* Second temporary iteration counter */
for i = $1 : x_{\text{rep}} + 1$
  for j = $1 : y_{\text{rep}}$
    counter = counter +1
    counterb = counterb +1
    E_{90}(counterb, :) = [counter, counter + 1, 2]
  end
  counter = counter +1
end
counter = 0
counterb = 0
for i = $1 : x_{\text{rep}}$
  for j = $1 : y_{\text{rep}}$
    counter = counter +1
    counterb = counterb +1
    E_{45}^{pos}(counterb, :) = [counter, counter + y_{rep} + 2, 3]
  end
  counter = counter +1
end
counter = 0
counterb = 0
for i = $1 : x_{\text{rep}}$
  for j = $1 : y_{\text{rep}}$
    counter = counter +1
    counterb = counterb +1
    E_{45}^{neg}(counterb, :) = [counter + y_{rep} + 1, counter + 1, 4]
  end
  counter = counter +1
end



(a) Unit Cell Array Creation Description.



(b) Description of the Trimming Algorithm. Once intersections are found, node matrices and element connectivities are accordingly updated.

Figure 4.3: Quick description of how the unit cell array is generated and how the trimming algorithm works.

point with the box. This algorithm is robust enough to handle any poly geometry, such as dogbone, and any defect geometry, such as holes, cracks, and notches. The code performs a binary search of which nodes are inside, outside, or on the boundary and then checks with Boolean operations which elements have nodes on either those three different sets. This resulted in an extremely fast search, as no mathematical operations were used. If using built-in MATLAB functions to check for each element intersection, it would have taken order of magnitude higher search time. Once a set of elements that need to be cut is found, the trimming operation is initiated and new boundary nodes are added to the node matrices and the element connectivities properly updated to reflect the changes at the boundary. This is shown in Figure 4.4. In Figure 4.4a an example of a rectangular ply (4mm by 2mm) with associated unit cell size (0.2mm by 0.3mm) is overlapped to the array. In Figure 4.4b the trimming algorithm finds the intersection nodes (highlighted in red) and updates the node and element connectivity matrices accordingly. In Figure 4.4c all the elements and nodes outside of the domain of the ply are removed. At this point, in Figure 4.4d, the ply is rotated back so that the unit cells will be incline at the wanted orientation, but the geometry will be align with the x-axis. The algorithm is capable of

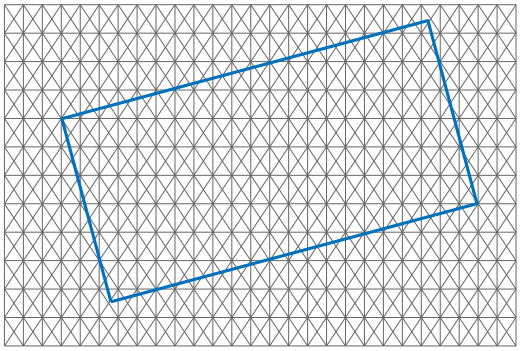
handling also complicated geometries and defects are shown in Figure 4.5 where a dogbone geometry is shown together with a plate with a hole.

This steps is also performed in parallel as the information needed to create one ply is independent to another. This speeds up considerably the time needed to generate the geometry, as sometimes laminate can have a stacking sequence of 40 or even more plies. MATLAB offers a very simple parallelization technique called Single Process Multiple Data (SPMD) where the same algorithm can be applied as the name implies to different input data [67].

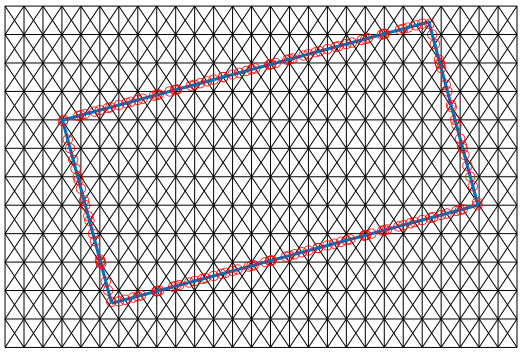
ALGORITHM 3: Major Steps of the Trimming Algorithm

Input : N – Node Matrix
E – Element Connectivity Matrix
N_b – Node Matrix of the boundary polygon
E_b – Element Connectivity Matrix of the boundary polygon
Output: N – Updated Node Matrix
E – Updated Element Connectivity Matrix

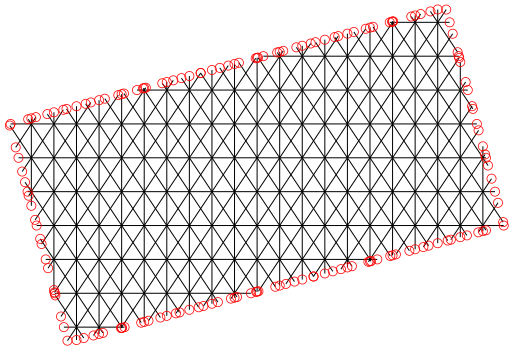
1 N_{inside} = Find nodes inside boundary polygon `/* Using Binary Search */`
2 E_{inside} = Find elements which have both nodes in N_{inside} `/* Using Binary Search */`
3 $E_{\text{to cut}}$ = Find elements which have either node in N_{inside} `/* Either the first or the second */`
4 **for** each element in $E_{\text{to cut}}$
5 Find intersection node between E_b and $E^{(i)}$
6 Add new node to N_{new}
7 Update connectivity of $E^{(i)}_{\text{to cut}}$ `/* by substituting index of outside node with the new one */`
8 **end**
9 $N = N_{\text{inside}} + N_{\text{new}}$ `/* Concatenate Matrices */`
10 $E = E_{\text{inside}} + E^{(i)}_{\text{to cut}}$ `/* Concatenate Matrices and properly reindexing */`



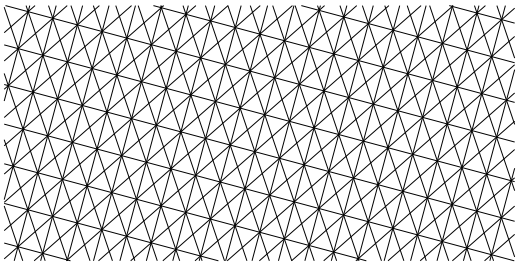
(a) Step 1: Creation of large enough array of unit cells and overlapped ply geometry.



(b) Step 2: Trimming Algorithm is used to find all intersection nodes.

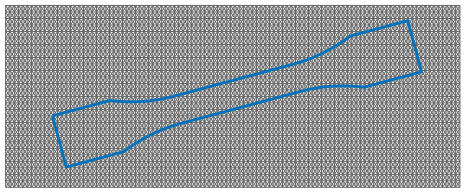


(c) Step 3: The nodes outside of domain of the ply are removed.

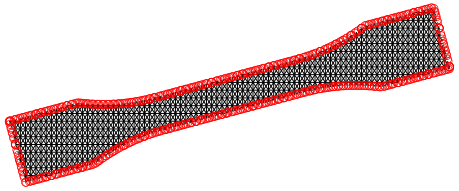


(d) Step 4: The ply is rotated accordingly.

Figure 4.4: Example of creation of a ply oriented at $\theta = 15^\circ$ with a width of 4 mm and height of 2 mm and unit cell size $L = 0.2$ mm, $H = 0.3$ mm.

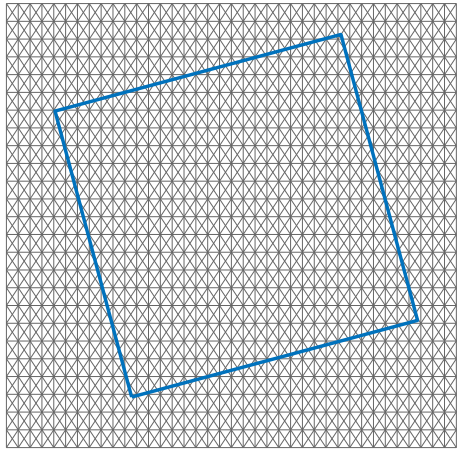


(a) Step 1: Array Unit Cells and Overlap of Dogbone Geometry.

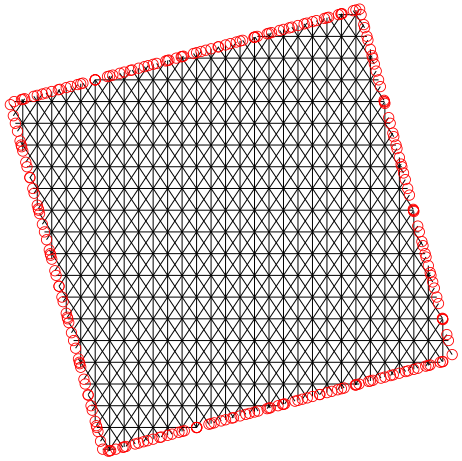


(b) Step 2: Trimming of the Dogbone Specimen.

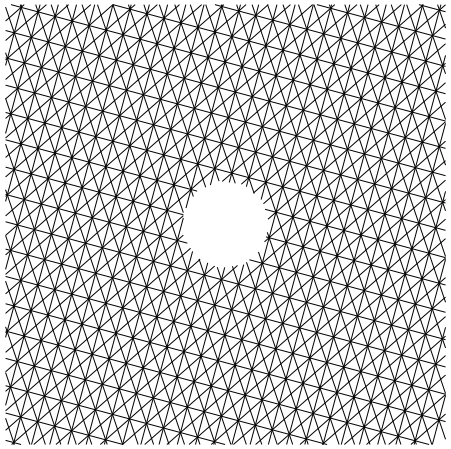
(c) Step 3: Final Rotation of the Dogbone Specimen.



(d) Step 1: Array Unit Cells and Overlap of a Square Geometry.



(e) Step 2: Trimming of the Square Geometry.



(f) Step 3: Final Rotation of the Square Specimen and Second Trimming Operation for the Hole Defect.

Figure 4.5: Example of the trimming algorithm applied to a dogbone geometry (figs. 4.5a to 4.5c) and square geometry with a hole defect (figs. 4.5d to 4.5f)

4.5 Stacking Algorithm for Multi-Layer Creation

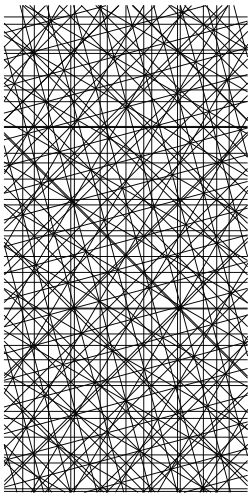
Once plies of a laminate are created independently in a 2D environment, they are all added into a 3D environment with proper z spacing equivalent to the ply thickness given as an input of the laminate model. An example with exaggerated z spacing is shown in Figure 4.6. Moreover, under a FEM point of view, the node matrices of each ply are merged and properly re-indexed, as well as the element connectivity matrix. In this case, the user has access to each single ply individually and also to the laminate as whole. This is done because the user can export the model to any third party software (Abaqus, Nastran, Patran, LS-Dyna, MARS) and import either single plies, a set of them, or all of them. Abaqus, for example, implements assemblies with instances, and node sets of an instance cannot be use as node sets of another and viceversa; therefore, the ability to have access to node and element connectivity matrices individually and all together is extremely convenient.

4.6 Ply Connectivity Methods

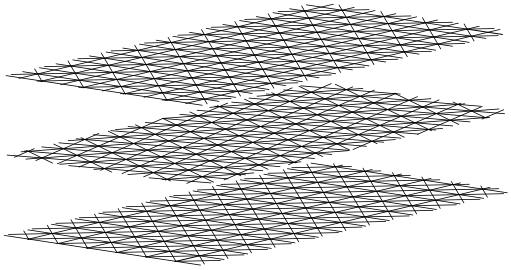
When the laminate stack is defined, plies need to be connected together. During the development of the FastDM4C model, multiple algorithms have been implemented and tested, but only three have shown possible applicability. As shown in Figure 4.7, these methods are based on (i) Cohesive Interaction, (ii) Cohesive Elements, and (iii) Nodal Connection. While all of them works up to a certain level, the last method was chosen to be further explored and used as it deemed to be the most efficient and fastest to run while still maintaining very high accuracy compared to the other two methods which instead showed cases of numerical instability.

4.6.1 Cohesive Interaction Method

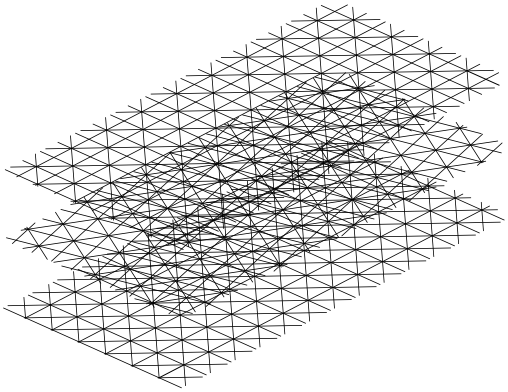
qUsing this methodology, the connection between each ply is created by first creating non-structural shell elements using the four nodes of each unit cell and splitting the border elements either with quadrilateral or triangular elements. Since they play a non structural role, the meshing definition becomes almost irrelevant. For inner non-trimmed unit cells, simply the four nodes are used to create a singular quadrilateral shell, while for the elements



(a) Top View of the simple laminate.



(b) 3D view of the laminate.



(c) Another 3D view of the laminate.

Figure 4.6: Example of a stacking laminate [0/30/45] with similar geometric properties of Figure 4.4. The only difference is that the unit cell size is square and not rectangular. In this case a 0° ply looks almost the same of a 45° ply. Note that the distance between plies is exaggerated to better show the stacking method.

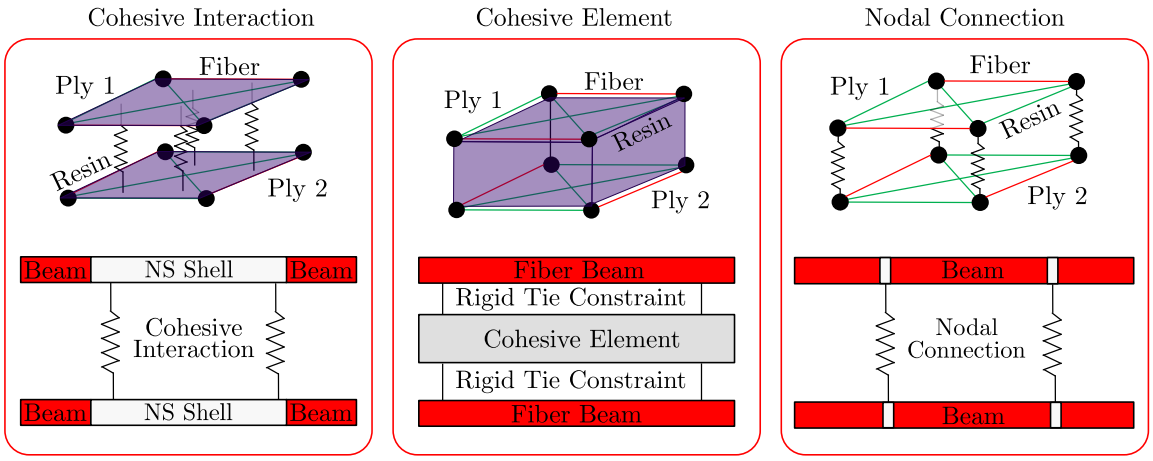


Figure 4.7: Three different proposed methodologies to connect different plies to each other

that are cut during the trimming process, a slightly different approach is used as shown in Figure 4.8. Depending on the intersection of the boundary with the unit cell, different cases are possible, where in some quadrilateral element can be created (highlighted in red) and for other where triangular elements (highlighted in blue) must be created in order to maintain the boundary line. Once the non-structural shell elements were created, a cohesive interaction is created between each surface of two adjacent plies. The reason why the shells needed to be created is that nodal-based surface do not work very well with cohesive interaction and even advanced software such as Abaqus did not implement cohesive interaction with node-based surfaces. Also, by non-structural is referred to the fact that they do not carry any load and force (unitary material properties for example) and are there just to be able to create an element-based surface, which on the other hand worked pretty well. As one might notice, the cohesive interaction is created between two surfaces that are not touching each other. While for some software this might be an issue, for software such Abaqus this was not a problem if the simulation was run using the implicit solver (Abaqus/Standard), while for the explicit solver the advanced contact initialization and controls are not implemented yet. They were very recently implemented in the 2020 version; however, the computational cost and complexity to create the non-structural shells and run the cohesive interaction was too high, and it was chosen a faster and more efficient

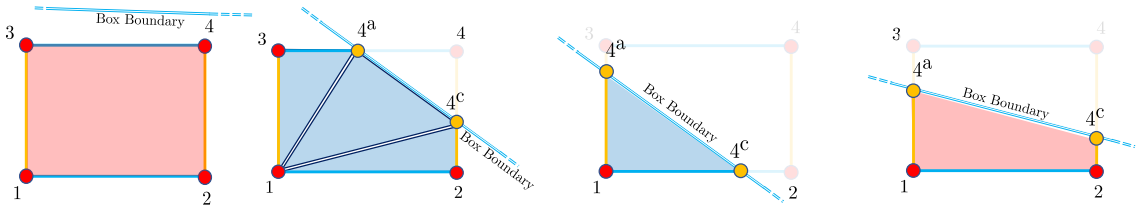


Figure 4.8: Different cases of how the non-structural shell elements are created using the perimeter nodes after the trimming algorithm. Red elements result in the creation of quadrilateral shells, while blue elements result in the creation of triangular shells. A mix of both is required to properly describe the boundary edges.

method. A simple de-bonding test was carried out to test the feasibility of the model and, again, together the high running time, the model did show numerical instabilities and difficulty to converge.

4.6.2 Cohesive Element Method

Once the Cohesive Interaction Model was deemed too computational expensive to run (mostly due the contact interaction), a similar model that uses cohesive elements and tie constraints (more stable than the contact used by the cohesive interaction) was implemented as shown again in Figure 4.7. Cohesive elements must either be hexaedra or pentahedra elements; therefore, making the geometry looking much more regular and cleaner than any other method. The cohesive elements were created starting from a 2D regular mesh, either triangular or quadrilateral, and finally extruded in 3D by the lenght of the ply thickness from the laminate input. If the specimen was rectangular or square, the mesh was obviously rectangular and created from scratch; however, if the geometry was complicated (dogbone geometry or any sort of defect), the mesh was triangular and created using a software suite written for Matlab by Engwirda [85] and converted to quadrilateral mesh using the steps described in [86]. An example of meshing test scenario is given in Figure 4.10 where the generation of any geometry and defect was extremely efficient and fast. Another reason the meshing algorithm was kept in Matlab was to avoid issues using different software dealing with importing and exporting codes, which would have added a considerable over-head

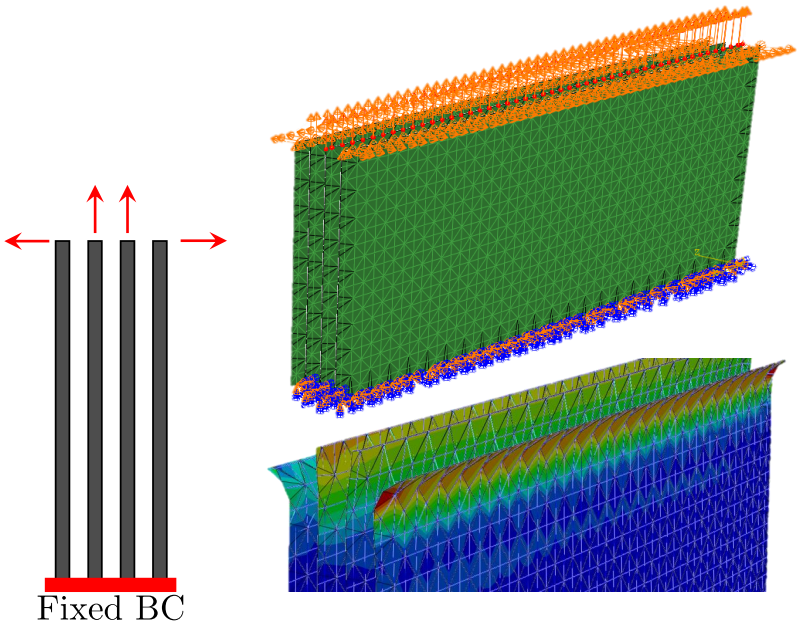
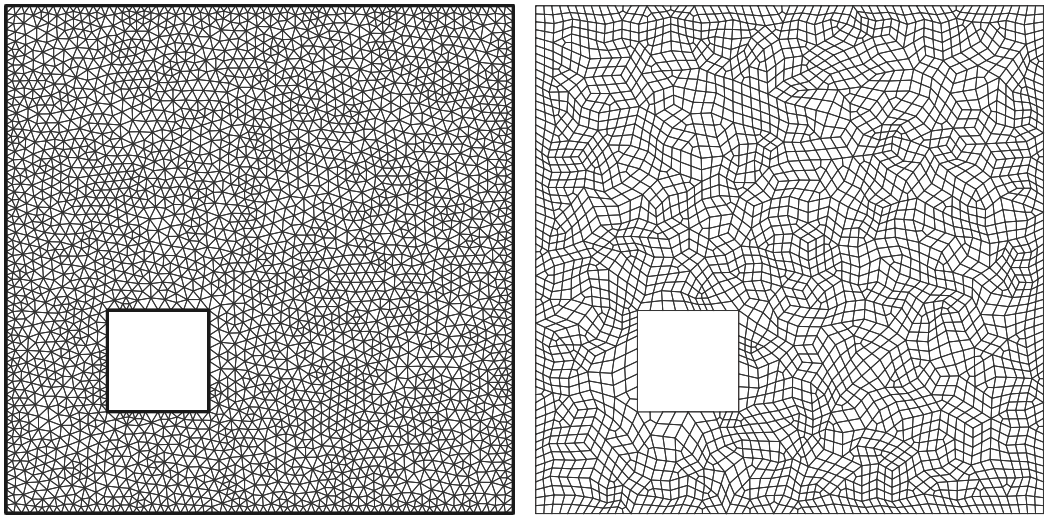


Figure 4.9: Quick De-Bonding Test to exploit problems and issues for the Cohesive Interaction Method. The model ran successfully, but the computational cost was too high to justify its use and future implementation.

time and it would have rendered optimization routine calls extremely slow. Another example of triangular mesh, then extruded, to have 3D 6-node cohesive elements is shown in Figure 4.11.

To test the full capabilities of the cohesive element model, a quasi-isotropic laminate was tested. Multiple configurations of $[45/90/\text{-}45/0]_s$ with dimensions 40×40 mm made out of T700 subjected to uniaxial load were simulated to check the accuracy of the elastic calibration and overall behavior of the model. Since cohesive elements do not carry forces in-plane, the elastic behaviour optimized for the unit cell was on-point. The initial fracture material property were adequately guessed to run the first cases. In Figure 4.12 are shown failure damages for both beam and cohesive elements. Before trying to fully optimize the fracture properties to match results from experiments, the qualitatively behaviour was very interesting. The final fracturing behavior was consistent with the experimental results; however, it is possible to notice many random regions away from the crack also completely failed, and the cohesive elements had too extensive damage considering it was a

quasi-isotropic laminate. After testing many different other simulations and even cross-ply specimens, similar results were obtained. The most educated guess was that the tie constraint between the element-based surface of the beam elements and top or bottom layer of the cohesive elements was unstable and during fracturing produced fictitious peak of stresses that caused certain beam and cohesive elements to reach their strength and either start failing or completely failing right away. Another test that was carried out to check the issue of random failure was the use of solid elastic elements instead of the cohesive elements. The elastic elements were modeled orthotropically, with unitary properties in the plane and similar elastic properties in the out of plane direction to the cohesive elements. During the fracturing of the beam elements, spikes of stresses were noticed in the solid elastic elements as well. As a consequence, it is still believed that the issue resides in the explicit formulation of the contacts in the tie constraints. This formulation is a black-box in software such as Abaqus; therefore, the Cohesive Element Model approach was abandoned. However, the feasibility of the model was proved, and if more access to the surface-to-surface contact interaction would be possible, improvements could be made. Finally, the model did ran faster compared to similar models ran using the Cohesive Interaction Model. The example shown in Figure 4.12 had 100000 degrees of freedom (DOFs) and ran on 12 threads on a local machine in about 40 minutes with standard mass scaling and strain rate stratagems, but far away from the wanted computational time goal. Therefore, taking advantage of a well known method, the ply connectivity was finally created using another approach.



(a) Example of triangular mesh creation (including defects) using [85].

(b) Example of Triangular to Quadrilateral mesh conversion using [86].

Figure 4.10: Triangular and Quadrilateral Mesh Creation for Cohesive Elements.

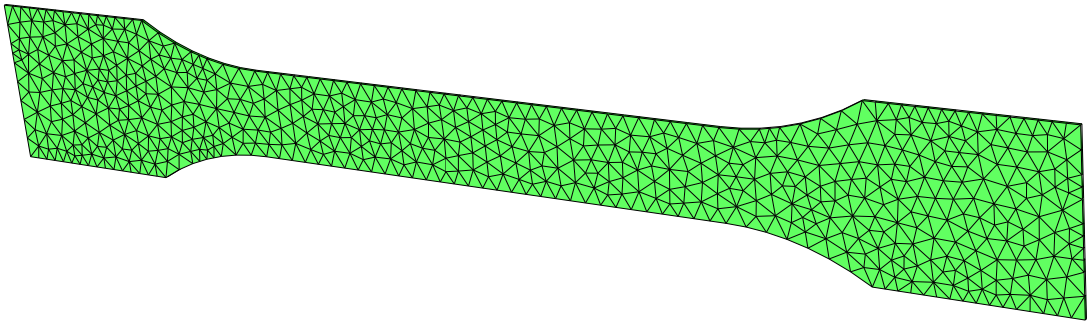
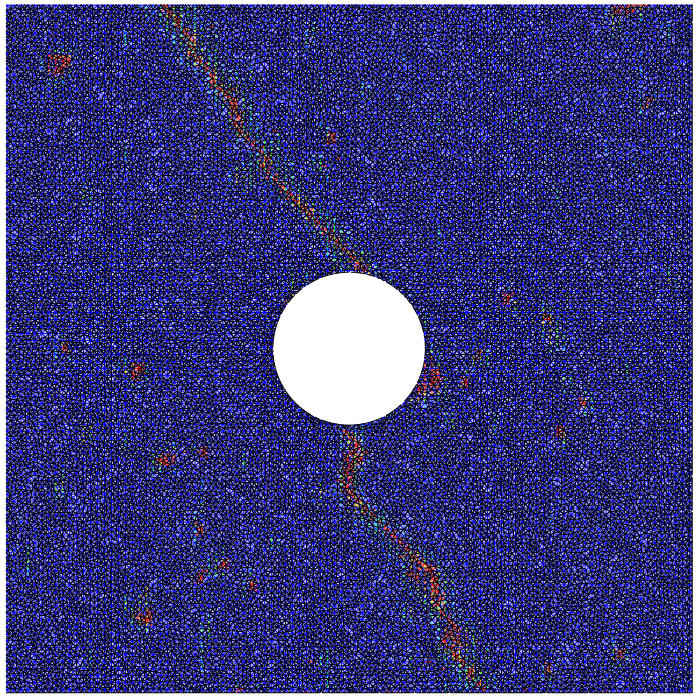
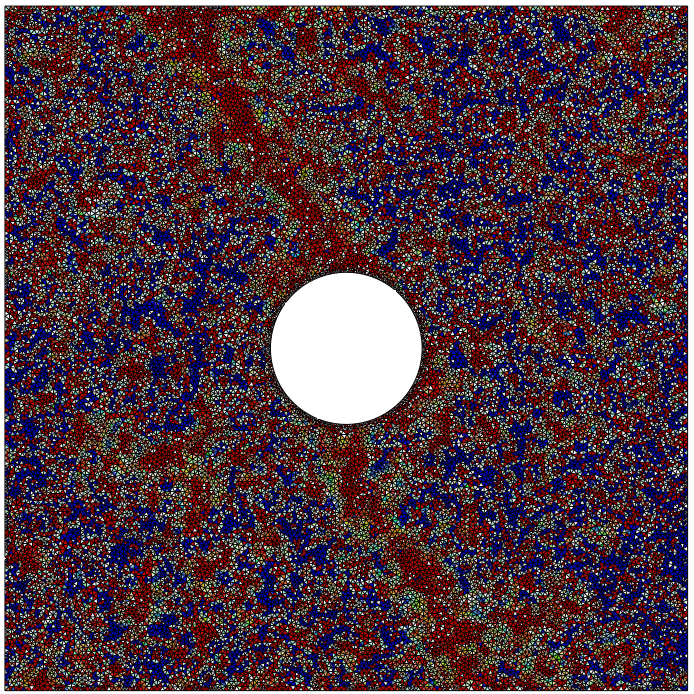


Figure 4.11: Example of 6-node cohesive elements mesh for a dogbone geometry. The planar triangular mesh is extruded to generate the tridimensional mesh.



(a) Damage visualization of fiber, matrix, and diagonal elements for the quasi-isotropic laminate (red color means completely failed). Qualitatively, the damaging behavior is similar to the one produced by experiments; however, the multiple localized damage regions away from the crack are non-representational of real fracturing behavior.



(b) Damage visualization of the cohesive elements. The extensive overall damage of the cohesive elements is also non-representational of real experiments. The tie constraints between the beams and the cohesive elements turned out to create sporadic stress peaks, causing both cohesive element and beam elements to fail.

Figure 4.12: $[45/90/\text{-}45/0]_s$ with dimensions 40×40 mm made out of T700 subjected to uniaxial load. The qualitative response

4.6.3 Nodal Connection Method

The last method used was actually one of the first method to be brainstormed. The initial reason why it was put aside was the fact that nodes on different plies do not align on the *z-axis* (normal plane of the laminate); therefore, most of the elements would be skew and the question of how connecting nodes should be done brought up too many random variables and difficulty in generating complex geometries. However, given the numerical instabilities and issues brought by the two previously mentioned methods, this method was further debugged and modified accordingly to work properly.

Multiple iterations of this method were performed, some with their own advantages and disadvantages; nonetheless, the final iteration chosen that was robust enough to be able to generate a complete variety of interlaminar connections was the following. Using a similar approach already explored in the development of the Lattice Discrete Particle Model [31], for every pair of plies, a constrained Delaunay Triangulation was performed. This triangulation yield a fully 3D tetrahedral mesh. However, of these tetrahedral elements, only the edges that are not co-planar to each plies pair was retained. This is shown in fig. 4.16 for the stacking example of fig. 4.6 and in fig. 4.17 for an example of a specimen with a center crack.

Once these edge elements are generated, their mid-points are taken (which by geometry construction lie exactly in the mid-plane between the two plies) and a further Voronoi Tessellation is performed on those sets of nodes. This is clearly explained in fig. 4.13. The advantages of this method is that first, the edge elements connect perfectly one ply to another, and secondly, the Voronoi Tessellation also leads to the creation of Facet Elements which can be used exactly with the formulation of the Discrete Matrix Model described in section 3.3. In this way, there is even a better flexibility of capturing delamination.

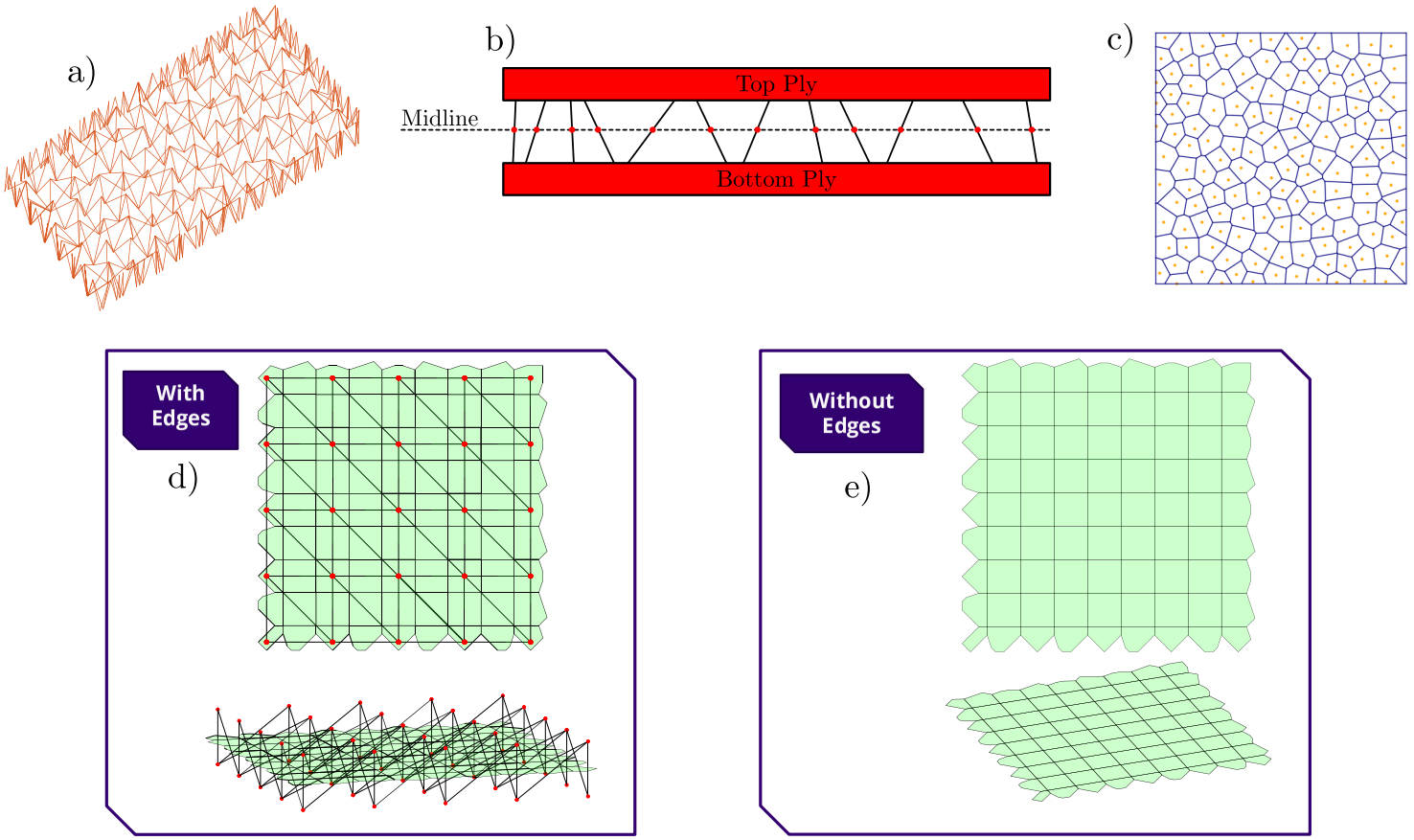


Figure 4.13: (a) Generation of inter-ply connection elements using the edges of a Constrained Delaunay Triangulation from the same stack example of Figure 4.6. (b-c) From these elements the mid-points are selected. (d-e) Using a Voronoi Tessellation, the plies are connected and the interlaminar facets created.

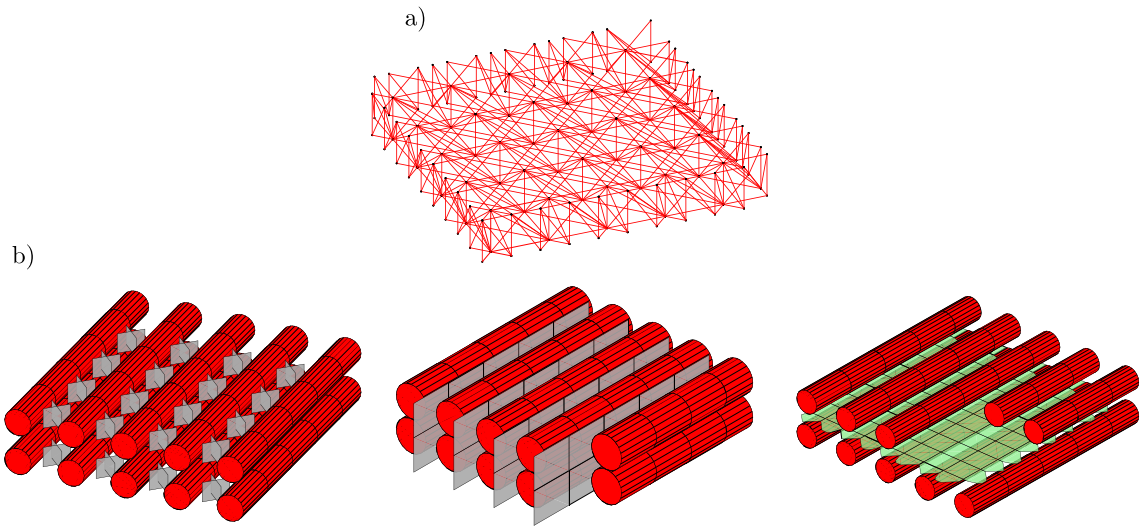


Figure 4.14: (a) Example of all the edges generated by the FastDM4C model, both in-plane and out-of-plane elements. (b) Example of rendered pictures where the fiber elements are rendered as beams and all the other elements using the Facet Element formulation. The grey facets correspond to the matrix and diagonal components and the green facets correspond to the interlaminar facets created using the Voronoi Tessellation.

4.7 Elastic Calibration

Once the model is fully generated, it is firstly calibrated for elastic properties as they are much easier and faster to run. The calibration methodology is split into two methods. The first one uses classical optimization approaches as mentioned in section 3.6. The second approach is a bit more flexible as it uses Machine Learning and Neural Networks to calibrate the model parameters to the macroscopic parameters. This is done via the use of massively parallelized HPC codes written in C++ using the software MARS [84] where also an SQLite database [87] is utilized to save all the simulation results. To fill results for the database a Task Scheduler was implemented. The MARS software is written in C++ and with many object-oriented capabilities. This allows the coupling of the solver with Message Passing Interface (MPI) techniques to run multiple simulations with different material parameters at the same time. The setup is quite straightforward. The user decides how many parameters to sweep and for each parameter how many values to be tested then, how many types of

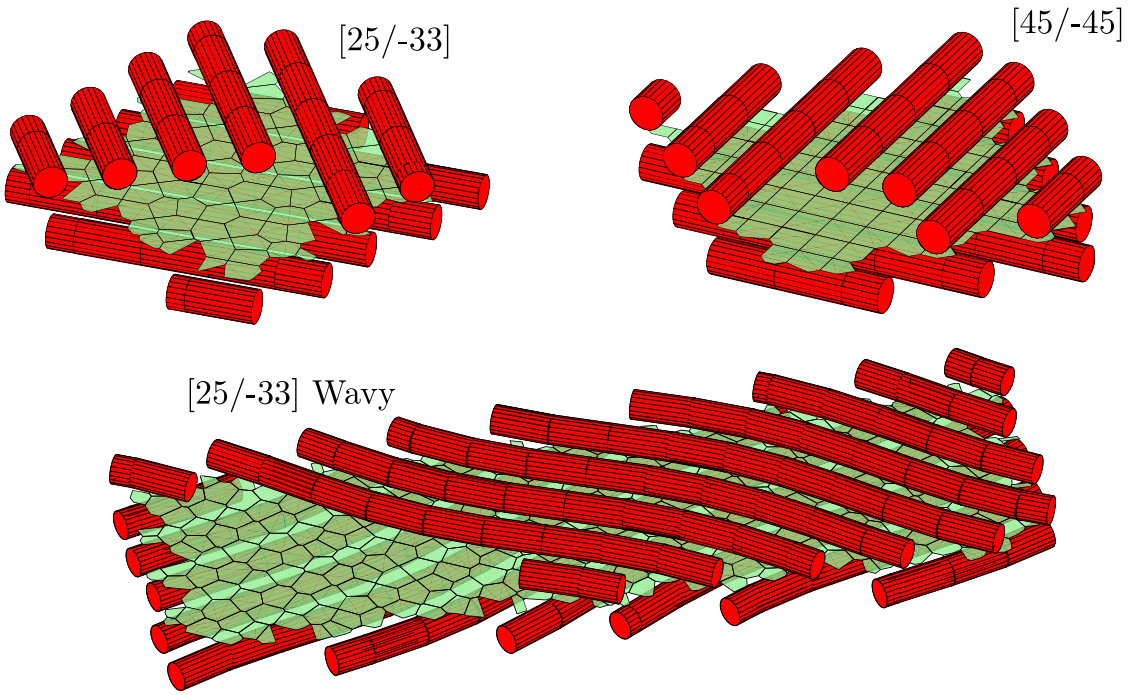


Figure 4.15: In this figure the robustness of the Voronoi Tessellation for interlaminar facets is presented. If the plies are regular or cross-ply, the tessellation always yields regular square elements. However, if the plies have special angles or present waviness, the algorithm generates well-defined polygon facets.

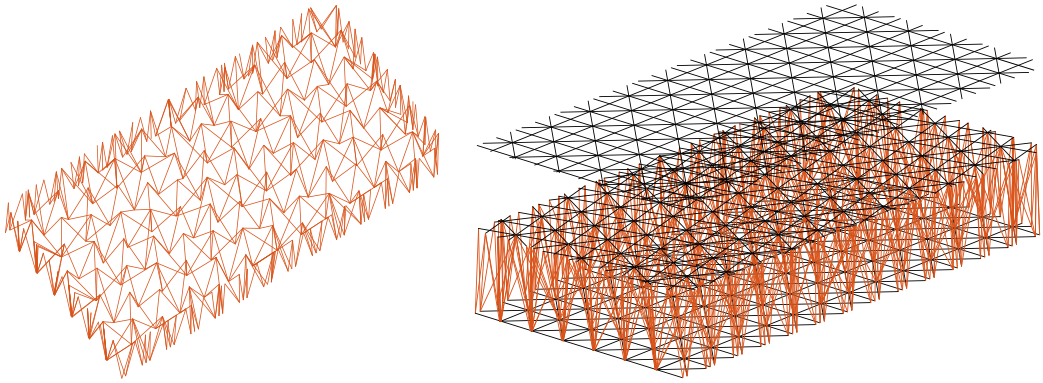


Figure 4.16: Generation of inter-ply connection elements using the edges of a Constrained Delaunay Triangulation from the same stack example of Figure 4.6. The *z-axis* is exaggerated for visualization purposes; therefore the elements look more distorted than they really are.

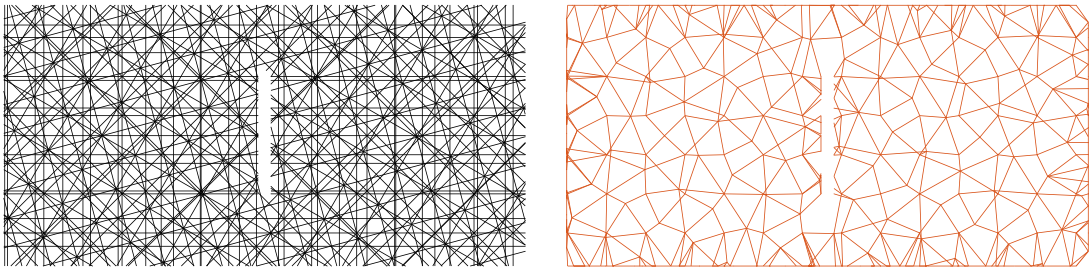


Figure 4.17: Top view generation of inter-ply connection elements with the addition of a center crack. The algorithm is very robust even for any defect geometry.

simulations to run and finally the software creates a list with all the simulations to run from any combination. The MPI code will have a zero-th rank which act as the scheduler, dispatching parameter and gathering results from all the other ranks. The flexibility of this system is that the HPC clusters usually take care of providing the Central Processing Unit (CPU) and Random Access Memory (RAM) automatically from each node and these nodes can be homogeneous or even heterogeneous without altering the structure of the code.

The SQLite relational database stores data into multiple tables which have relations with each other. Each table is made of multiple rows and columns which need to have unique indices for each row. The relationships between tables can be of many types, for example one-to-one, one-to-many, many-to-one or many-to-many. With all these configurations, tables can store redundant data extremely efficiently as behind the scenes, only unique indices are used for calculations and queries. This is of great importance for sweep simulations because a single value of a parameter can be used many times while sweeping other parameters; therefore, that number does not need to be copied multiple times as a user would do using Excel. Multiple types of databases are available open-source, but for its flexibility, portability, easy server-less deployments and in-memory capabilities, SQLite was chosen. SQLite developers provide also C++ API, allowing the easy coupling with MARS as only static libraries are used. Another advantageous aspect, perhaps the most important, is the capability to store heterogeneous data. This means that different types of results, for example both from linear elastic simulations and fracture simulations, can be stored in the same way and both queried at the same time. Moreover, different users

can add more results to the database without deranging and disorganizing already existing data. The used relationship schema of the database implemented in DM4C and in MARS is provided in fig. 4.18

If the machine learning approach is chosen instead of the standard optimization techniques, the large database is constructed with hundreds if not thousands of results from simulations sweeping different parameters, so that a Neural Network can be used to match simulation parameters to different material systems. The advantage of this system is that once the network is trained, switching to a different material system is much easier that re-perform another optimization/minimization study for a single material system. For calibrations that require few parameters, a shallow NN of a few hidden layers is more than enough as adding more might cause over-training and issues in general. However, if more parameters are added (due to fracturing and plastic behaviors) a deeper neural network might be used. In preliminary studies, elastic calibration of the FastDM4C model was accomplished by running 2592 simulations and the subsequent calibration of the NN yielded average errors of around 1%.

4.8 Fracture Calibration

All the methods described in section 4.7 hold true also for the calibration of the fracture properties, with only one main issue, the running time of such simulations. As shown in fig. 4.19, it is fairly fast for an elastic simulation to converge. A Moving Average (MA) approach was implemented in MARS so that when the MA is below a certain pre-determined threshold, the simulation exits and the results are provided back to the user. However, for fracturing behavior, the peak force and time at peak force are used to determine if the solver needs to exit the simulation. This is done by either percentage drop, or percentage of time reached after peak force. All these methods together allow the user to easily monitor the simulations, but not only. If the database with thousands of simulations needs to be generated, the user would not be able to know before hand how long to run each simulation for; therefore, having an automatic system which allows the solver itself to decide when to stop the simulation is remarkable. Moreover, these techniques are not present in any commercially available software as they do not allow this degree of flexibility as they are

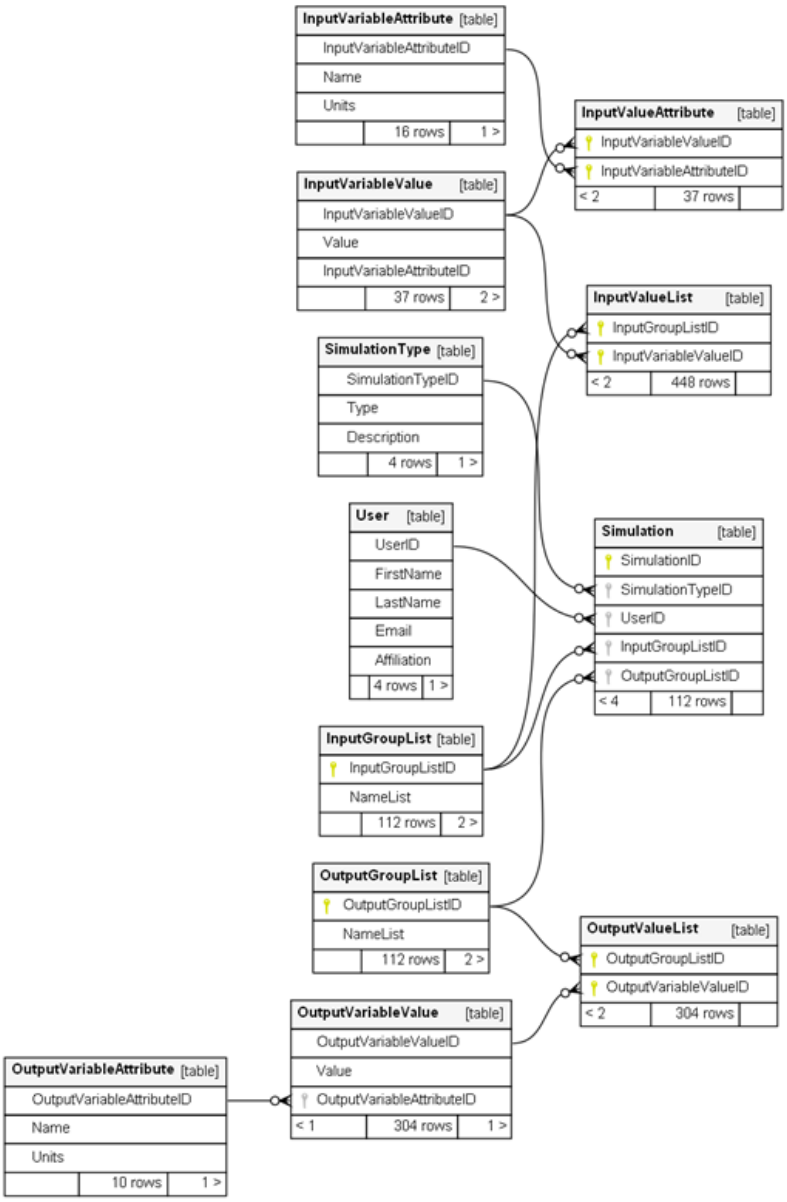


Figure 4.18: The relationship schema between each different table used to build the database to store all the simulation results is shown. It allows to very efficiently store data coming from any type of simulation, user, solver, and different input and output parameters. Moreover, retrieving data is as easy, as SQL queries can be pre-formatted and saved beforehand.

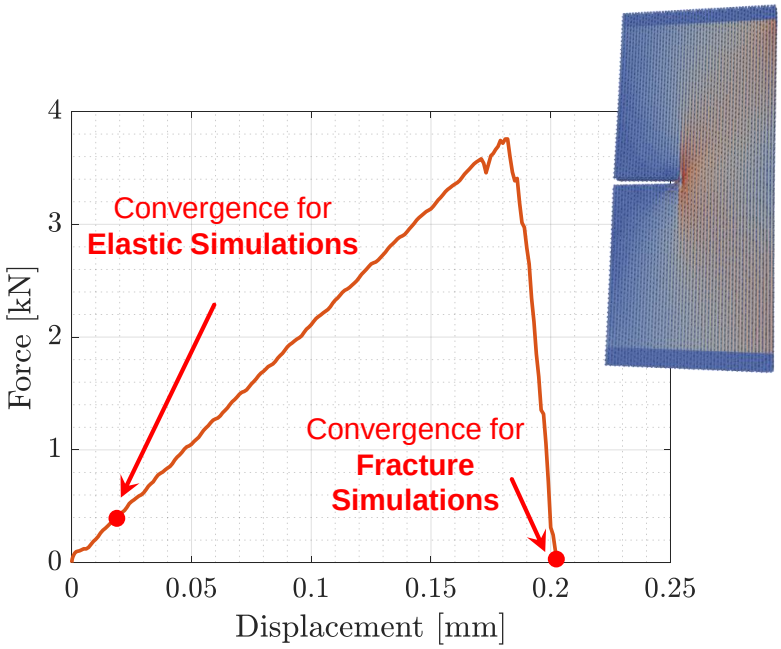


Figure 4.19: The methods used to calibrate fracture properties are exactly the same of the ones used to calibrate elastic properties. The only difference is in the running times of these simulations.

provided mostly as black-boxes tools where users cannot make changes nor have access to the base code.

Chapter 5

RESULTS

In this chapter, the calibration results are provided together with morphological predictive analyses of similar notched composite laminates. The FastDM4C was also calibrated for a variety of material systems using the techniques laid out in section 4.7; however, the focus of the computational simulations was aimed towards the characterization of the IM7/977-3 material system. This material system is the preferred material choice for the Air Force Research Laboratory which is also sponsoring this work. The experimental properties of this material system is provided by Clay [88].

5.1 Calibration Results

Thanks to all the tools and techniques developed in conjunction with ES3 with the solver MARS, the calibration for both elastic and fracture strength properties are shown in table 5.1. The three test specimens chosen are shown in fig. 5.1. To calibrate fiber-dominated parameters, a $[0]_4$ laminate was used. To calibrate matrix-dominated parameters, a $[90]_4$ and $[+45/\text{-}45]_s$ laminates were used.

The elastic properties were found via the use of the MA technique while the fracture

Table 5.1: Elastic and Strength Calibration Results for IM7/977-3 Material System

Property	**Units**	**Target**	**Achieved**
E_1	GPa	164.3	164.6
E_2	GPa	8.85	8.70
G_{12}	GPa	4.94	4.98
ν_{12}		0.318	0.301
X_T	MPa	2905	2904
X_C	MPa	1569	1465
Y_T	MPa	78.9	77.7
Y_C	MPa	248	250

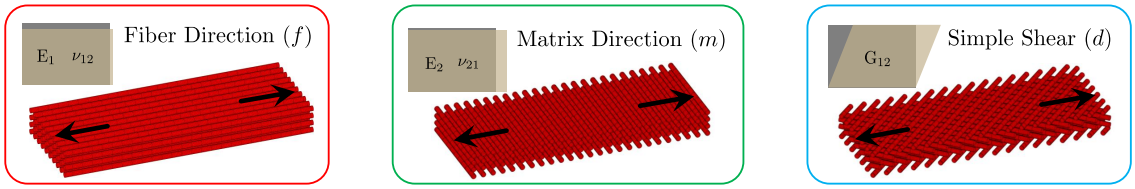


Figure 5.1: Test specimens used for the calibration process. To calibrate fiber-dominated parameters, a $[0]_4$ laminate was used. To calibrate matrix-dominated parameters, a $[90]_4$ and $[+45/\text{-}45]_s$ laminates were used.

properties (strengths in both longitudinal and transverse, tensions and compression) were found by running the test specimen simulations to failure. The accuracy of the calibration is outstanding. The unit cell was chosen to be squared with a length and width of 0.2 mm which is similar to the average distance of the splitting cracks in fracturing of composites.

The curves of the standard macroscopic parameters E_1, E_2, and ν_{12} are not provided as they are linear to failure and table 5.1 already summarizes the results. However, the non-linearity of the $[+45/\text{-}45]_s$ laminate to find G_{12} is worth mentioning. Figure 5.2 shows how well the non-linear is captured by FastDM4C and how similar the mode of fracture is with real experiments. Done this, the model can now be used for larger size specimens and structures.

5.2 Notes on Compression Calibration

Particular attention was paid to the calibration of the compressive properties as they are quite tricky to calibrate and replicate. To exactly measure both the transverse, but particular the longitudinal strength properties, the specimen cannot fail due to buckling, but instead due to the shear failure. In reality, fibers in composite laminates are not perfectly oriented to the ply, and many studies show that these fibers form bands which exhibit waviness [89]. Computationally, that is implemented by applying a sinusoidal transformation to the straight fibers so that they also do exhibit waviness, both in-plane and out-of-plane with values taken from the literature [89]. Once this was implemented, the calibration results also showed that the compression specimens failed due to shear and not due to buckling as

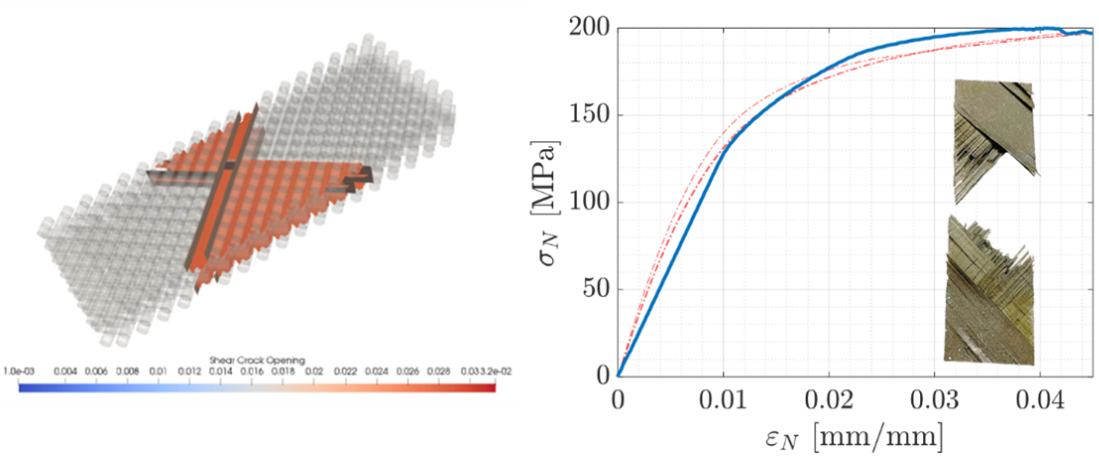


Figure 5.2: Once the model is fully calibrated for the IM7/977-3 material system, a shear laminate is tested and the response curves compared to the experimental ones provided by Clay [88].

it should expected from a real experimental test.

5.3 Laboratory Size Simulation Results

Once all the macroscopic moduli and strengths are calibrated almost perfectly to the material system, larger simulations can be ran to gather both qualitatively and quantitatively results. While real tests' stress-strain curves were not available as they currently a work-in-progress, the results obtained were compared to similar specimens found in the literature.

5.3.1 Single Edge Notched Tension$[0/90]_{2s}$ Laminate

A simulation of a cross-ply Single Edge Notch Tension (SENT) laminate of IM7/977-3 of the dimensions of 20 mm in width and 44.5 mm with an edge crack of 8 mm ($\alpha = 0.4$) was performed and its morphological results compared to literature [91]. A detailed picture of damage at initiation is shown in fig. 5.4. The results can be compared to fig. 5.5 where damage morphology is more clear. It can be noticed that the driving failure mechanisms are splitting cracks in the 0-degree plies and delamination in all the interfaces between the 0- and 90-degree plies. This is consistent with all the experimental results. Final fracture is localized at the notch's band.

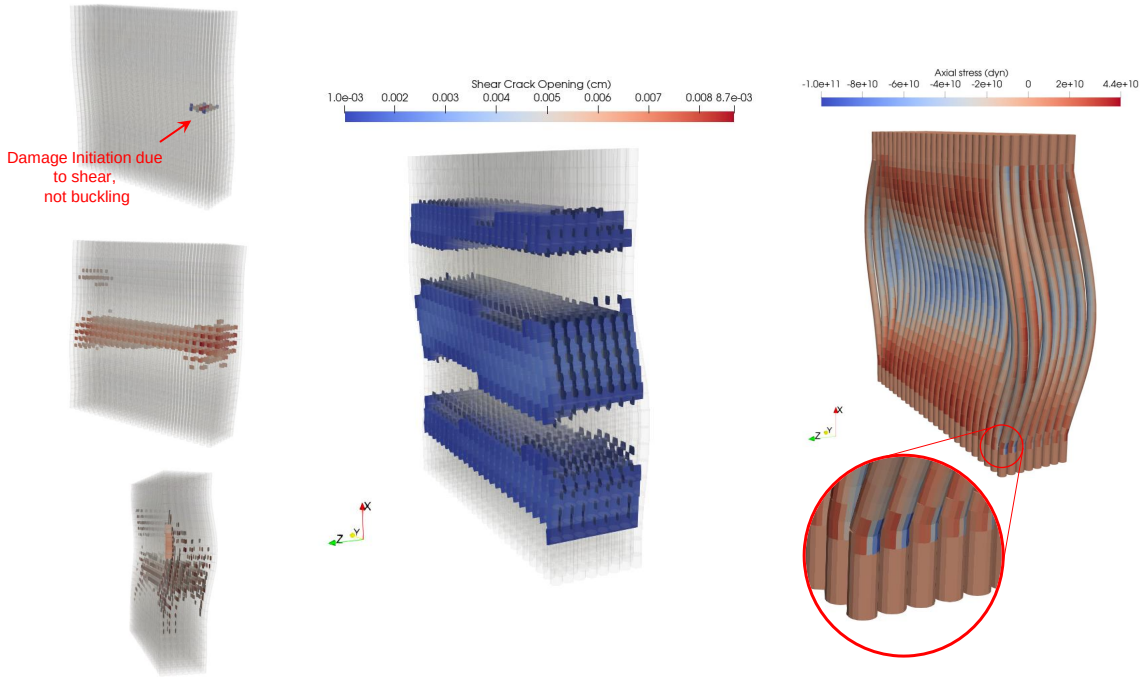


Figure 5.3: Compression calibration results. The damage initiation in these compression specimens is due to shear and not due to the buckling instability. The bands formed are consistent with results already obtained in real and simulation experiments [90]. Moreover, on the right a little detail of the benefits of using DFM as fiber breakage is capture explicitly and no element erosion or element deletion is needed.

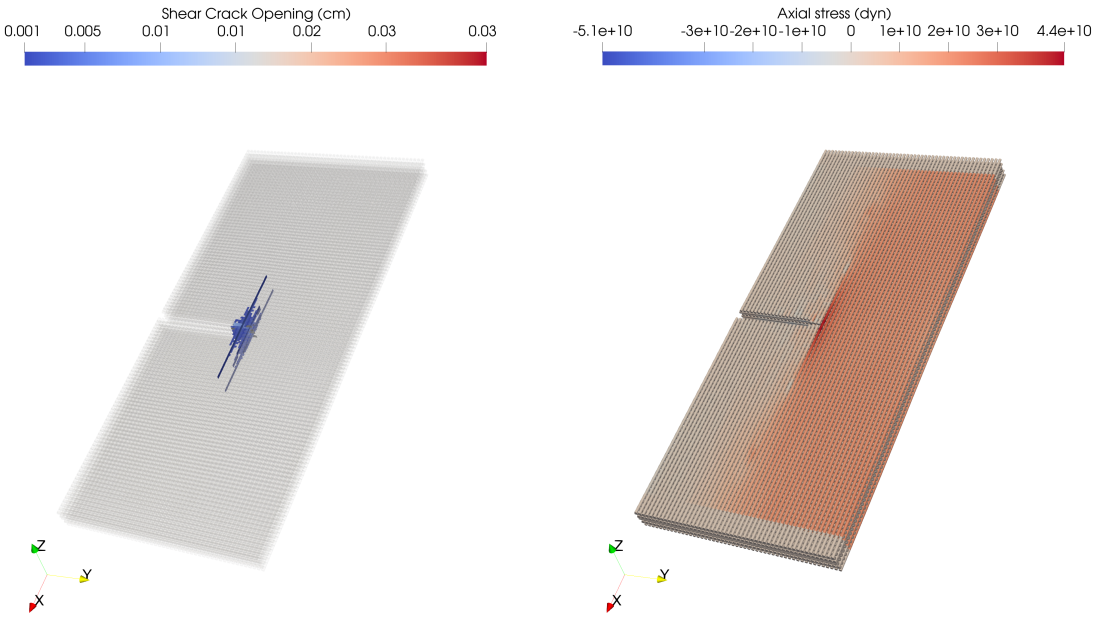


Figure 5.4: Detailed picture of the cross ply $[0/90]_{2s}$ at initiation of failure. Extensive splitting cracks are found in the 0-degree plies while extensive delamination is found in the interfaces between the 0- and 90-degree plies.

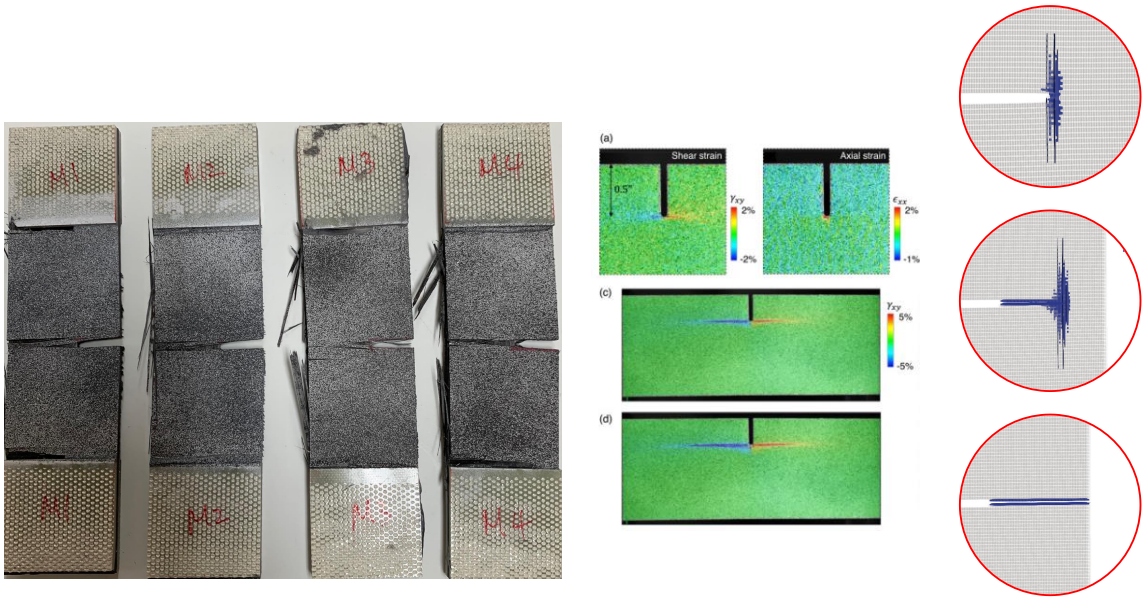


Figure 5.5: Comparison of fracture morphology for the $[0/90]_{2s}$ laminate with results found in [91].

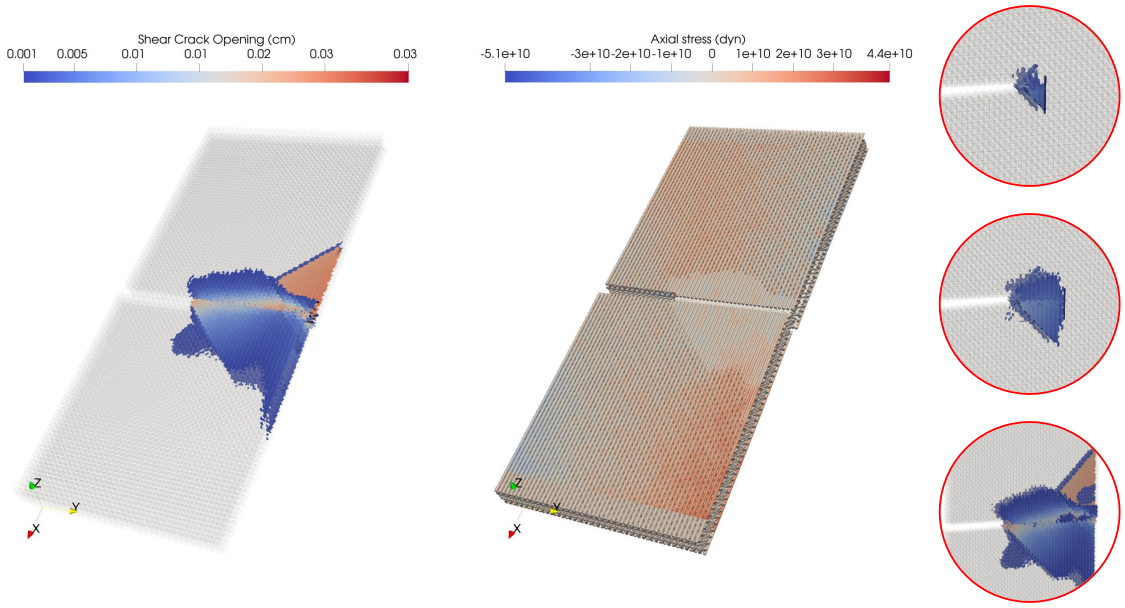


Figure 5.6: Fracture morphology for the $[0/45/90/\text{-}45]_s$ laminate.

5.3.2 Single Edge Notched Tension $[0/45/90/\text{-}45]_s$ Laminate

While experimental tests were not found for the quasi-isotropic SENT IM7/977-3 laminate, the simulation was ran to be able to verify the model runs without issue and the results are still consistent with fracture morphology expectations. The results are shown in fig. 5.6 and the fracture is driven by both the 0- and 45-degree plies failing. Splitting cracks is still present but less predominant compared to the cross-ply specimens; however, delamination behavior is heavily dominant. Areas of delamination span much further than the notch's band. Another reason why discrete models work better in capturing the complex and multi-faceted fracturing behavior of composites is because all these damages can be captured explicitly and no assumptions need to be made on the location of the damage bands if continuum approaches are used. If this simulation was ran using a continuum approach, let's say the Crack Band Model [21], all the delamination and splitting might have been much harder to fully capture.

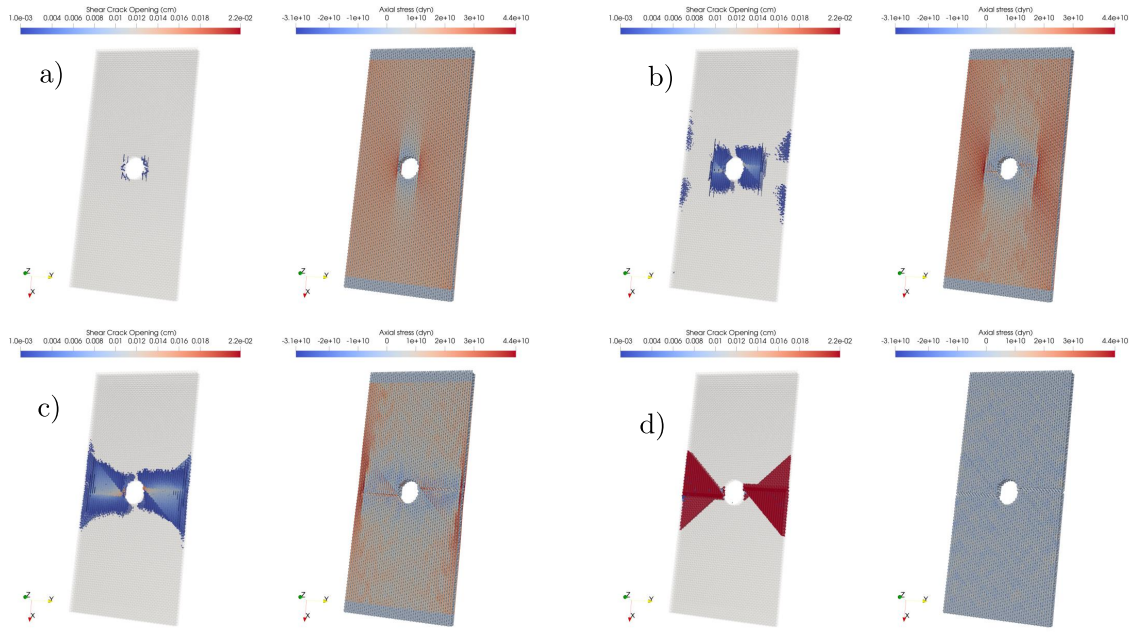


Figure 5.7: Fracture morphology for the $[0/45/90/-45]_s$ laminate.

5.3.3 Open Hole Tension $[0/45/90/-45]_s$ Laminate

Finally the last verification simulation was performed by analyzing an Open Hole Tension (OHT) cross-ply $[0/45/90/-45]_s$ specimen and the results are shown in fig. 5.7. Figure 5.7a shows the moment damage starts occurring around the hole. Figure 5.7b shows how the crack propagates and half-way through final failure and fig. 5.7c-d show the moments right before and after final failure occurs. These morphological results were compared also to existing similar composite laminates as shown in fig. 5.8 [92, 93]. The classical butterfly final failure is observed due to the angled 45-degree plies and through the thickness, spiraling behavior is also observed.

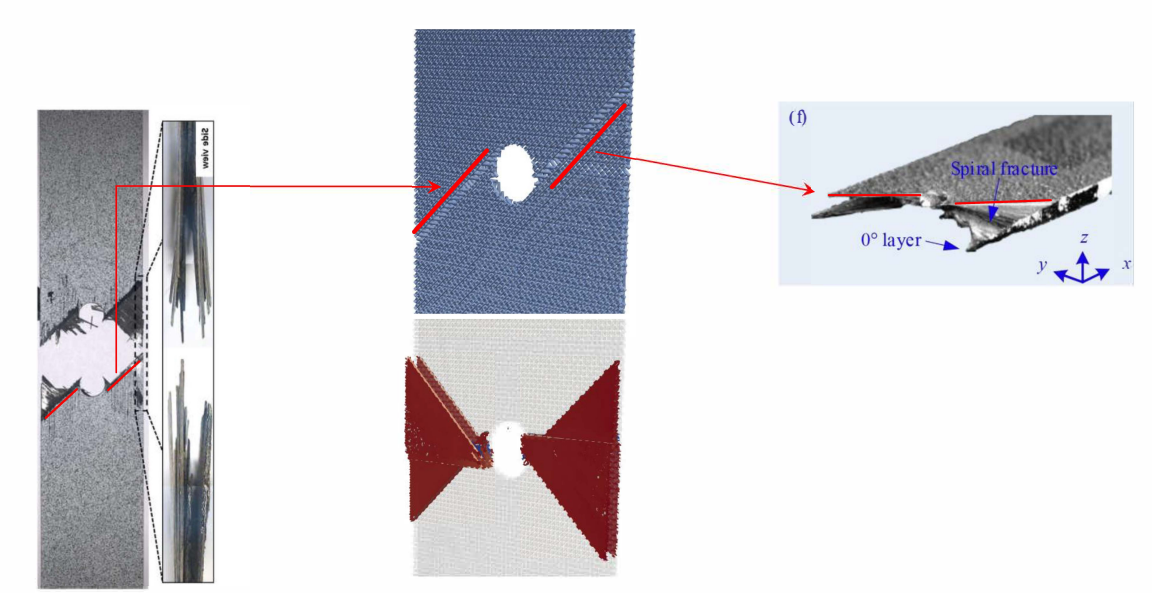


Figure 5.8: Comparison of Fracturing Behavior of OHT from [92, 93].

CONCLUSIONS

To conclude, the novel discrete approach FastDM4C with a working scale capable of covering from microscale (at the ply level) up to the mesoscale (at the laminate level) is presented. The model is based on the use of Representative Unit Cells whose sizes are in the order of the average distance between splitting cracks found in composites. The geometries of interest are overlaid on top of these arrays of RUCs and an efficient trimming algorithm is used to create final ply geometries. A Delaunay and Voronoi Tessellation techniques are used to connect the different plies via nodal connections where also Facet Elements are placed in the mid-plane parallel to the plies in order to better capture interlaminar behavior. The material and geometrical properties of the RUC are found via optimization techniques both for the elastic and the fracture behavior. Moreover, the optimization techniques presented covered both standard gradient-based and derivative-free methods, together with more advanced Machine Learning and Artificial Intelligence techniques with the use of Neural Network coupled with a relational SQLite database deployed on HPC clusters on thousands of CPUs. The computational results show very good agreement with experimental results during the calibration process, capturing very well also the inherent and multi-faceted complex failure mechanisms of the fracturing of composites. The FastDM4C is also presented as a very fast alternatives to general discrete approaches. Thanks to the development of the Discrete Fiber Model and Discrete Matrix Model, the new elements and their constitutive relationships can capture the orthotropic behavior of composites, not only in the elastic regime, but also during failure with fewer integration points. While more validation results are needed to further showcase the model, the current results already show that this new computational framework is advantageous as computationally cheap to run and still able to capture multiple failure mechanisms in details. Similarly to its sibling model DM4C developed in parallel, both discrete approaches are paving the way for an effective tool to revolutionize the composite design in the aerospace field.

FUTURE WORK

There is always room for improvements and this section is not meant to address all the problems of the proposed methodology; nonetheless, a few work-in-progress and future modifications and implementations are presented. The first improvement is about the orientation of the polygon facets in the FacetElement. In the presented model, the facets are oriented in a pre-determined fashion depending on what their are trying to characterize (either pure matrix, interlaminar, or diagonal coupling behaviors). However, from basic mechanics, cracks tend to develop in relation to the plan of maximum principal stress and strain. Therefore, all the facets in the models could have the possibility to rotate to satisfy such mechanism. Being a discrete implementation where the stresses and strains are described in a vectorial manner rather than a tensorial one, such strains at the facet level cannot be instantaneously rotated using classical rotational matrices but rather need to be rotated and corrected using the Principle of Virtual Work. These equations have been derived and already implemented, providing exciting preliminary results which for certain will be better verified and validated in the near future.

Another work-in-progress is the application of the FastDM4C for Additive Manufacturing (AM) and Fused Deposition Modeling (FDM) of composites. Such preliminary work has already been done with the sibling model DM4C and it is of particular interest as new printing technologies for composites are quickly emerging and are being actively adopted by both academic and industry sectors. The capability of inserting fiber waviness perpendicular to the direction of loading has shown great results in maintaining the targeted structural strength while improving energy dissipation.

Finally, given the predisposition of the solver MARS to OOP protocols, a new parallel implementation using Graphics Processing Unit (GPU) can be fairly beneficial. CUDA as a programming language based on C++ has already been proven to provide a massive boost to problems where routines can be easily parallelized such as the ones in explicit methods.

BIBLIOGRAPHY

[1] Charles Edward Inglis. Stresses in a plate due to the presence of cracks and sharp corners. *Trans Inst Naval Archit*, 55:219–241, 1913.

[2] Alan Arnold Griffith. 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):163–198, 1921.

[3] A Griffith. The theory of rupture. In *First Int. Cong. Appl. Mech*, pages 55–63, 1924.

[4] Hideo Kobayashi and Hisahiro Onoue. Brittle fracture of liberty ships. *Failure Knowledge Database*, 100:67, 1943.

[5] PA Withey. Fatigue failure of the de havilland comet i. *Engineering failure analysis*, 4(2):147–154, 1997.

[6] William R Hendricks. The aloha airlines accident—a new era for aging aircraft. In *Structural integrity of aging airplanes*, pages 153–165. Springer, 1991.

[7] M Jon Turner, Ray W Clough, Harold C Martin, and LJ Topp. Stiffness and deflection analysis of complex structures. *journal of the Aeronautical Sciences*, 23(9):805–823, 1956.

[8] Wesley J Cantwell and John Morton. The impact resistance of composite materials—a review. *composites*, 22(5):347–362, 1991.

[9] Javier LLorca, Carlos González, Jon M Molina-Aldareguía, Javier Segurado, Rocio Seltzer, Federico Sket, Marcos Rodríguez, Sergio Sádaba, Raul Muñoz, and Luis Pablo Canal. Multiscale modeling of composite materials: a roadmap towards virtual testing. *Advanced materials*, 23(44):5130–5147, 2011.

[10] Y KUEN. *Composite materials: materials, manufacturing, analysis, design and repair*. CreateSpace, 2015.

[11] A. Elmarakbi. *Advanced Composite Materials for Automotive Applications: Structural Integrity and Crashworthiness*. Wiley, United Kingdom, 2013.

[12] A. Baker, S. Dutton, and D. Kelly. *Composite Materials for Aircraft Structures*. American Institute of Aeronautics and Astronautics, 2004.

[13] AM. Brandt. Fibre reinforced cement-based (frc) composites after over 40 years of development in building and civil engineering. *Composite Structures*, 86(3):3–9, 2008.

[14] C. Carloni and F. Focacci. Frp-masonry interfacial debonding: An energy balance approach to determine the influence of the mortar joints. *European Journal of Mechanics A-Solids*, 55:122–33, 2016.

[15] C. Ceccato, M. Salviato, C. Pellegrino, and G. Cusatis. Simulation of concrete failure and fiber reinforced polymer fracture in confined columns with different cross sectional shape. *International Journal of Solids and Structures*, 108:216–29, 2017.

[16] A. D'Ambrisi, L. Feo, and F. Focacci. Masonry arches strengthened with composite unbonded tendons. *Composite Structures*, 98:323–29, 2013.

[17] R. Kaiser. *Automotive Applications of Composite Materials*. National Technical Information Service, 1978.

[18] VM. Karbhari, J. Chin, D. Hunston, B. Benmokrane, T. Juska, R. Morgan, JJ. Lesko, U. Sorathia, and AD. Reynaud. Durability gap analysis for fiber-reinforced polymer composites in civil infrastructure. *Journal of Composites for Construction*, 7(3):238–47, 2003.

[19] A Forghani, M Shahbazi, N Zobeiry, A Poursartip, and R Vaziri. An overview of continuum damage models used to simulate intralaminar failure mechanisms in advanced composite materials. In *Numerical modelling of failure in advanced composite materials*, pages 151–173. Elsevier, 2015.

[20] Yrn R Rashid. Ultimate strength analysis of prestressed concrete pressure vessels. *Nuclear engineering and design*, 7(4):334–344, 1968.

[21] Zdeněk P Bažant and Byung H Oh. Crack band theory for fracture of concrete. *Matériaux et construction*, 16(3):155–177, 1983.

[22] Zdenek P Bažant, Jia-Liang Le, and Marco Salviato. *Quasibrittle fracture mechanics and size effect: a first course*. Oxford University Press, 2021.

[23] Donghyun Yoon, Sangdeok Kim, Jaehoon Kim, and Youngdae Doh. Development and evaluation of crack band model implemented progressive failure analysis method for notched composite laminate. *Applied Sciences*, 9(24):5572, 2019.

[24] Edmund F Rybicki and Melvin F Kanninen. A finite element calculation of stress intensity factors by a modified crack closure integral. *Engineering fracture mechanics*, 9(4):931–938, 1977.

[25] G Co Sih, PC Paris, and GR Irwin. On cracks in rectilinearly anisotropic bodies. *International Journal of Fracture Mechanics*, 1(3):189–203, 1965.

[26] IS Raju. Calculation of strain-energy release rates with higher order and singular finite elements. *Engineering Fracture Mechanics*, 28(3):251–274, 1987.

[27] Ronald Krueger, Isabelle L Paris, T Kevin O'Brien, and Pierre J Minguet. Fatigue life methodology for bonded composite skin/stringer configurations. *Journal of Composites, Technology and Research*, 24(2):308–331, 2002.

[28] Grigory I Barenblatt. The formation of equilibrium cracks during brittle fracture. general ideas and hypotheses. axially-symmetric cracks. *Journal of Applied Mathematics and Mechanics*, 23(3):622–636, 1959.

[29] R de Borst, Lambertus J Sluys, H-B Muhlhaus, and Jerzy Pamin. Fundamental issues in finite element analyses of localization of deformation. *Engineering Computations: Int J for Computer-Aided Engineering*, 10(2):99–121, 1993.

[30] JCJ Schellekens and René De Borst. On the numerical integration of interface elements. *International Journal for Numerical Methods in Engineering*, 36(1):43–66, 1993.

[31] Gianluca Cusatis, Daniele Pelessone, and Andrea Mencarelli. Lattice discrete particle model (ldpm) for failure behavior of concrete. i: Theory. *Cement and Concrete Composites*, 33(9):881–890, 2011.

[32] Gianluca Cusatis, Zdeněk P Bažant, and Luigi Cedolin. Confinement-shear lattice model for concrete damage in tension and compression: I. theory. *Journal of engineering mechanics*, 129(12):1439–1448, 2003.

[33] Hang Si. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software (TOMS)*, 41(2):1–36, 2015.

[34] Gianluca Cusatis, Andrea Mencarelli, Daniele Pelessone, and James Baylot. Lattice discrete particle model (ldpm) for failure behavior of concrete. ii: Calibration and validation. *Cement and Concrete composites*, 33(9):891–905, 2011.

[35] B Nayroles, G Touzot, and P Villon. Generalizing the finite element method: diffuse approximation and diffuse elements. *Computational mechanics*, 10(5):307–318, 1992.

[36] Ivo Babuška, Frank Ihlenburg, Ellen T Paik, and Stefan A Sauter. A generalized finite element method for solving the helmholtz equation in two dimensions with minimal pollution. *Computer methods in applied mechanics and engineering*, 128(3-4):325–359, 1995.

[37] Nicolas Moës, John Dolbow, and Ted Belytschko. A finite element method for crack growth without remeshing. *International journal for numerical methods in engineering*, 46(1):131–150, 1999.

[38] ML Williams. Stress singularities resulting from various boundary conditions in angular corners of plates in extension. *Journal of applied mechanics*, 19(4):526–528, 1952.

[39] Ted Belytschko, Giulio Ventura, and Jingxiao Xu. New methods for discontinuity and crack modeling in efg. In *Meshfree Methods for Partial Differential Equations*, pages 37–50. Springer, 2003.

[40] Theofanis Strouboulis, Kevin Copps, and Ivo Babuška. The generalized finite element method. *Computer methods in applied mechanics and engineering*, 190(32-33):4081–4193, 2001.

[41] Kuo-Chen Chou, Gerald M Maggiora, and Boryeu Mao. Quasi-continuum models of twist-like and accordion-like low-frequency motions in dna. *Biophysical journal*, 56(2):295–305, 1989.

[42] VB Shenoy, R Miller, EB Tadmor, R Phillips, and M Ortiz. Quasicontinuum models of interfacial structure and deformation. *Physical Review Letters*, 80(4):742, 1998.

[43] LAA Beex, RHJ Peerlings, and MGD Geers. A multiscale quasicontinuum method for dissipative lattice models and discrete networks. *Journal of the Mechanics and Physics of Solids*, 64:154–169, 2014.

[44] M Jon Turner, Ray W Clough, Harold C Martin, and LJ Topp. Stiffness and deflection analysis of complex structures. *journal of the Aeronautical Sciences*, 23(9):805–823, 1956.

[45] John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[46] Daryl L Logan. *A first course in the finite element method*. Cengage Learning, 2011.

[47] G Abaqus. Abaqus 6.11. *Dassault Systemes Simulia Corporation, Providence, RI, USA*, 2011.

[48] Starting Matlab. Matlab. *The MathWorks, Natick, MA*, 2012.

[49] Zdeněk P Bažant. Size effect in blunt fracture: concrete, rock, metal. *Journal of Engineering Mechanics*, 110(4):518–535, 1984.

[50] Robert M Jones. *Mechanics of composite materials*. CRC press, 1998.

[51] Stephen Timoshenko. *History of strength of materials: with a brief account of the history of theory of elasticity and theory of structures*. Courier Corporation, 1983.

[52] Cory Hage Mefford, Yao Qiao, and Marco Salviato. Failure behavior and scaling of graphene nanocomposites. *Composite Structures*, 176:961–972, 2017.

[53] Yao Qiao and Marco Salviato. A cohesive zone modeling study on the fracturing behavior of thermoset polymer nanocomposites. In *33rd ASC conference*, 2018.

[54] Yao Qiao, Antonio Alessandro Deleo, and Marco Salviato. A study on the multi-axial fatigue failure behavior of notched composite laminates. *Composites Part A: Applied Science and Manufacturing*, 127:105640, 2019.

[55] Marco Salviato, Kedar Kirane, Shiva Esna Ashari, Zdeněk P Bažant, and Gianluca Cusatis. Experimental and numerical investigation of intra-laminar energy dissipation and size effect in two-dimensional textile composites. *Composites Science and Technology*, 135:67–75, 2016.

[56] Yao Qiao and Marco Salviato. Study of the fracturing behavior of thermoset polymer nanocomposites via cohesive zone modeling. *arXiv:1808.09787*, 2018.

[57] Marco Salviato, Kedar Kirane, Zdeněk P Bažant, and Gianluca Cusatis. Mode i and ii interlaminar fracture in laminated composites: a size effect study. *Journal of Applied Mechanics*, 86(9), 2019.

[58] Seunghyun Ko, James Davey, Sam Douglass, Jinkyu Yang, Mark E Tuttle, and Marco Salviato. Effect of the thickness on the fracturing behavior of discontinuous fiber composite structures. *Composites Part A: Applied Science and Manufacturing*, 125:105520, 2019.

[59] Seunghyun Ko, Jinkyu Yang, Mark E Tuttle, and Marco Salviato. Effect of the platelet size on the fracturing behavior and size effect of discontinuous fiber composite structures. *Composite Structures*, 227:111245, 2019.

[60] Yao Qiao and Marco Salviato. Study of the fracturing behavior of thermoset polymer nanocomposites via cohesive zone modeling. *Composite Structures*, 220:127–147, 2019.

[61] Weixin Li, Yao Qiao, Joel Fenner, Kyle Warren, Marco Salviato, Zdeněk P Bažant, and Gianluca Cusatis. Elastic and fracture behavior of three-dimensional ply-to-ply angle interlock woven composites: Through-thickness, size effect, and multiaxial tests. *Composites Part C: Open Access*, 4:100098, 2021.

[62] Yuta Kumagai, Sota Onodera, Marco Salviato, and Tomonaga Okabe. Multiscale analysis and experimental validation of crack initiation in quasi-isotropic laminates. *International Journal of Solids and Structures*, 193:172–191, 2020.

[63] Yao Qiao, Qiwei Zhang, TROY Nakagawa, and MARCO Salviato. A size effect study on the splitting crack initiation and propagation in off-axis layers of composite laminates. In *36th American Society for Composites Conference*, 2021.

[64] Yao Qiao, Kaiwen Guo, and Marco Salviato. Size effect and scaling in quasi-static and fatigue fracture of graphene polymer nanocomposites. *Polymer Composites*, 2023.

[65] Jeremy Brockmann and Marco Salviato. The gap test–effects of crack parallel compression on fracture in carbon fiber composites. *Composites Part A: Applied Science and Manufacturing*, 164:107252, 2023.

[66] Marco Salviato, Viet T Chau, Weixin Li, Zdeněk P Bažant, and Gianluca Cusatis. Direct testing of gradual postpeak softening of fracture specimens of fiber composites stabilized by enhanced grip stiffness and mass. *Journal of Applied Mechanics*, 83(11), 2016.

[67] MATLAB. *version 9.9 (R2020b)*. The MathWorks Inc., Natick, Massachusetts, 2020.

[68] II Dikin. Iterative solution of problems of linear and quadratic programming. In *Doklady Akademii Nauk*, volume 174, pages 747–748. Russian Academy of Sciences, 1967.

[69] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311, 1984.

[70] Richard H Byrd, Jean Charles Gilbert, and Jorge Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical programming*, 89(1):149–185, 2000.

[71] Richard H Byrd, Mary E Hribar, and Jorge Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.

[72] Richard A Waltz, José Luis Morales, Jorge Nocedal, and Dominique Orban. An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Mathematical programming*, 107(3):391–408, 2006.

[73] H-M Gutmann. A radial basis function method for global optimization. *Journal of global optimization*, 19(3):201–227, 2001.

[74] Garret N Vanderplaats. *Numerical optimization techniques for engineering design: with applications*, volume 1. McGraw-Hill New York, 1984.

[75] John R Stinchcombe, Aneil F Agrawal, Paul A Hohenlohe, Stevan J Arnold, and Mark W Blows. Estimating nonlinear selection gradients using quadratic regression coefficients: double or nothing? *Evolution*, 62(9):2435–2440, 2008.

[76] Zdeněk P Bažant, Jia-Liang Le, and Martin Z Bazant. Scaling of strength and lifetime probability distributions of quasibrittle structures based on atomistic fracture mechanics. *Proceedings of the National Academy of Sciences*, 106(28):11484–11489, 2009.

[77] Zdeněk P Bažant and Yunping Xi. Statistical size effect in quasi-brittle structures: Ii. nonlocal theory. *Journal of engineering Mechanics*, 117(11):2623–2640, 1991.

[78] Yentl Swolfs, Ignaas Verpoest, and Larissa Gorbatikh. Issues in strength models for unidirectional fibre-reinforced composites related to weibull distributions, fibre packings and boundary effects. *Composites Science and Technology*, 114:42–49, 2015.

[79] Maha Alqam, Richard M Bennett, and Abdul-Hamid Zureick. Three-parameter vs. two-parameter weibull distribution for pultruded composite material properties. *Composite Structures*, 58(4):497–503, 2002.

[80] K Naresh, K Shankar, and R Velmurugan. Reliability analysis of tensile strengths using weibull distribution in glass/epoxy and carbon/epoxy composites. *Composites Part B: Engineering*, 133:129–144, 2018.

[81] MV Menon. Estimation of the shape and scale parameters of the weibull distribution. *Technometrics*, 5(2):175–182, 1963.

[82] Emil Artin. *The gamma function*. Courier Dover Publications, 2015.

[83] Bjarne Stroustrup. *The C++ programming language*. Pearson Education, 2013.

[84] Pelessone, Daniele. *MARS, Multi-physics Analysis of the Response of Structures, User's Manual, Version 22-02*. ES3 Document SD2202-1, San Diego, CA, 2022-02.

[85] Darren Engwirda. *Locally optimal Delaunay-refinement and optimisation-based mesh generation*. PhD thesis, University of Sydney, 10 2014.

[86] Takayuki Itoh and Kenji Shimada. Automatic conversion of triangular meshes into quadrilateral meshes with directionality. *Int. J. CAD/CAM*, 1(1):11–21, 2002.

[87] Richard D Hipp. SQLite, 2020.

[88] Stephen B Clay and Philip M Knoth. Experimental results of quasi-static testing for calibration and validation of composite progressive damage analysis methods. *Journal of Composite Materials*, 51(10):1333–1353, 2017.

[89] S Kyriakides, R Arseculeratne, EJ Perry, and KM0875 Liechti. On the compressive failure of fiber reinforced composites. *International journal of solids and structures*, 32(6-7):689–738, 1995.

[90] Geng Han, Zhidong Guan, Xing Li, Ruipeng Ji, and Shanyi Du. The failure mechanism of carbon fiber-reinforced composites under longitudinal compression considering the interface. *Science and Engineering of Composite Materials*, 24(3):429–437, 2017.

[91] Minh Hoang Nguyen and Anthony M Waas. Detailed experimental and numerical investigation of single-edge notched tensile cross-ply laminates. *Composite Structures*, 279:114731, 2022.

[92] Ronan M O'Higgins, Michael A McCarthy, and Conor T McCarthy. Comparison of open hole tension characteristics of high strength glass and carbon fibre-reinforced composite materials. *Composites Science and Technology*, 68(13):2770–2778, 2008.

[93] Hangyuan Luo, Hongshuai Wang, Zhiyong Zhao, Hongqian Xue, and Yujun Li. Experimental and numerical investigation on the failure behavior of bouligand laminates under off-axis open-hole tensile loading. *Composite Structures*, 313:116932, 2023.

[94] Zdeněk P Bažant and Pere C Prat. Microplane model for brittle-plastic material: I. theory. *Journal of Engineering Mechanics*, 114(10):1672–1688, 1988.

[95] Geoffrey Ingram Taylor. Plastic strain in metals. *J. Inst. Metals*, 62:307–324, 1938.

[96] Donald S Dugdale. Yielding of steel sheets containing slits. *Journal of the Mechanics and Physics of Solids*, 8(2):100–104, 1960.

[97] Boris Delaunay et al. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2, 1934.

Appendix A

STORAGE OF FEM MATRICES

The nodal information are stored following eq. (A.1). The first column is the index of the node, while the last three columns are the nodal coordinates.

$$\mathbf{N} = \begin{array}{c} \begin{array}{cccc} \text{Node Number} & \text{X-component} & \text{Y-component} & \text{Z-component} \end{array} \\ \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 2 & 1 & 1 & 1 \\ 3 & 2 & 2 & 2 \\ 4 & 3 & 3 & 3 \end{array}\right] \end{array} \tag{A.1}$$

The element connectivity matrix is stored following the convention from eq. (A.2). Since all the elements are 1D, only two nodes are expected to be assign for each element. The last column is for the property of the element. In the case of the FastDM4C, it is usually 1 for the fiber elements, 2 for the matrix elements, 3 for the diagonal elements, and finally 4 for the ply connectivity elements.

$$\mathbf{E} = \begin{array}{c} \begin{array}{cccc} \text{Element Number} & \text{Node 1} & \text{Node 2} & \text{Property} \end{array} \\ \left[\begin{array}{cccc} 1 & 1 & 2 & 1 \\ 2 & 2 & 3 & 1 \\ 3 & 3 & 4 & 1 \end{array}\right] \end{array} \tag{A.2}$$

A general form for the property matrix is shown in eq. (A.3). The second column is connected to the material card, while the rest of the columns are used to describe the geometrical property of the section.

$$\mathbf{P} = \begin{array}{c} \begin{array}{cccc} \text{Property Number} & \text{Material Number} & \text{Area} & \text{Inertia} \end{array} \\ \left[\begin{array}{cccc} 1 & 1 & 2 & 1 \\ 2 & 1 & 3 & 1 \\ 3 & 1 & 4 & 1 \end{array}\right] \end{array} \tag{A.3}$$

For material information, the material matrix shown in eq. (A.4) is used. Multiple parameters can be stored for each material card as the implementation of the solver can be generally different.

$$\mathbf{M} = \begin{matrix} \text{Material Number} \\ \\ \\ \end{matrix} \begin{matrix} E & \nu & G_f & f_t \end{matrix} \begin{bmatrix} 1 & 1 & 2 & 1 & 1 \\ 2 & 1 & 3 & 1 & 1 \\ 3 & 1 & 4 & 1 & 1 \end{bmatrix} \tag{A.4}$$

The boundary conditions information are stored following eq. (A.5). The first column is the node of interest, while the second column is the relative Degree of Freedom (DOF) and the last column is whichever value needs to be assigned. Assigning a value of 0 corresponds to fixing that particular DOF.

$$\mathbf{BC} = \begin{matrix} \text{Node Number} & \text{DOF} & \text{Value} \end{matrix} \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 0 \\ 3 & 0 & 0 \end{bmatrix} \tag{A.5}$$

Similarly to eq. (A.5), the forces can be assigned to specific DOF of each node as shown in eq. (A.6).

$$\mathbf{F} = \begin{matrix} \text{Node Number} & \text{DOF} & \text{Value} \end{matrix} \begin{bmatrix} 1 & 1 & 100 \\ 2 & 1 & 100 \end{bmatrix} \tag{A.6}$$

Appendix B

COMPUTATIONAL IMPLEMENTATION OF THE JACOBIAN MATRIX FOR NON-LINEAR ANALYSES WITHIN THE FEM FRAMEWORK

The closed form calculation of the Jacobian Matrix in an implicit scheme within the framework of the Finite Element Method (FEM) can be easily and efficiently implemented during the construction of the global stiffness matrix during the elemental sweep of the simulation domain. To further render the code efficient, particularly for larger scale problems, sparse matrix assembling is used. The format used is the *Compressed Sparse Column* (CSC) format, typical of MATLAB.

To give the reader a better understanding of the storage of sparse matrices using the CSC format, eq. (B.1) shows an example of a matrix with many zero entries as typical in global stiffness matrices. Equation (B.2) shows how eq. (B.1) can be stored using the *triplets* arrays instead.

The arrays I and J are integer arrays which store the row and column, respectively, of the non zero entries of the $[A]$ matrix. The size of such arrays is equal to the number of non-zero entries of the $[A]$ matrix. Computationally speaking, there are two ways to efficiently pre-allocate such arrays:

1. Calculate the number of non-zero entries and assign the proper size.
2. Pre-allocate the size of a given percentage size of the full matrix first and delete the unused elements after.

Depending on the density of the matrix and initial size, either method could be faster, not necessarily the first one.

$$A = \begin{bmatrix} 10 & 20 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 40 & 0 & 0 \\ 0 & 0 & 50 & 60 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \end{bmatrix} \tag{B.1}$$

$$\begin{aligned} I &= \begin{bmatrix} 1 & 1 & 2 & 2 & 3 & 3 & 3 & 4 \end{bmatrix} \\ J &= \begin{bmatrix} 1 & 2 & 2 & 4 & 3 & 4 & 5 & 6 \end{bmatrix} \\ W &= \begin{bmatrix} 10 & 20 & 30 & 40 & 50 & 60 & 70 & 80 \end{bmatrix} \end{aligned} \tag{B.2}$$

The arrays I, J, W can also be stored in different orders as long as the triplet coordination is respected. This allow an easy assemblage of the global stiffness matrix, for example, where same entries could also be called multiple times. Another important step is to re-order the sparse matrix using the Cuthill–McKee algorithm. In this way the bandwidth of the matrix is minimized and operations such as inversion are greatly reduced. Figure B.1 shows the difference between a non pre-conditioned (left) matrix and a matrix pre-conditioned using the Cuthill-McKee algorithm (right).

Non-linearities in FEM are generally due to (i) material non linearities, (ii) large deformation, and (iii) non-linear contacts. In this simple document, only the first case is explored. Materials can have multiple behaviors, but a simple way to describe many of them is to use a damage variable D which degrades the young modulus E. In the example of elastic-perfectly-plastic behavior, the damage variable D can be derived in the following way.

Assuming classical Hooke's law with the addition of the damage factor D:

$$\sigma = (1 - D)E\varepsilon \tag{B.3}$$

and knowing that after plastic initiation σ must be equal to the strength f_t, eq. (B.3) can be set equal to f_t and D accordingly found:

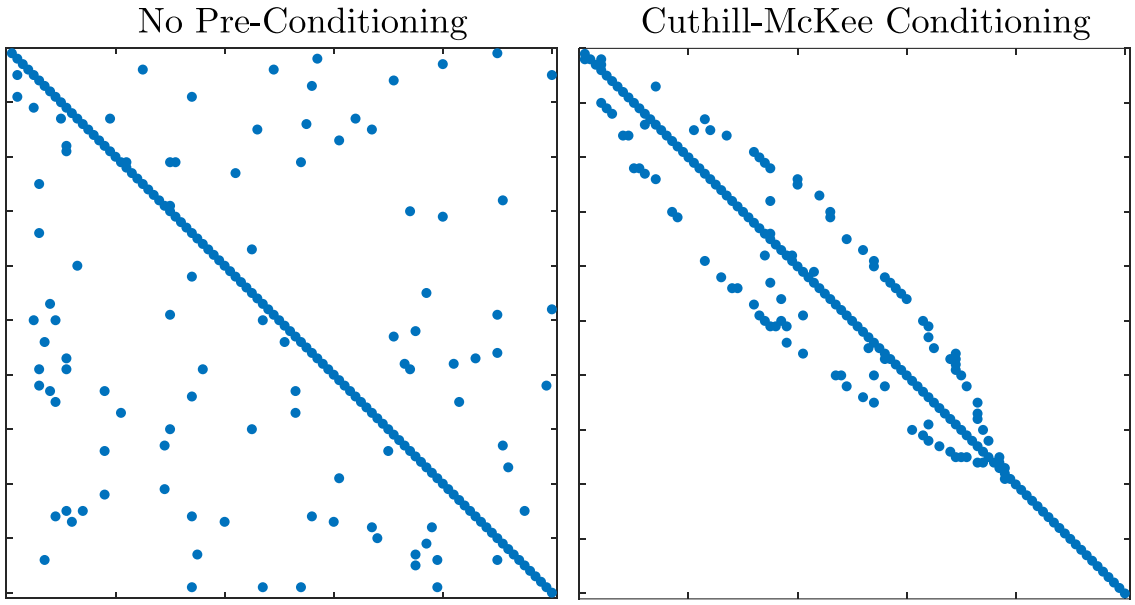


Figure B.1: Output plots from the given code. Graphical representation of non-zero elements in a matrix. Left matrix is not pre-conditioned while the matrix of the right is.

$$D = 1 - \frac{f_t}{E\varepsilon} \tag{B.4}$$

Equation (B.4) shows the highly non linear behavior of D, as ε is present in the denominator. The advantage of using the *degrading* method is that while the Hooke's law seems linear, the degrading factor D which is itself highly non-linear with respect to ε can be iterated and the basic linear FEM computational implementation still used.

Assuming the reader knows how to derive the element stiffness matrix of the bar element in 1D, the addition of the damage variable via principle of virtual work is straightforward:

$$\begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} = \frac{E(1-D)A}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \tag{B.5}$$

And the global system, assembled using standard techniques:

```
for i = 1 : Number_elements
    counter     =   counter + 1;
    % Find Nodal Coordinates of Node 1
    N1          =   N(N(:,1) == E(i,2),2:end);
    % Find Nodal Coordinates of Node 2
    N2          =   N(N(:,1) == E(i,3),2:end);
    % Find Length of the Element
    L(i)        =   N2(1)-N1(1);
    % Find Element Property ID
    propID      =   E(i,end);
    % Find Element Cross Sectional Area
    A(i)        =   P(P(:,1) == propID,3);
    % Find Element Material ID
    propMAT     =   P(P(:,1) == propID,2);
    % Find Young Modulus
    Emod(i)     =   M(M(:,1) == propMAT,2);
    % Find Strength
    ft(i)       =   M(M(:,1) == propMAT,3);

    K_local{i}  =   (A(i)*Emod(i)*(1-D(i))/L(i))*[1,-1;-1,1];

    E_conn{i}   =   [(E(i,2)*1),...
                     (E(i,3)*1)];
    % Assembly in the Global Stiffness Matrix
    K_global(E_conn{i},E_conn{i}) = ...
                    K_global(E_conn{i},E_conn{i}) + K_local{i};

    % If using sparse matrices, triplets are saved instead
    counterb = 0;
    for j = 1:2
        for k = 1:2
        counterb            =   counterb + 1;
            I(counterb)     =   j;
            J(counterb)     =   k;
            W(counterb)     =   Klocal{i}(j,k)
        end
    end
end
```

is generally written in matrix form as the following:

$$\{F\} = [K]\{d\} \tag{B.6}$$

which after applying boundary conditions it is written in reduced form:

$$\{F^{\text{red}}\} = [K^{\text{red}}]\{d^{\text{red}}\} \tag{B.7}$$

Since now D is non-linear, simple inversion of the $[K^{\text{red}}]$ cannot be done, and an iterative method must be used. The simplest to start with is the Newton-Raphson method. As a side note, multiple methods and hybrid or modified Newton methods are generally used, but out of the scope of this simple document.

During the extra Newton-Raphson step, the non-linear equilibrium equations can be approximated as:

$$\begin{aligned} &K^{\text{red}}(u_1+\Delta u_1,\cdots,u_n+\Delta u_n)\begin{Bmatrix} u_1+\Delta u_1 \\ \vdots \\ u_n+\Delta u_n \end{Bmatrix} - \{d^{\text{red}}\} = \{O\} \\ &\approx K^{\text{red}}(u_1,\cdots,u_n)\begin{Bmatrix} u_1 \\ \vdots \\ u_n \end{Bmatrix} - \{d^{\text{red}}\} + J(u_1,\cdots,u_n)\begin{Bmatrix} \Delta u_1 \\ \vdots \\ \Delta u_n \end{Bmatrix} \end{aligned} \tag{B.8}$$

where $J(u_1,\cdots,u_n)$ is the Jacobian matrix.

A clear example of a 2 DOF system 1D system should clarify how to construct the Jacobian Matrix J:

$$\begin{aligned} &K^{\text{red}}(u_1+\Delta u_1,u_2+\Delta u_2)\begin{Bmatrix} u_1+\Delta u_1 \\ u_2+\Delta u_2 \end{Bmatrix} - \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} = \\ &\approx \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}\begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} - \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix} + \\ &\begin{bmatrix} \dfrac{\partial}{\partial u_1}(K_{11}u_1+K_{12}u_2-F_1) & \dfrac{\partial}{\partial u_2}(K_{11}u_1+K_{12}u_2-F_1) \\ \dfrac{\partial}{\partial u_1}(K_{21}u_1+K_{22}u_2-F_2) & \dfrac{\partial}{\partial u_2}(K_{21}u_1+K_{22}u_2-F_2) \end{bmatrix}\begin{Bmatrix} \Delta u_1 \\ \Delta u_2 \end{Bmatrix} \end{aligned} \tag{B.9}$$

where reduced global stiffness matrix components K_{11} to K_{22} are assembled by sweeping over the elements in the domain and filling the corresponding DOF of each local elemental

stiffness matrix into the global one. At this point, the new increments are found using:

$$\begin{Bmatrix} \Delta u_1 \\ \vdots \\ \Delta u_n \end{Bmatrix} = -J(u_1, \cdots, u_n)^{-1} \left\{ K^{\text{red}}(u_1, \cdots, u_n) \begin{Bmatrix} u_1 \\ \vdots \\ u_n \end{Bmatrix} - \{d^{\text{red}}\} \right\} \tag{B.10}$$

and the updated nodal displacements are found iteratively as:

$$\begin{Bmatrix} u_1 \\ \vdots \\ u_n \end{Bmatrix}^{\text{new}} = \begin{Bmatrix} u_1 \\ \vdots \\ u_n \end{Bmatrix}^{\text{old}} + \begin{Bmatrix} \Delta u_1 \\ \vdots \\ \Delta u_n \end{Bmatrix} \tag{B.11}$$

by minimizing the following tolerance:

$$\left\| [K^{\text{red}}]\{u_n\} - \{d^{\text{red}}\}] \right\| < 10^{-6} \tag{B.12}$$

The tricky part is to systematically assemble the Jacobian matrix given any geometry and element connectivity matrix. This can be easily achieved with the introduction of a 3D matrix (similarly to a 3rd order tensor, even though it does not satisfy tensor properties) of the following form:

$$\frac{\partial K}{\partial u}(i, j, k) \tag{B.13}$$

Where the derivatives of the global (or reduced) stiffness matrix with respect each of the DOF u are stored. The indices i, j follow the DOF convention of the model, while the index k refers to the differentiation index. Figure B.2 is a visual representation of the 3D matrix. Now, it is possible to add a simple if-condition in order to start assemble the 3D matrix, which must be pre-allocated properly outside of the elemental loop. If the condition of plasticity is satisfied, the damage variable calculated (by either using elemental strain or nodal displacements) and the 3D matrix filled element by element.

```
1 % e(i) = axial strain of element i
```

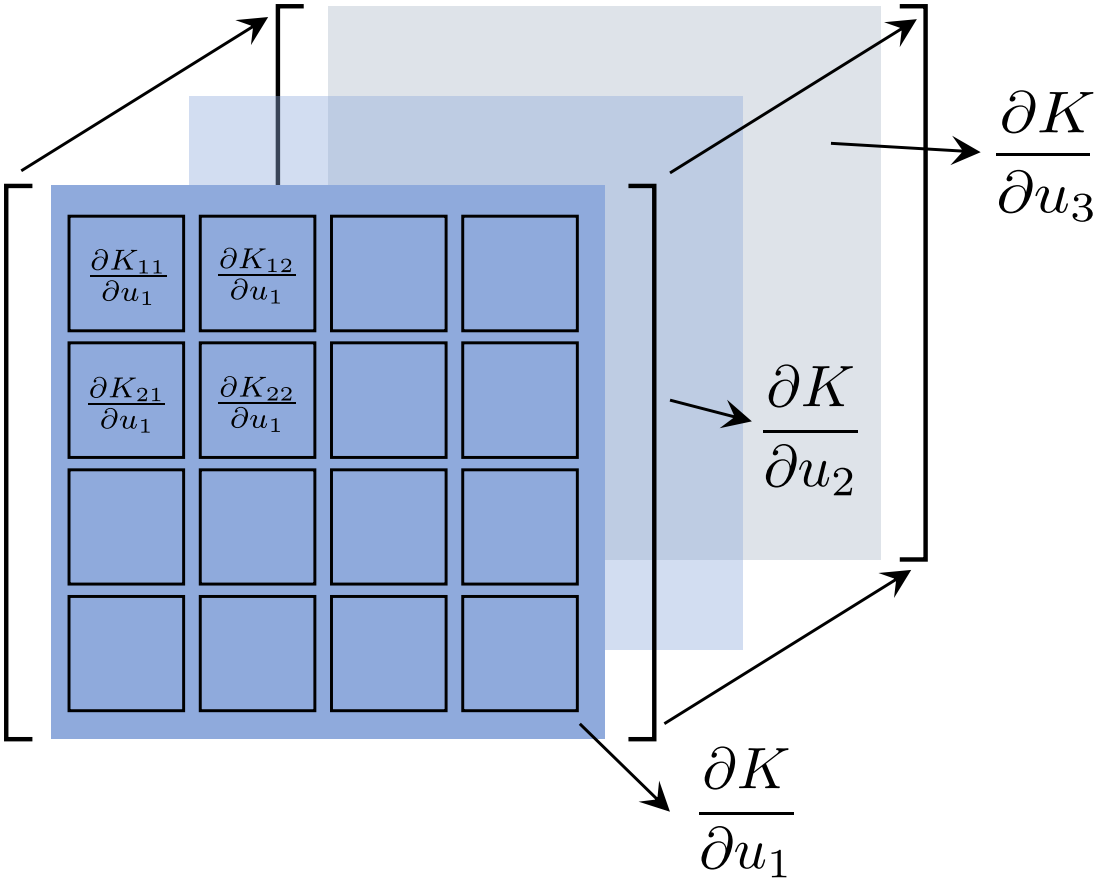


Figure B.2: Visual representation of the 3D matrix which contains the derivatives of the global stiffness matrix with respect each DOF.

```
% This if-condition is satisfied at plastic phase
if e(i) > ft(i)/Emod(i)
    % Calculate Damage Variable of the element
    D(i)       =   1 - ft(i)/(Emod(i) * e(i));
    % Calculate Derivative of the Damage Variable wrt nodal displacements
    dDdu(i,:)  =   [- ft(i)/Emod(i) * L(i) * 1 / (-e(i))^2,...
                    ft(i)/Emod(i) * L(i) * 1 / (-e(i))^2];
    % Fill the local dKdu and global one
    for j = 1 : size(dDdu,2)
        dK_local{i}(:,:,j)  =
↪   [1,-1;-1,1].*-dDdu(i,j)*(A(i)*Emod(i)/L(i));
        dK_global(E_conn{i},E_conn{i},E_conn{i}(j))    =
↪   dK_local{i}(:,:,j);
    end
end
```

where the derivative of the local elemental stiffness matrix is found using the following steps:

$$\frac{\partial K}{\partial u} = \frac{\partial}{\partial u}\left(\frac{AE(1-D)}{L}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\right) = \frac{AE}{L}\begin{bmatrix}-\frac{\partial D}{\partial u} & \frac{\partial D}{\partial u}\\ \frac{\partial D}{\partial u} & -\frac{\partial D}{\partial u}\end{bmatrix} \tag{B.14}$$

or simply:

$$\frac{\partial K}{\partial u} = -\text{sign}(K)\frac{\partial D}{\partial u} \tag{B.15}$$

please, note the negative sign in front of the sign operator.

Now, the full Jacobian matrix J can be calculated as shown in eq. (B.9):

```
% Pre-allocate the size of the Jacobian matrix
% This step is performed after reducing the global stiffness matrix
% rowes_to_keep = DOF not eliminated by BC
% u_old are the previous Newton step nodal displacements
% F_reduced are the reduced nodal forces from {F} = [K]{u}
% F_nh are the non-homogeneous nodal forces due to prescribed BC
J   =   zeros(length(rowes_to_keep));
counteri    =   0;
counterj    =   0;
```

```
for i = rowes_to_keep
    counteri    =   counteri + 1;
    for j = rowes_to_keep
        counterj    =   counterj + 1;
        J(counterj,counteri) = dot(dK_global(j,rowes_to_keep,i),u_old) -
↪   ...
                               F_reduced + ...
                               F_nh(j)*dK_global(j,j,i);
    end
    counterj    =   0;
end
J   =   J + K_global_reduced;
```

where the double for-loop can be simplified into a vector operation instead, by properly multiplying the rows of the derivative of the stiffness matrix with respect to nodal coordinates with the previous Newton step nodal displacement. Due to laziness, the `dot` command was used instead to avoid problems with array orientation in Matlab. The vectorization of the operation ans sparse conversion is very straightforward and left to the reader to find out as this was a temporary code to test the algorithm.

Finally, a `while` loop can be used to iterative both the Newton step and the incremental load step. As a side note, even if not shown in this document, the choice of the load increment can be dynamic as it can be increased if the previous step converged easily, or properly reduced if not. This will allow for a faster run-time and less problems.

A final example is shown in fig. B.3 where 3 1D bar elements undergo a prescribed displacement where the first element has a slightly less plastic strength to induce localization and perform better debugging. On the left, a too large load increment step is shown, where the Newton Raphson fails to properly minimize the tolerance, while on the right a smaller load increment is chosen to keep the tolerance below the wanted threshold at all times. Material and geometric properties can be found in the code attached at the end of the document.

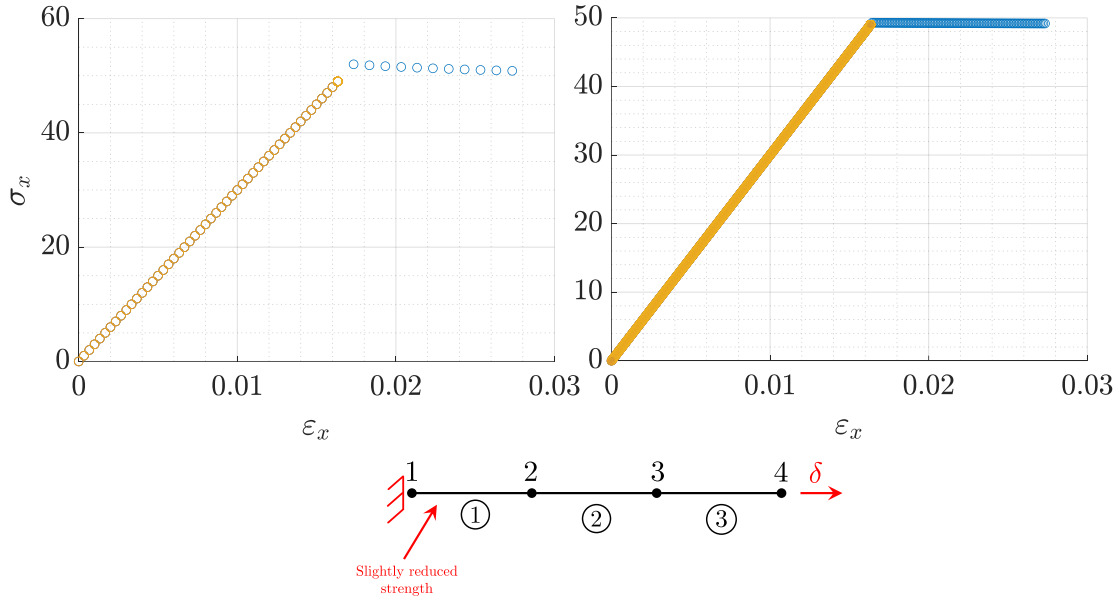


Figure B.3: Example results of a three-element structure with one element with a slight less strength of the other, to induce localization for debug purposes. On the left, a too large step is purposely used to show the issue to the reader if not being careful at convergence issues. On the right a much smaller load increment step is chosen.

Appendix C

DERIVATION OF THE STIFFNESS MATRIX FOR A BAR ELEMENT WITH DAMAGE USING THE PRINCIPLE OF VIRTUAL WORK

Using the Principle of Virtual Work (PVW) at the element level written in matrix form reads:

$$\begin{aligned}\{d\}^T\{f\} &= \int_{\Omega_e} \{\sigma\}^T\{\varepsilon\}A(s)ds = \\ &= \int_{\Omega_e} [B]^T E\,(1-D)\,[B]\,A\,(s)\,\mathrm{d}s\;\;\{d\}\end{aligned} \tag{C.1}$$

where $\{d\}$ are the nodal displacement vector, $\{f\}$ the nodal force vector, $\{\sigma\} = E(1-D)\{\varepsilon\}$ the stress vector, $\{\varepsilon\} = [B]\{d\}$ the strain vector, and finally $A(s)$ the element cross section. Then, it is easy to show, assuming linear shape functions which yields $[B] = [-1/L \quad 1/L]$, that:

$$\{f\} = \int_{\Omega_e} [B]^T E\,(1-D)\,[B]\,A\,(s)\,\mathrm{d}s\;\;\{d\} \tag{C.2}$$

which, for the case of uniform cross section area $A(s) = A$, gives:

$$\begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} = \frac{E(1-D)A}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \tag{C.3}$$

Using eq. (C.3), for the linear case (see Figure 3.6) the area under the curve can be expressed as the total fracture energy G_f divided by the length of the element L:

$$\frac{G_f}{L} = \frac{f_t \varepsilon_f}{2} \quad \rightarrow \quad \varepsilon_f = \frac{2G_f}{L f_t} \tag{C.4}$$

Accordingly, using a general form of a line and finding the slope and intercept which result in a stress equation $\sigma = f(\varepsilon)$ that when $\varepsilon = f_t/E \rightarrow \sigma = f_t$ and $\varepsilon = \varepsilon_f \rightarrow \sigma = 0$, and

comparing it with the definition of simple damage in eq. (3.19), it yields:

$$E(1-D)\varepsilon = \frac{Ef_t(2G_f - \varepsilon f_t L)}{2EG_f - f_t^2 L} \tag{C.5}$$

and solving for the damage variable D:

$$D = \frac{2\varepsilon EG_f - 2f_t G_f}{2\varepsilon EG_f - \varepsilon f_t^2 L} = \frac{2G_f(\varepsilon E - f_t)}{\varepsilon(2EG_f - f_t^2 L)} \tag{C.6}$$

Now, assuming a linear set of shape functions for the deformation of the bar:

$$u(x) = u_i\left(1 - \frac{x}{L}\right) + u_j\left(\frac{x}{L}\right) = u_i N_i + u_j N_j \tag{C.7}$$

where N_i and N_j are called indeed the shape functions. The strain can be defined as:

$$\varepsilon = \frac{du}{d\varepsilon} = \frac{u_j - u_i}{L} = u_i\frac{dN_i}{dx} + u_j\frac{dN_j}{dx} \tag{C.8}$$

Substituting eq. (C.8) into eq. (C.6), it is possible to obtain explicitly the damage variable D as a function of nodal displacements u_i and u_j of the bar:

$$D = \frac{2G_f(f_t L + E(u_i - u_j))}{(2EG_f - f_t^2 L)(u_i - u_j)} \tag{C.9}$$

Equation (C.9) is inherently highly non-linear; therefore, in the framework of the Finite Element Method, when constructing the global stiffness matrix, the damage variables $D^{(i)}$ for each i element pose a mathematical challenge as being non-linear. However, in an implicit scheme it can be easily solved implementing a simple Newton–Raphson step. In an explicit scheme it would not be an issue as the global stiffness matrix does not inversion. Generally, in compact form the FEM tries to solve the following system:

$$\{F\} = [K]\{d\} \tag{C.10}$$

which after applying boundary conditions it is written in reduced form:

$$\{F^{\text{red}}\} = [K^{\text{red}}]\{d^{\text{red}}\} \tag{C.11}$$

During the extra Newton-Raphson step, the non-linear equilibrium equations can be approximated as:

$$\begin{aligned} K^{\text{red}}(u_1+\Delta u_1,\cdots,u_n+\Delta u_n)\begin{Bmatrix} u_1+\Delta u_1 \\ \vdots \\ u_n+\Delta u_n \end{Bmatrix} - \{d^{\text{red}}\} = \{O\} \\ \approx K^{\text{red}}(u_1,\cdots,u_n)\begin{Bmatrix} u_1 \\ \vdots \\ u_n \end{Bmatrix} - \{d^{\text{red}}\} + J(u_1,\cdots,u_n)\begin{Bmatrix} \Delta u_1 \\ \vdots \\ \Delta u_n \end{Bmatrix} \end{aligned} \tag{C.12}$$

where $J(u_1,\cdots,u_n)$ is the Jacobian matrix and represents the derivatives of the damage variables $D^{(i)}$ with respect the nodal displacements u_i. Accordingly:

$$\begin{Bmatrix} \Delta u_1 \\ \vdots \\ \Delta u_n \end{Bmatrix} = -J(u_1,\cdots,u_n)^{-1}\left\{ K^{\text{red}}(u_1,\cdots,u_n)\begin{Bmatrix} u_1 \\ \vdots \\ u_n \end{Bmatrix} - \{d^{\text{red}}\} \right\} \tag{C.13}$$

and the updated nodal displacements are found iteratively as:

$$\begin{Bmatrix} u_1 \\ \vdots \\ u_n \end{Bmatrix}^{\text{new}} = \begin{Bmatrix} u_1 \\ \vdots \\ u_n \end{Bmatrix}^{\text{old}} + \begin{Bmatrix} \Delta u_1 \\ \vdots \\ \Delta u_n \end{Bmatrix} \tag{C.14}$$

by minimizing the following tolerance:

$$\left\| [K^{\text{red}}]\{u_n\} - \{d^{\text{red}}\}] \right\| < 10^{\text{-6}} \tag{C.15}$$

Equations (C.4) to (C.15) are extremely powerful because they outline a way to keep

in closed form all the damage description in a Finite Element framework. Moreover, the Jacobian matrix eq. (C.12) can be slightly modified to be able to serve as Hessian matrix in an optimization framework as derived in Section 3.6.

Appendix D

TRANSFORMATION MATRIX FOR A 1D BAR IN A 3D SPACE

While eq. (C.3) is for a bar element aligned in the x-axis, it is possible to transform both displacements and reaction forces into a three-dimensional framework with the use of direction cosine equations.

Local displacements (and similarly elemental forces), now expressed with a hat, can be rewritten for a 3D space:

$$\begin{Bmatrix} \hat{u}_1 \\ \hat{u}_2 \end{Bmatrix} = \begin{bmatrix} C_x & C_y & C_z & 0 & 0 & 0 \\ 0 & 0 & 0 & C_x & C_y & C_z \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ w_1 \\ u_2 \\ v_2 \\ w_2 \end{Bmatrix} \tag{D.1}$$

where the direction cosines C_x, C_y, C_z can be found using the nodal coordinates:

$$\begin{aligned} C_x &= \frac{x_2 - x_1}{L} \\ C_y &= \frac{y_2 - y_1}{L} \\ C_z &= \frac{z_2 - z_1}{L} \end{aligned} \tag{D.2}$$

At this point, pre-multiplying and post-multiplying both the nodal forces vectors and nodal displacements of eq. (C.3) by the direction cosine matrix in eq. (D.1) and simplifying the results it is possible to obtain the stiffness matrix for a bar arbitrarily oriented in a 3D space:

$$[K] = \begin{bmatrix} C_x & 0 \\ C_y & 0 \\ C_z & 0 \\ 0 & C_x \\ 0 & C_y \\ 0 & C_z \end{bmatrix} \frac{AE(1-D)}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} C_x & C_y & C_z & 0 & 0 & 0 \\ 0 & 0 & 0 & C_x & C_y & C_z \end{bmatrix} \tag{D.3}$$

VITA

Antonio Alessandro Deleo was born in Pesaro, Italy, in 1993. He studied most of his life in Italy and initially believed to undertake a career as a classical musician. He graduated from the National Conservatory of Music "G. Rossini" in Pesaro at the age of 16 with a Master of Music, but decided to instead study engineering subjects as his love for aerospace followed him everywhere. After graduating high school, he moved to Seattle thanks to his grandfather, the very first aerospace engineer of the family, where he started his Bachelor of Science at the University of Washington Seattle in 2012. Once graduated with the bachelor, he had plans to finish his master degree in Europe; however, he met his current advisor, Prof. Marco Salviato, who offered him a Ph.D. position; and therefore, he started a direct-PhD path in the same department. In 2023 he graduates with a Ph.D. from the Aeronautics and Astronautics Department at the University of Washington with his research focusing on advanced computational methods for the fracturing behavior of composites.

He welcomes your comments to `adeleo@uw.edu`.