©Copyright 2023
Collins S. Davis

Rain Erosion: From Multi-physics Modelling to Efficient and Cost-effective Qualification

Collins S. Davis

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

MASTER OF SCIENCE IN AERONAUTICS & ASTRONAUTICS

University of Washington

2023

Reading Committee:

Marco Salviato, Chair

Ed Habtour

Program Authorized to Offer Degree:
Aeronautics and Astronautics

University of Washington

Abstract

Rain Erosion: From Multi-physics Modelling to Efficient and Cost-effective Qualification

Collins S. Davis

Chair of the Supervisory Committee:
Associate Professor Marco Salviato
Department of Aeronautics and Astronautics

This thesis explores the structural implications of the rain erosion of thin film coatings on exterior aircraft surfaces through the lenses of testing and finite element modelling (FEM). Currently, there is no accurate method to model the erosion of exterior aircraft surfaces by rain during flight. The primary objective of this research project is to develop novel experimental and simulation techniques to predict the rain erosion degradation of coatings used by the aerospace industry. With the use of FEM, physical testing can be limited; thereby lowering costs, reducing time to market, and streamlining the qualification process.

This study began by recreating an analysis performed on the modelling of rain erosion on the leading edge of a wind turbine blade. This approach models a rain droplet with smooth particle hyrdo-dynamics (SPH) and has been successfully recreated. For the rain erosion of aircraft, computational fluid dynamics (CFD) was utilized to determine the stresses resulting from a rain droplet impact in lieu of SPH. Thin film materials were characterized through tensile and nanoindentation testing in order to model materials' elasto-viscoplastic properties for the finite element model. Nanoindentation testing allows for the direct extraction of a compressive reduced modulus and iteration techniques may be used to determine the plastic behaviour of each material. Interfacial property characterization is in progress which will aid in future crack propagation studies. Finally, the results from CFD simulations can be combined with material profiles and interfacial characteristics in order to model rain erosion across aircraft exteriors.

The tensile models have been successfully calibrated for all materials and the nanoindentaion modelling, although still in progress, is showing correlation for the materials that have been modelled thus far. Preliminary rain erosion simulations include a full 3D, elasto-viscoplastic flat plate model showing promising results matching the expected material response.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

ASTM: Abbreviation for the American Society for Testing and Materials.

CAD: Abbreviation for Computer Aided Design

CFD: Abbreviation for computational fluid dynamics.

CSM: Abbreviation for Chopped Strand Mat.

CHOPPED STRAND MAT: An isotropic composite fiber reinforcement mat made of chopped strands of roving.

CROSSHEAD: Moving component of a testing apparatus.

DOGBONE: A type of testing sample that resembles the shape of a dog's bone.

DIC: Abbreviation for Digital Image Correlation.

DISPLACEMENT CONTROLLED: Mechanism for controlling the movement of a crosshead based on the amount it moves.

FE: Abbreviation for Finite Element.

FEA: Abbreviation for Finite Element Analysis.

FEM: Abbreviation for Finite Element Model, Finite Element Modelling, or Finite Element Method, depending on the context.

GFRP: Abbreviation for Glass Fiber Reinforced Plastic.

HOMOGENEOUS: Made of the same substance throughout.

ISOSCOPE: Handheld precision measuring tool for use with non-conductive thin film coatings over non-magnetic metals such as aluminum or copper. Isoscopes use the eddy current method.

ISOTROPIC: (Of a material) identical properties as measured from any direction.

LIMIT LOAD: Maximum expected service load in a worst-case scenario.

LOAD CONTROLLED: Mechanism for controlling the movement of a crosshead based on the load it feels.

RAYLEIGH WAVE: An acoustic wave that travels over or near to the surface of a material.

SAMP: Abbreviation for a Semi-Analytical Model for the simulation of Polymers.

SPH: Abbreviation for Smooth Particle Hydrodynamics.

ULTIMATE LOAD: Maximum design load for a part.

WTB: Abbreviation for Wind Turbine Blade

ACKNOWLEDGMENTS

In my pursuit of a Master's degree, there have been many challenges to overcome; academically, socially, and personally, I am fortunate to have had the support I did. With this in mind, there are several people for whom I wish to express my gratitude.

First, I extend my thanks to my advisor, Professor Marco Salviato. While taking his Finite Element Modelling class in the Winter of my first year as a graduate student, Dr. Salviato posted a Canvas announcement regarding potential thesis research projects. Upon speaking to him after class, Dr. Salviato offered to take me under his wing as a rain erosion researcher. This initial vote of confidence inspired my hard work over the following year and put me in a place to begin researching early enough to be able to graduate on time.

Secondly, I would like to thank my family for their continued and unwavering support during my entire life and particularly throughout my academic career. My mother, Sally Ault; father, Peter Davis; and my grandparents, Glenn and Donna Ault, have all played a paramount role in my academic success.

Third of all, I am thankful for several professors who believed in and supported me over my academic career thereby helping to push me toward a Master's and Ph.D. In particular, Lieutenant Colonel John Bridge and Dr. Steven Collins were instrumental in pushing me toward success in my undergraduate career while Dr. Ed Habtour's faith in me, academically, inspired my initial desire for a Ph.D.

Next, I thank Scott Adams from Boeing for his mentorship throughout this project and beyond. His friendly and respectful interactions and guidance have aided in my success with this project and have motivated me to consider a technical fellowship in my future.

Finally, I am thankful for my girlfriend, Noelle Hardman, who I met one month prior to starting the Master's program at UW. Noelle's strong and steady love, support, and head scratches have helped to ease my stress and sustain my self-confidence.

DEDICATION

to my parents, Peter and Sally

Chapter 1

INTRODUCTION

1.1 Rain Erosion

Throughout history, water has been and continues to be the cause of the most astounding erosion. From the massive glaciers that carved canyons to the rain that attacks airplanes in the sky, water never ceases eroding. Looking at aircraft specifically, the concern of rain erosion made itself abundantly clear during and directly following World War II with an early erosion test unintentionally taking place in the pacific theater on B-29 bombers in 1944 [1]. Since then, research has been performed in the name of understanding rain erosion. Wings with eroded leading edges were found to significantly increase aerodynamic drag and decrease fuel efficiency [2]. Because such rain erosion during flight is an inevitable part of any service environment, be it military missions, commercial, or recreational travel, it is imperative to defend aircraft from this natural threat.

Before exploring the means by which surfaces are protected from rain erosion, we must consider the mechanism of rain erosion. The worst case scenario angle of droplet impact for rain erosion (and other liquid impingement) has been shown to be perfectly normal to the surface of interest [1], [3]. In the process of erosion by rain at a normal angle of attack, a water droplet impacts a surface and applies an initial pressure. This sudden introduction of localized pressure gives way to a longitudinal wavefront followed by a shearing wavefront as depicted in Figure 1.1 [1]. Likewise, a Rayleigh wave (surface wave) is formed and disperses approximately 2/3 of the total energy of the impingement [1]. Below the surface, a reflected tensile wave pulls on the interface.

To protect exterior surfaces from rain erosion and other degrading conditions, modern aircraft employ multi-material coating systems. These coatings must be able to withstand a myriad of conditions during their 3-10+ year lifetime and must pass qualification before they have a spot on the production line [4]. The current method of qualification for coating

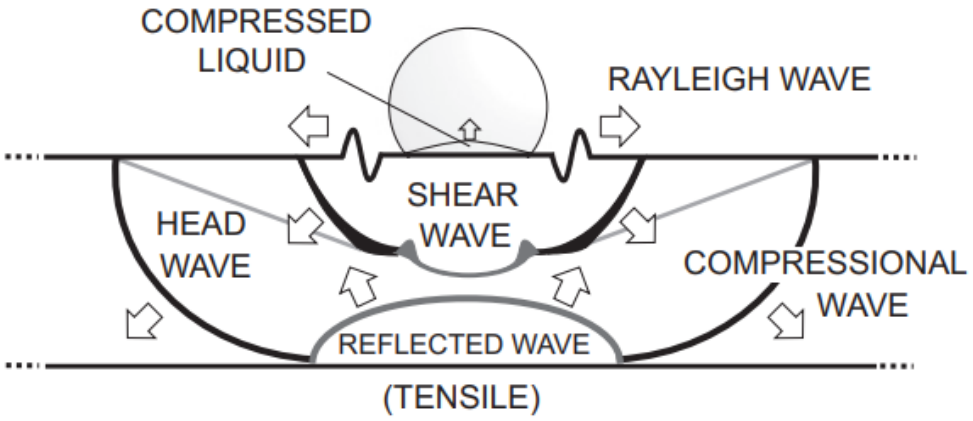


Figure 1.1: Wave Propagation of Droplet Impact [1]

systems against rain erosion is time consuming, expensive, and inefficient.

1.1.1 Whirling Arm Test

A whirling arm test is used in the coating qualification process and often requires the production of thousands of test coupons, each of which takes approximately 24 hours to be prepared [5]. Further, this type of test is expensive and creates conditions potentially more extreme than the actual service environment [1], [5].

The whirling arm test dates back to eighteenth century England where it was first employed by mathematician Benjamin Robins [6]. Robbins used the whirling arm apparatus to prove that air resistance had a substantial impact on the flight of projectiles in 1746. The whirling arm has evolved much since then and can be used in various fields of study and types of testing. For instance, a whirling arm may be used in a wind tunnel to characterize lift and drag over an airfoil. This thesis will focus on the whirling arm's application to rain erosion and henceforth, any mention of the whirling arm test will refer to the subsequently described rain erosion whirling arm.

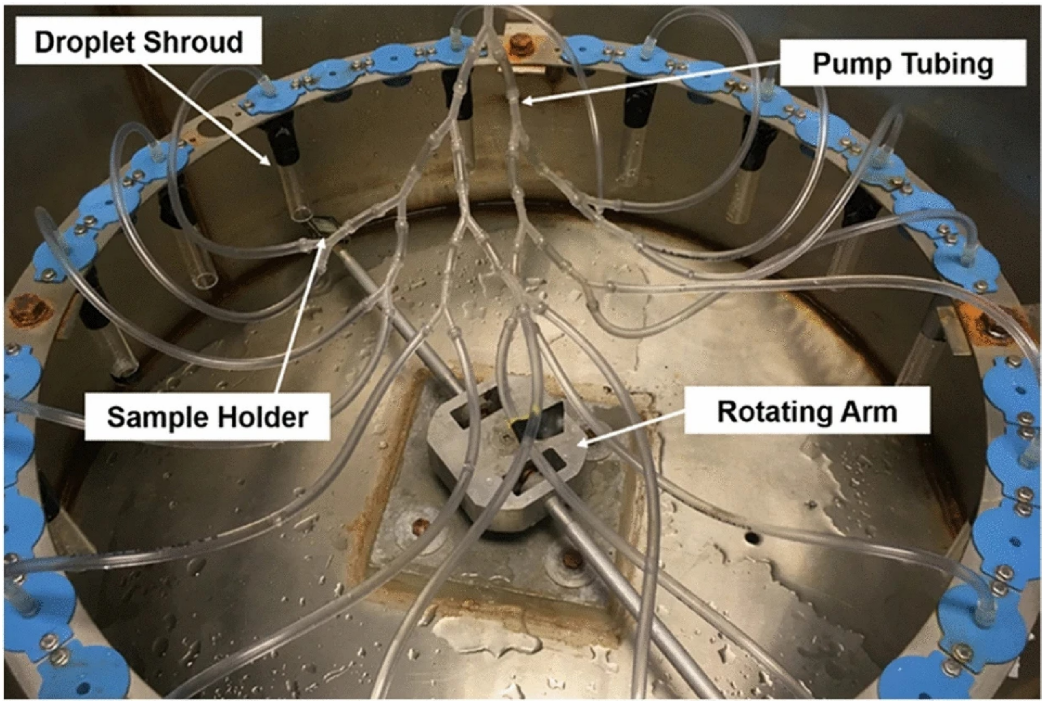


Figure 1.2: Annotated Whirling Arm Apparatus [7]

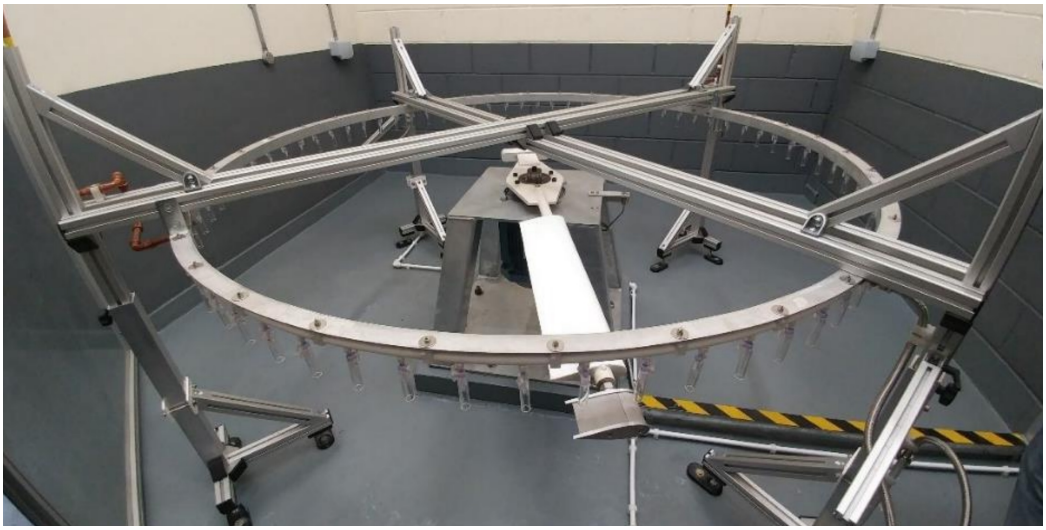


Figure 1.3: Whirling Arm Test Rig [8]

In a whirling arm test, test coupons are fixed at the end of a propeller type blade rotating at speeds usually around 220 m/s or 1100-1200 RPM [5], [8]. While spinning, samples are subject to a mist of water to simulate rain and tested to a critical number of cycles, run-time, or to failure. There raindrops are often simulated by way of dripping water through a needle to obtain a precise droplet size, usually 2 mm [8], [9]. We can see an example of a whirling arm test rig in Figure 1.2 from Pugh and Stack's study into rain erosion maps. In the figure, we notice several shrouds from which droplets are expelled as the sample rotates at the end of an arm beneath them. The shrouds act as a guard to protect the droplets from the swirling air around the whirling arm preceding their impact with the sample [9]. For a better idea of how large these machines often are, Figure 1.3 shows an apparatus made by the Energy Technology Centre [8].

1.2 Thin Film Coatings and Coating Systems

Thin film coatings are thin layers of material applied to a surface, often in the form of spray paint, although not necessarily like the spray paint one can purchase from a hardware store. They have very strong adhesion, and the films tend to be epoxy or polymer based. The primary purposes of aircraft coatings are to:

- Protect the aircraft from environmental damage such as erosion, corrosion, and ultra-violet rays.
- Reduce drag and improve lift as compared to a bare metal or composite surface.
- Give the airplane its color and branding.

The possibilities for types of coating systems stretch as wide as the human imagination. This thesis will focus specifically on two and three layer coating systems.

Three layer coating systems begin with a primer painted onto a bare substrate followed by a basecoat over the primer and clearcoat over top of the basecoat. A two layer coating system consists of a primer applied directly to a bare substrate with a topcoat applied over the primer where the topcoat fulfills the role of both a basecoat and clearcoat simultaneously. Each layer

of paint has a specific duty. Primers are always the bottom layer of a coating system and the most important for adhesion to the substrate. They provide a base from which the rest of the system is built on, not unlike the foundation of a building. Primers seal the substrate to protect the parent material from environmental attacks as listed above and must exhibit strong adhesion. They often have specialized chemicals such as hexavalent chromium (for corrosion resistance) built in to further protect the substrate or aid in adhesion. Next, the basecoat is what gives a coating system its color. Finally, the clearcoat provides a smooth and strong enamel like finish to the coating system to protect the entire layup from the service environment. As mentioned above, in two layer systems, the topcoat accomplishes the job of both the basecoat and clearcoat, that is, the topcoat colors the system and provides a protective outer shell at the same time.

Four coating materials have been analyzed in the work presented herein and they are referred to by their material type as there is one of each type mentioned above: The Topcoat, the Primer, the Basecoat and the Clearcoat. Therefore, capital letters indicate the specific paint analyzed while lowercase letters indicate the broader type of paint (e.g., "Topcoat" vs "topcoat").

1.2.1 Applications

Thin film coatings and paints are as ubiquitous as the air we breathe. Many applications are mainly aesthetic, but few are solely aesthetic. Take cars for instance. Paint is a way someone can uniquely call their car their own but more importantly, the car's paint protects the metal and plastic beneath it from road debris, rain, corrosion, UV light, and more. Automotive painting applications often utilize a three layer coating system.

1.3 Finite Element Modelling

FEM is used as a numerical method to solve complex engineering and physics problems across a wide realm of disciplines including but not limited to fluid flow, structures, and heat transfer [10]. FEM is the desired method of problem solving for applications where the geometry and/or interactions are too complex to be fully captured in any existing analytical closed form solution as well as dynamic simulations where time dependence comes into play.

1.3.1 LS-Dyna

All of the FEA discussed herein takes place in the LS-Dyna environment. LS-Dyna is a keyword based multi-physics solver known for its explicit solving capabilities. The software spawns from DYNA3D which was originally developed at the Lawrence Livermore National Laboratory in 1976 for the stress analysis of impact loadings and is now owned by ANSYS [11]. 1989 saw the name change from DYNA3D to LS-Dyna as many new and useful features were added to the program, helping it become what it is today. As the years went by, even more updates and additional features continued and still continue to grow and improve upon LS-Dyna. A full list is available in Volume 1 of the LS-Dyna keyword user's manual [11]. LS-Dyna was selected as the modelling software for this project because it was proven useful to model rain erosion in a study on wind turbine blades, and because of its renowned explicit solving capabilities [3].

1.4 Thesis Motivation

The motivation for this thesis stems from a desire to better understand the material response to high velocity subsonic rain erosion and to streamline the qualification of coating systems by developing accurate models to predict the erosion of these coatings by rain. The largest benefit for Boeing is steps toward a quicker coating qualification process that will save the company time and money developing new coating systems in the future. They will also have an increased ability for design optimization as well as more information regarding the predicted lifetimes of various coatings, old and new. Similarly, this project lays the finite element groundwork for other companies, such as the airlines, to model and better understand the long term behaviour of coatings, thereby improving any planned maintenance and inspection of aircraft coatings. Further, the research work presented herein may prove helpful for those working with other types of erosion. For instance, the understanding of sand erosion on aircraft and water erosion on personal watercraft would both benefit from a similar analysis.

1.4.1 Normalization of Data

In an effort to keep Boeing's intellectual property, proprietary information, and coating material data safe, all test data regarding Boeing's materials in this thesis have been normalized with respect to the Topcoat. Stresses and strains are normalized with respect to the Topcoat in tension while loads and depths are normalized with respect to the Topcoat's nanoindentation results; and linear deflections are normalized with respect to the Topcoat in lap shear.

1.5 Literature Review

Numerical investigation of rain droplet impact on offshore wind turbine blades under different rainfall conditions: A parametric study [3]

1.5.1 Scope

This paper was chosen as a primary literature review because the authors demonstrated the ability to model rain erosion on a coating system. The application is different, but the underlying structural theory is very similar, and analyses are performed in both single droplet impact and multiple droplet impact cases, making this a great article from which to start.

The work by Amrit Shankar Verma and associates evaluates the effects of rain erosion on the leading edge of composite wind turbine blades and lays a modelling foundation in LS-Dyna which is used as a starting point for further modelling in this thesis. Specifically, A.S. Verma et al. analyze a layup of [-45 GFRP/45 GFRP/CSM] composite layers covered with a protective gelcoat as shown in Figure 1.4 where the gelcoat is similar in function to a protective thin film paint system. In the same figure, we see a rain drop modelled with SPH.

1.5.2 Methodology Overview

Verma's methods begin with an SPH validation on a 20 mm square by 1 mm tall aluminum plate. This validation involves simulating the impact of a 2.70 mm, 2.90 mm, and 3.54 mm SPH rain droplet moving at impact velocities ranging from 1.36 m/s to 2.99 m/s against

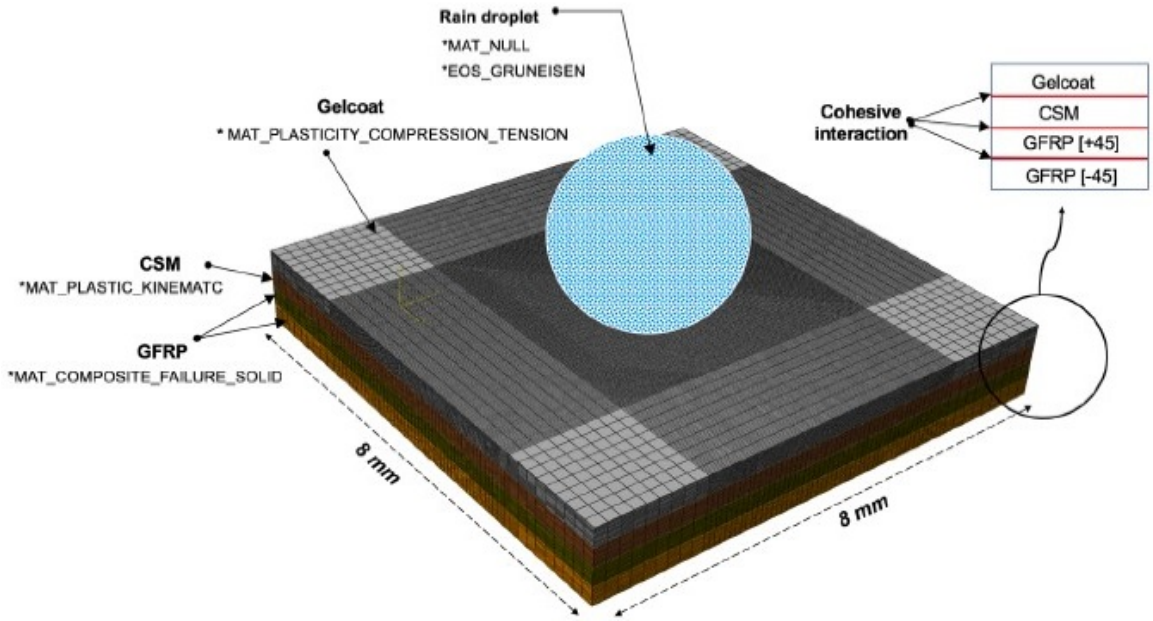


Figure 1.4: A.S. Verma and Company's Numerical Model [3]

the plate where the number of SPH particles making up the droplet is varied from 50,000 to 150,000 until the maximum force converges. This droplet convergence is coupled with a droplet morphology comparison between an actual rain drop and the SPH model.

The next step for Verma and associates was an analysis of the coating layup shown in Figure 1.4 under a single droplet impact. For this study, the paper considers droplets with diameters ranging from 1.30 mm to 2.74 mm analyzed at angles of attack of 30°, 45°, 60°, and 90°; and an impact velocity of 107.6 m/s. Then, each rainfall condition was modelled at different impact velocities from 87.66 m/s to 147.66 m/s in steps of 20 m/s with the worst-case scenario of the former reported at each velocity. From here, the WTB paper models the erosion by repeated rain droplet impacts to determine the number of impacts before failure in different conditions.

1.5.3 Modelling

This section is a high level analysis of the LS-Dyna modelling used by Verma and company. More detailed descriptions and insights into the specific parts relevant to the project goals of this thesis are in Section 3.1. For more information regarding SPH and other concepts that are not carried through this thesis, reference [3] is available from Elsevier. Models in the paper were built in LS-Dyna using the properties listed in Tables 1.1 through 1.3. Beginning with the convergence model, they modeled the aluminum plate with material card *MAT_PIECEWISE_LINEAR_PLASTICITY and the water droplet with *MAT_NULL paired with an equation of state based on Gruneisen formulation. Inside of the model, initial conditions prescribe a velocity upon the SPH droplet such that it will contact the plate at the desired impact velocity. The bottom of the plate is constrained in all degrees of freedom and the contact force between the droplet and the plate is taken as the impact force.

Material	$\rho\ (kg/m^3)$	$E(GPa)$	ν	$\sigma_y^t(MPa)$	$\sigma_y^c(MPa)$	$G(GPa)$
Aluminum	2820.0	70.00	0.3	240	240	27
Gelcoat	1150.0	2.50	0.4	90	120	—
CSM	1452.8	8.00	0.3	190	190	—
GFRP	1864.0	(1,1): 44.87 (2,2): 12.13 (3,3): 12.13	(1,2): 0.3 (1,2): 0.3 (2,3): 0.5	—	—	3.38

Table 1.1: Al, Gelcoat, CSM, and General GFRP Properties used by Verma et al. [3]

X_T	X_C	$Y_T = Z_T$	$Y_C = Z_C$	$S_{12} = S_{13} = S_{23}$	NFLS	SFLS
1006.3	487	45.95	131.90	49.51	45.95	49.51

Table 1.2: Additional GFRP Properties (MPa) used by Verma et al. [3]

Following the validation study, we move onto the droplet impact study. The leading edge flat plate model is smaller (planar area) than the convergence model at 8 mm square by

$\rho\ (kg/m^3)$	$\nu_s(Pa \cdot s)$	$C(m/s)$	a	$S1$	$S2$	$S3$
1000	0.001	1480	0	2.56	-1.986	0.226

Table 1.3: Water Properties used by Verma et al. [3]

1.2 mm deep. The bottom layer (GFRP -45) is fixed in all degrees of freedom and each layer is cohesively tied to its neighbors with the *CONTACT_AUTOMATIC_SURFACE_TO_SURFACE_TIEBREAK card. One of the benefits to using LS-Dyna is the existence of contact cards like this one. Cohesive interactions often require defining cohesive elements which are very small relative to the layers they are between and thus require a smaller time step and more computing power to solve explicitly than with a cohesively defined contact. Being a composite, GFRP was modelled with the *MAT_COMPOSITE_FAILURE_SOLID card which allows the simulation to consider progressive damage as this was of interest in the WTB study. The CSM layer made use of card *MAT_PLASTIC_KINEMATIC since it is reasonably isotropic and finally, the Gelcoat layer was modelled with material card *MAT_PLASTICITY_COMPRESSION_TENSION which is a material model that can consider the effects of strain rate and different behaviour in compression versus tension.

Lastly, the paper performed a repeated impact study to determine how many impacts of a droplet under certain conditions would result in rain erosion damage. This step makes use of the LS-Dyna full restart analysis which takes a binary dump file of all the stress and strain history as input and runs the same simulation over again. The resulting dump file then describes the material response after two droplet impacts and is used as input for the next restart and so on.

1.5.4 Results

The Verma paper presents several results from their modelling and the most important points are summarized below:

- From the convergence study, Verma and associates found convergence around 130,000 SPH particles. With a droplet of 130,000 SPH particles, they were able to match

the droplet morphology and experimental curve for a 2.7 mm droplet impacting at 2.67 m/s. We can see these results in Figure 1.5.

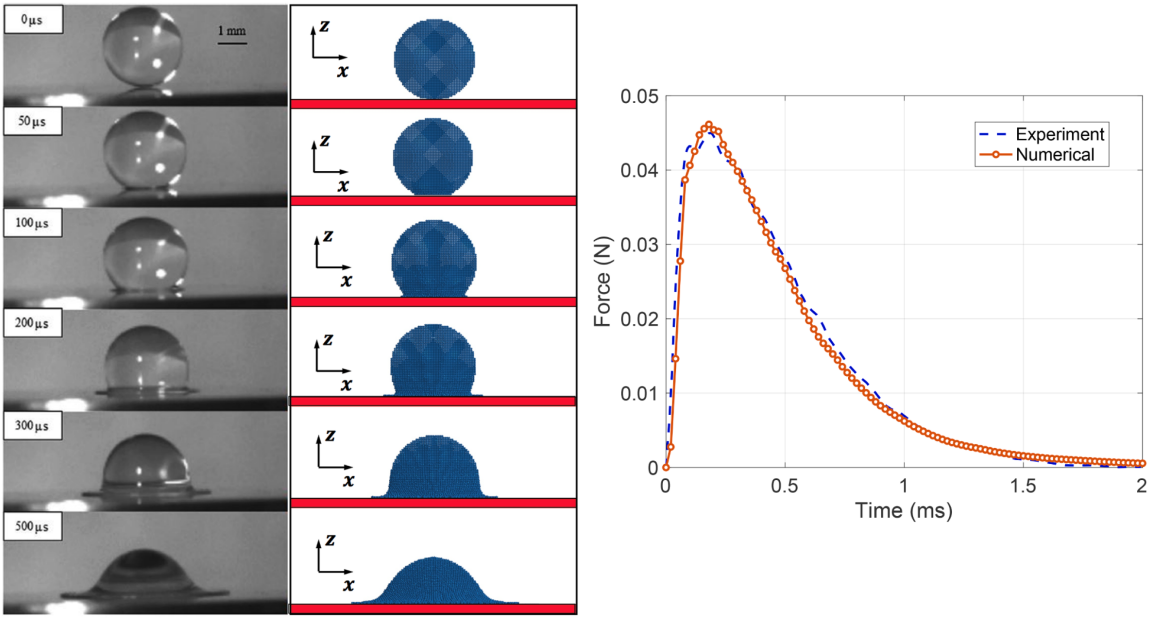


Figure 1.5: Droplet Morphology and Reaction Force Validation [3]

- Single droplet impacts on this system do not produce significant damage. This point is described thoroughly in the Verma paper and outline in Figure 1.6 where we notice that after one impact, the plastic strain of the gelcoat layer is less than 0.1 mm/mm under a worst case type loading condition.

- Over several impacts, fatigue damage causes erosion of coatings. Figure 1.6 demonstrates this point well. Here, we notice that higher velocity impacts create more damage per impact as expected and thus, the higher velocity leading edges erode much faster than the slower ones.

1.5.5 Conclusion as Applicable to This Thesis

We learned a lot from Verma et al. Some of the primary useful takeaways are as follows:

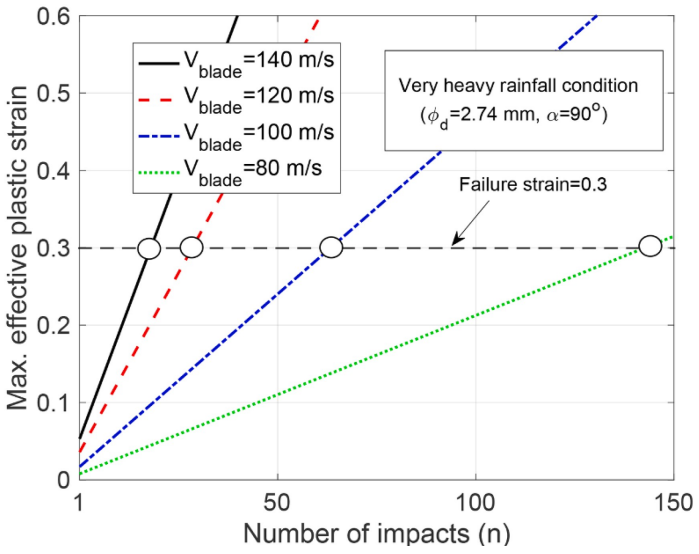


Figure 1.6: Impacts Until Failure Strain under Heavy Rainfall [3]

- **LS-Dyna is able to model the rain erosion of thin film coatings.** LS-Dyna is verified as an environment in which the erosion thin film coatings applied to a substrate may be modelled in single impact as well as in repeated impact scenarios as shown by the use of LS-Dyna's restart analysis. This insight is incredibly useful as the ultimate goal of the rain erosion project presented in this thesis is to predict the failure of thin film coating systems under fatigue loading.

- **LS-Dyna is capable of modelling cohesion with contacts.** The cohesive contacts used demonstrate LS-Dyna's capability to model a cohesive interface using contacts rather than cohesive elements. Cohesive contacts are useful in modelling the interface between each coating layer and the bottom layer with the substrate. Compared the alternative option of using cohesive elements (which must be very thin relative to their parent) cohesive contacts do not require the use of new elements allowing to model to consider the full thickness of each layer and theoretically run faster as there are not small elements to decrease the minimum time step.

Finally, we must address one major flaw in this study. The droplet morphology is cal-

ibrated at 2.67 m/s while impact velocities reach well over 100 m/s. A result of this is the carried assumption of a spherical droplet upon impact. For this reason, one cannot be entirely confident in the results as the morphology of the droplet is different at these faster speeds. The paper mentions that these results are conservative, and this is a likely reason why.

1.6 From CFD

This project has two sides: fluids and structures. The latter is presented in this thesis and is built on the work of the Fluids team. With that in mind, we must consider their chief discoveries. Using ANSYS Fluent and Computational Fluid Dynamics, the Fluids team was able to determine that when a rain droplet impacts an aircraft at cruising speed, it does not impact as a sphere. Rather, the rain droplet flattens and begins to split as it approaches the flow boundary layer into an wider disc-like shape which then impact the aircraft [12]. We can see evidence of this in Figure 1.7 from a CFD simulation [12]. Then, Figure 1.8 details the resulting surface wave of applied shear stresses [12]. It is important to note that this is *applied* and is independent of surface material. The material response to the applied stresses found with CFD is one of the major outcomes of the structures side of the project. The CFD simulation is outlined in Appendix A.

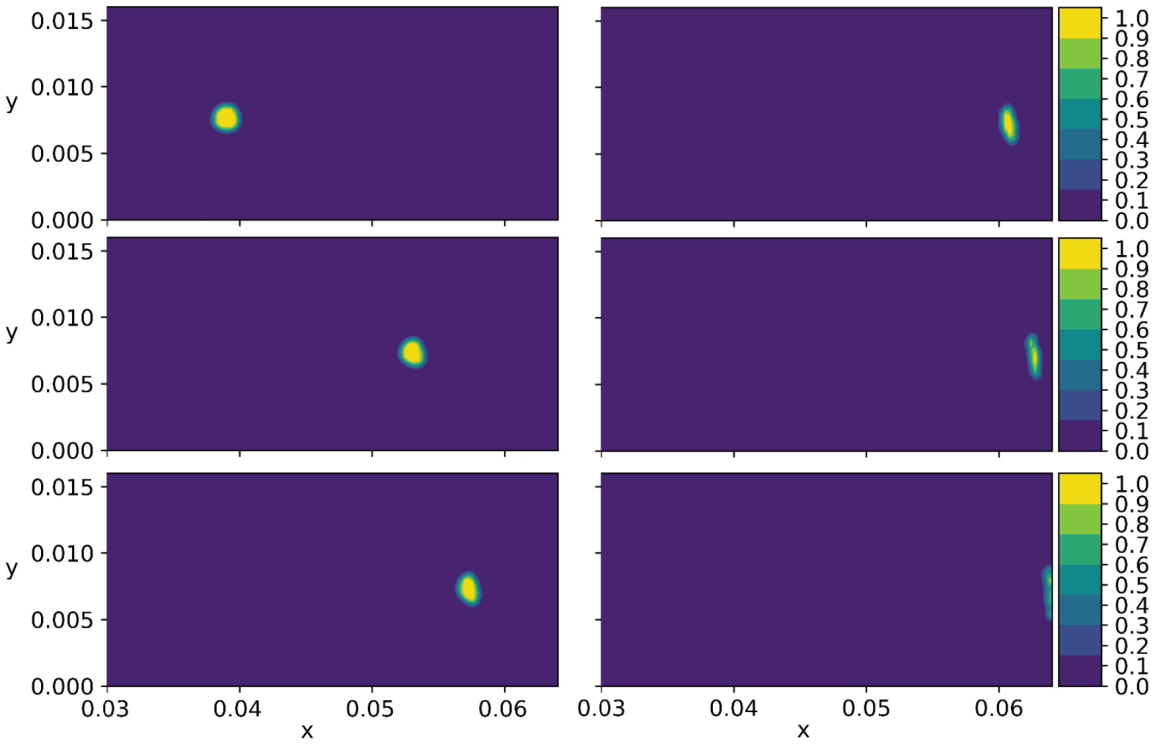


Figure 1.7: CFD Droplet Morphology [12]

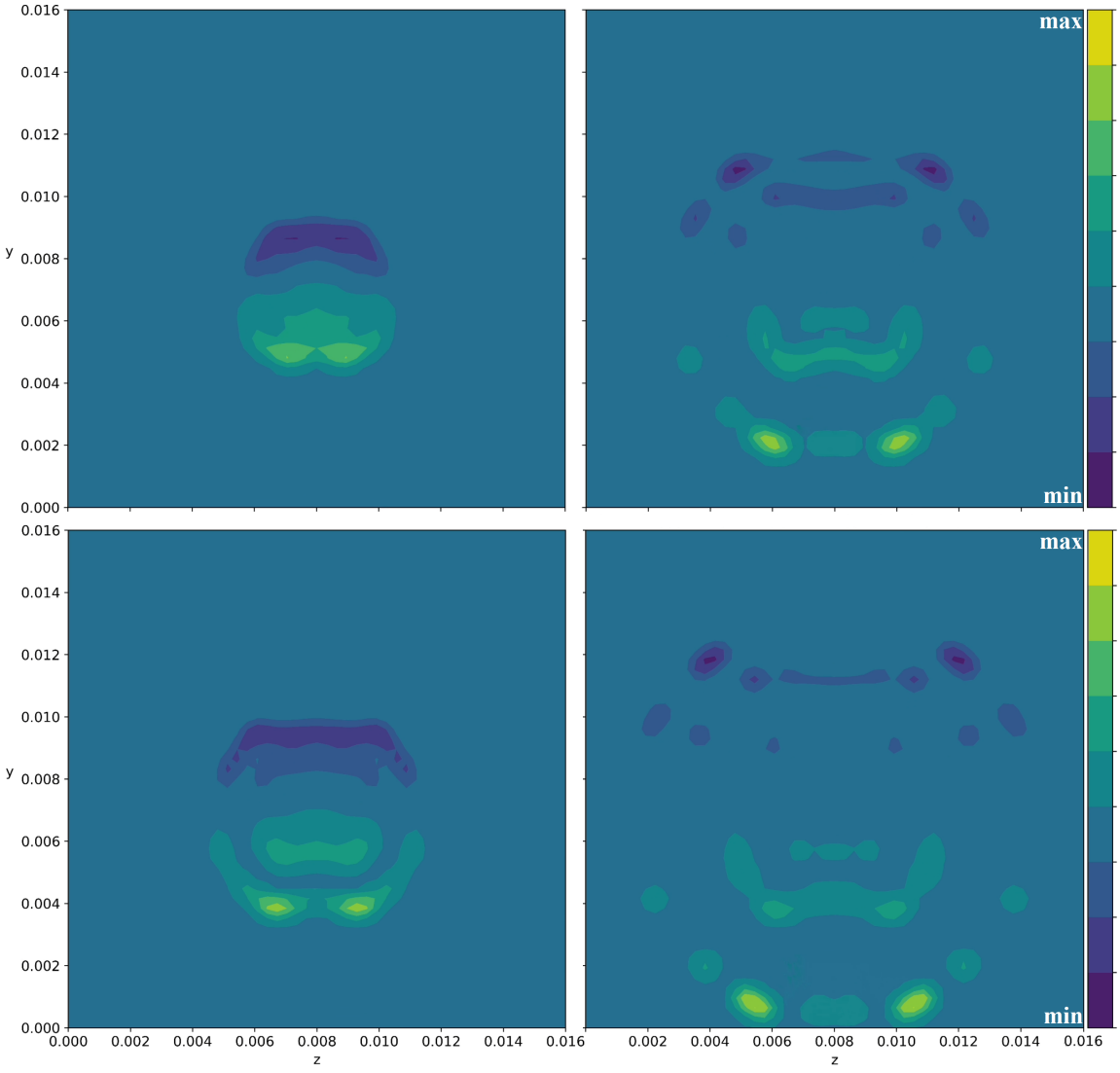


Figure 1.8: CFD Surface Shear [12]

Chapter 2

EXPERIMENTATION AND THEORY

2.1 Tensile Testing

In its simplest form, tensile testing involves stretching a material sample to a specific load or displacement under known or intentional conditions. Static tensile tests usually load a sample to a limit load, ultimate load, or failure. They are often either load or displacement controlled and may take place in a variety of controlled environments. Some testing machines even have built in chambers for temperature control. Reasons for conducting static tension tests include:

- New material characterization
- Test part or material performance in various environments, such as very hot or cold.
- Part qualification
- Quality assurance

In this thesis, we will focus on new material characterization, specifically that of thin polymeric films. The tensile test generates a curve of applied force versus crosshead displacement. From here, the curve is transformed into a stress versus strain curve where the stress, σ, is calculated from the force, P, and the area, A, per equation 2.1 and the strain is calculated per 2.2 where L_0 is the initial length of the sample and ΔL is change is the crosshead displacement. For true stress A is the instantaneous area, while in the case of engineering stress, A is the original area. This is usually sufficient for metals but thin film polymers often require the use of DIC to account for localized strain. We will have more on that in section 2.1.4.

$$\sigma = \frac{P}{A} \tag{2.1}$$
$$\sigma^T = \frac{P}{A_i};\ \sigma^E = \frac{P}{A_0}$$

$$\epsilon^E = \frac{\Delta L}{L_0} \tag{2.2}$$

All of the tensile tests discussed herein were performed at Boeing on a TA-Discovery HR-2 hybrid rheometer using the tensile test attachment. The sample size limitations of this testing apparatus are 40 mm in length and 15 mm in width.

2.1.1 Sample Preparation Overview

In order to perform the tensile tests, sample needed to be prepared. This occurred in a lab at Boeing. All tensile samples for each material were cut from a larger coating of thin film which had been painted on a polyblock without adhering making them easy to lift off after the samples were cut. Once cut out, each sample's thickness was measured with an isoscope at several points across the specimen. The first samples were simply cut out rectangles and subsequent samples used ASTM standard geometry. Early samples relied solely on the machine's crosshead displacement to calculate strain while later samples made use of Sharpie marks and a microscope which evolved into full-fledged DIC.

2.1.2 The First Samples

In the first iteration of tensile testing, samples were rectangles cut from the polyblock using a metal guide and razor knife. Their primary purpose was to get a first look at how the materials behaved. In Figure 2.1, we see a Primer and Topcoat specimen loaded into the test machine. Notice how the Primer specimen is about twice as wide as the Topcoat. This is because the Primer is so thin and brittle that smaller samples would either crumble during preparation or fail too quickly during testing. Putting numbers to it, the Topcoat and Clearcoat samples were 6.35 mm thick while the Primer samples were 12.5 mm and the Basecoat was not tested with rectangular samples.

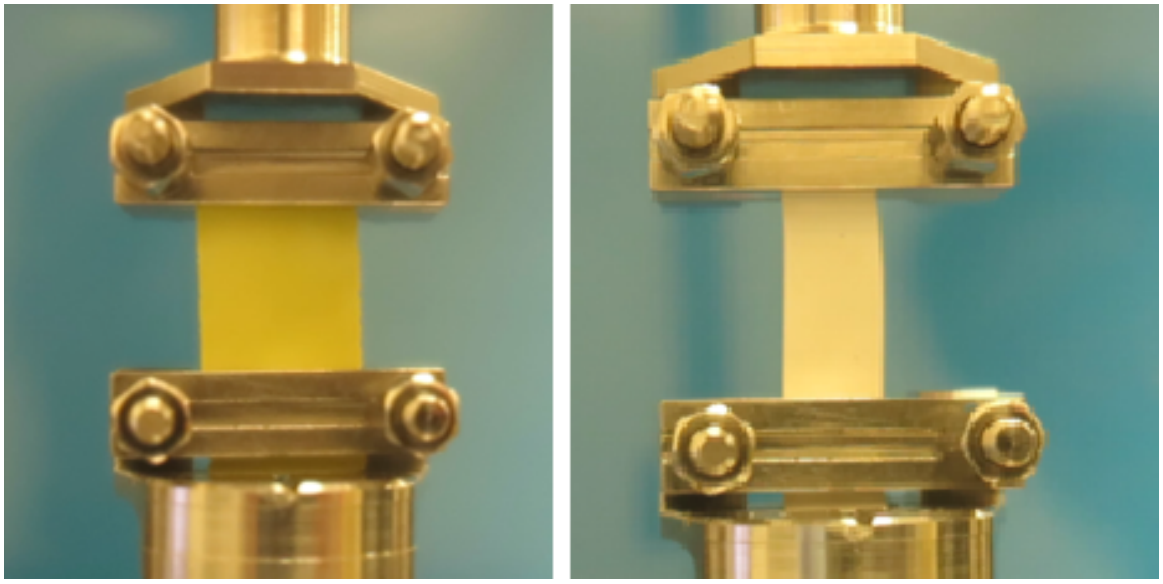


Figure 2.1: Initial Rectangular Specimens of the Primer (Left) and Topcoat (Right)

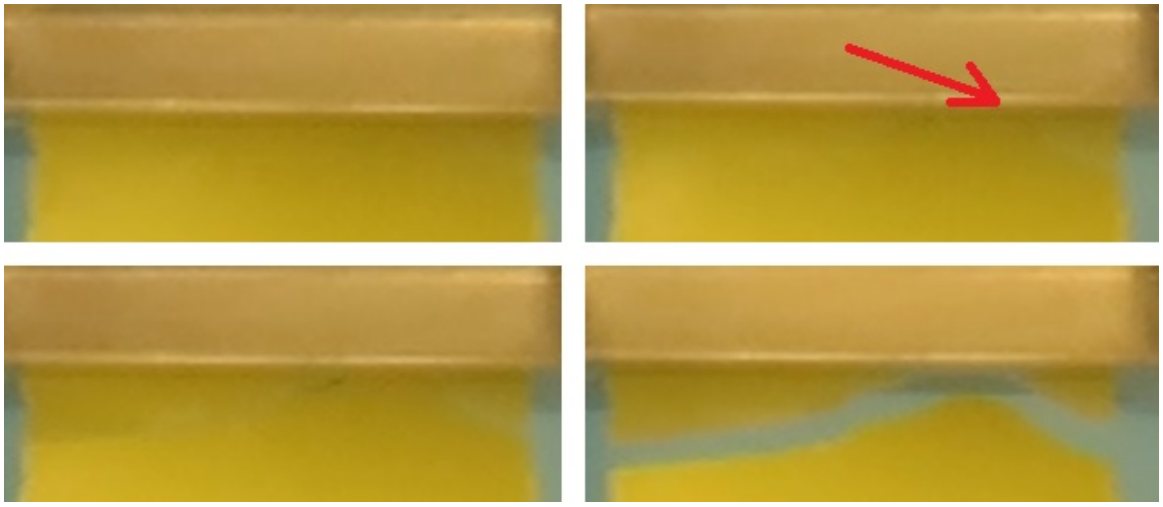


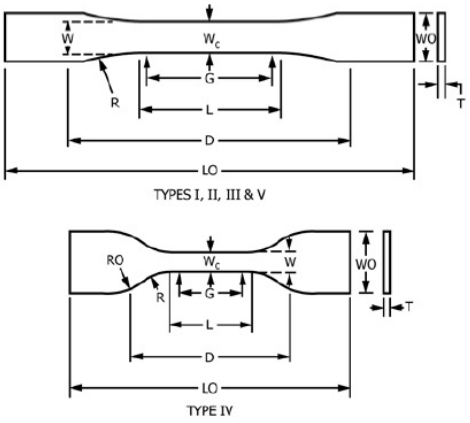
Figure 2.2: Failure at Specimen Grip (Primer)

One major problem of using rectangular samples is that they could fail anywhere as the cross-sectional area is close to constant throughout. Another issue that stems from this is the possibility of stress concentrations where the grips make contact. With thin film samples,

it is important not to over tighten the grips and crush the specimen and thus, a Mountz MT120-N torque tool was used to tighten the nuts seen in Figure 2.1 to 50 cN-m $\pm 1.0\%$. In the case of rectangular samples, this crushing at the grips has greater influence than with a dogbone type specimen and so these specimens would sometimes fail prematurely at the grips. An example of such failure may be seen with the primer in Figure 2.2. In this case, we see a crack that either initiated at the grip itself or at the side of the sample and propagated toward a stress concentration at the grip. Realizing that rectangular cutouts would not cut the cake, it was decided to use dogbone type samples.

2.1.3 Dogbone Samples

The next set of samples were prepared to ASTM standard D638, *Standard Test Method for Tensile Properties of Plastics* [13]. This standard offers a few options for dogbone shapes which we see in Figure 2.3. At this point in time, no one had thought of determining the height limitations of the HR-2 and it was unspokenly assumed that samples per D638 would fit. Of these potential samples, Type III was quickly ruled out because none of the samples would be thicker than 7 mm. Type V was subsequently ruled out because the width of the narrow section sparked concerns over the Primer which was already brittle and having trouble with crumbling apart during production or failing when trying to set a width of 6.35 mm. This left Types I, II, and IV. CAD models of the remaining types were made to be 3D printed and used as cutting templates. Ultimately, Type IV was chosen for testing as Boeing already had a cutting die on hand. Nonetheless, the template was still used because, in addition to being dogbones, these samples differed from the first samples with the introduction of several dots across the gauge length. These dots were created by holding the template in place over a die-cut sample and poking a fine tipped Sharpie into each of the through holes (see Figure 2.4) in the template. The purpose of the dots was to check for significant amounts of localized strain with a technique that could be called extremely rudimentary DIC and will be referred to as 'dot checking.' Regretfully, none of the samples per ASTM D638 would make it to testing as a huge problem made itself aware upon the first attempts: the samples did not fit in the test machine. They were too long and had to



Specimen Dimensions for Thickness, *T*, mm (in.)[A]

Dimensions (see drawings)	7 (0.28) or under		Over 7 to 14 (0.28 to 0.55), incl	4 (0.16) or under		Tolerances
	Type I	Type II	Type III	Type IV[B]	Type V[C,D]	
W—Width of narrow section[E,F]	13 (0.50)	6 (0.25)	19 (0.75)	6 (0.25)	3.18 (0.125)	±0.5 (±0.02)[B,C]
L—Length of narrow section	57 (2.25)	57 (2.25)	57 (2.25)	33 (1.30)	9.53 (0.375)	±0.5 (±0.02)[C]
WO—Width overall, min[G]	19 (0.75)	19 (0.75)	29 (1.13)	19 (0.75)	...	+ 6.4 (+ 0.25)
WO—Width overall, min[G]	...	...	...	...	9.53 (0.375)	+ 3.18 (+ 0.125)
LO—Length overall, min[H]	165 (6.5)	183 (7.2)	246 (9.7)	115 (4.5)	63.5 (2.5)	no max (no max)
G—Gage length[I]	50 (2.00)	50 (2.00)	50 (2.00)	...	7.62 (0.300)	±0.25 (±0.010)[C]
G—Gage length[I]	...	...	...	25 (1.00)	...	±0.13 (±0.005)
D—Distance between grips	115 (4.5)	135 (5.3)	115 (4.5)	65 (2.5)[J]	25.4 (1.0)	±5 (±0.2)
R—Radius of fillet	76 (3.00)	76 (3.00)	76 (3.00)	14 (0.56)	12.7 (0.5)	±1 (±0.04)[C]
RO—Outer radius (Type IV)	...	...	...	25 (1.00)	...	±1 (±0.04)

Figure 2.3: Tensile Test Specimens per ASTM D638 [13]

undergo significant folding or cutting in order to fit.

With the troubles of using ASTM D638, other options had to be considered. That's when ASTM D1708, *Standard Test Method for Tensile Properties of Plastics by Use of Microtensile Specimens*, was discovered [14]. The size of the tensile specimen per ASTM D1708 (illustrated in Figure 2.5) is almost perfect and fits in the HR-2 without necessarily requiring sample modification. (Initially, these samples were trimmed to a maximum width of 11.75 *mm* which was later deemed unnecessary. Since this trimming did not show any evidence of effecting the test outcome, it will not be further discussed.) Now, having a sample size that fits in the testing machine, a cutting and dotting template was made similar to those in Figure 2.4 but for D1708 instead. Dot checking involves measuring the spacing between dots under a Keyence VHX digital microscope as shown in Figure 2.6 before testing to establish a baseline. These measurements include the dot heights, sample width at several points, and axial distance of each side of each dot from a datum in the center of the sample.

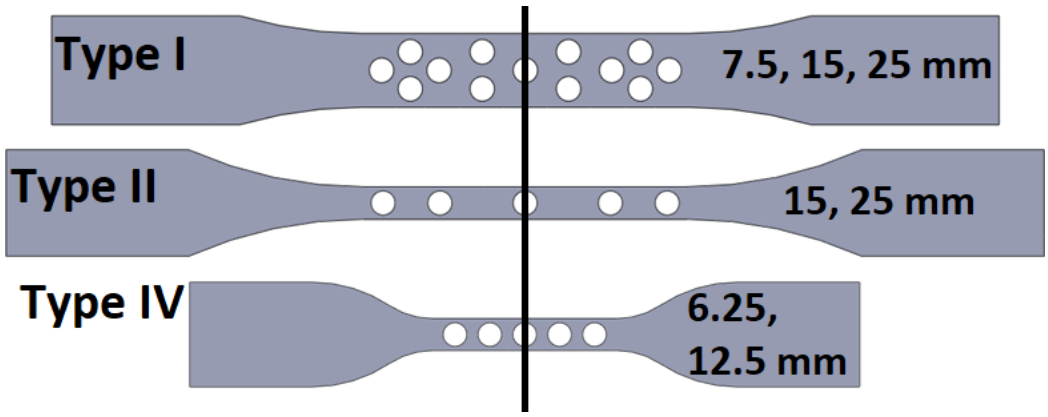


Figure 2.4: Cutting Templates for ASTM D638 Tensile Specimens

Then, the test is videotaped and a software image tracking program such as ImageJ is used to determine the relative displacements of the dots as well as the change in sample width to calculate Poisson's ratio.

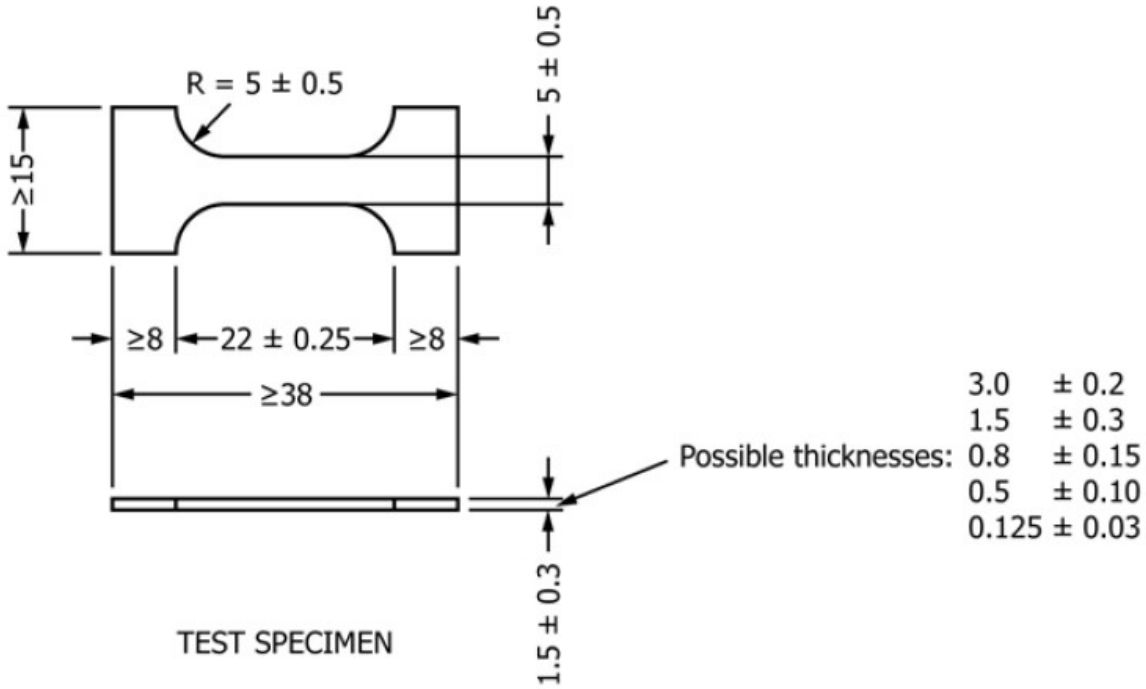


Figure 2.5: Microtensile Test Specimen per ASTM D1708 [14]

In addition to showing the measurements, Figure 2.6 details the cut edges of the sample. Here, we notice that there are several stress risers present that could lead to premature failure. Thankfully, this specific sample failed in the middle of the reduced section in an apparently natural manner. However, this was not the case for all samples as detailed in section 4.2.1. With the specimen shape per D1708 proven to fit in the test machine, an order was put in for a die. In the mean time, testing continued to the other materials using template cut specimens and a DIC apparatus was setup.

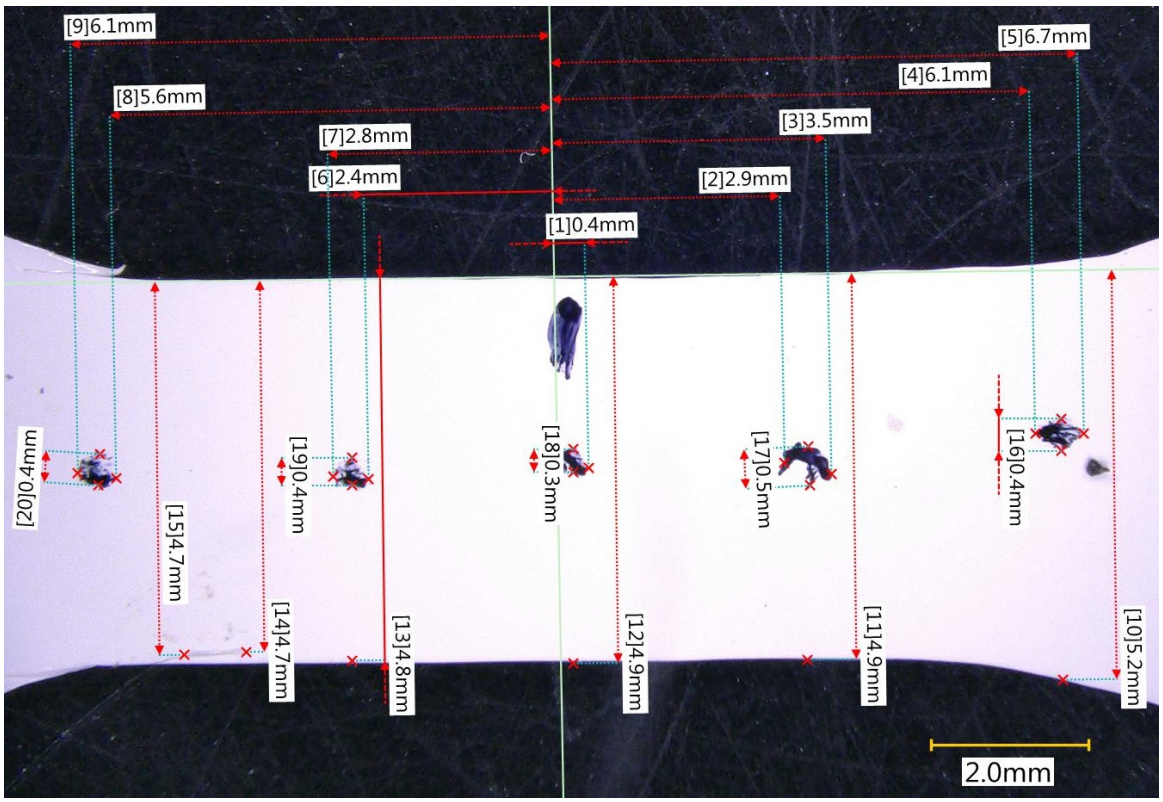


Figure 2.6: Dotted Test Specimen Made from Cutting Die

Tests completed on the TA Discovery HR-2 output data including load (as measured by the machine's transducer) and displacement (measured by the machine's crosshead motion) at discrete time steps. For these pre-DIC samples, this data was processed using equations 2.1 and 2.2 to obtain engineering stress-strain curves. With the engineering curves, true

stress and strain at each point is found with equations 2.4 and 2.7 below [15].

$$\epsilon^T = \int_{L_0}^{L_i} \frac{dL}{L_i} = ln\left(\frac{L_i}{L_0}\right) \tag{2.3}$$

Considering $L_i = L_0 + \Delta L$ and equation 2.2, we have:

$$\epsilon^T = ln(1 + \epsilon^E) \tag{2.4}$$

Introducing the assumption that the volume of the sample is conserved, we may calculate the true stress from the true strain:

$$A_0 L_0 = A_i L_i = C; \ \textit{where} \ \ C \in \mathbb{R}^+$$
$$\Rightarrow \frac{A_0}{A_i} = \frac{L_i}{L_0}$$

combining this result with equations 2.3 and 2.4 we find:

$$\frac{A_0}{A_i} = 1 + \epsilon^E \tag{2.5}$$

Now, we may draw from equation 2.1 two useful relationships:

$$\sigma^T = \frac{P}{A_i} \tag{2.1}$$
$$P = \sigma^E \cdot A_0$$
$$\Rightarrow \sigma^T = \sigma^E \cdot \frac{A_0}{A_i} \tag{2.6}$$
$$\therefore \sigma^T = \sigma^E \cdot (1 + \epsilon^E) \tag{2.7}$$

It is important to understand the underlying assumptions in the above equations. First, they assume volume conserving deformation. Such an assumption is reasonable for small deformations as the change in cross-sectional area will be close to insignificant. However, this is seldom the case with any material as volume conservation means the material possesses a Poisson's ratio of $\nu = 0.5$. As such, these equations are only approximations which is acceptable for basic analysis but, in the scope of this project, the true plastic behaviour is vital in modelling rain erosion. Thus, the off-axis strain will need to be measured directly. Secondly, the materials are assumed to be perfectly isotropic which is reasonable for thin film coatings as paint is essentially a random formulation of so many discrete points that a

continuous isotropic sheet is formed. Third of all, equations 2.5, 2.6, and 2.7 are only valid under uniform strains. That is to say that any localized strain or necking invalidates these equations and the section of the true curves after the onset of localized strain. Finally, it is inherently assumed that the machine's compliance has a negligible impact on the strain measurements. This may or may not be true, but DIC will eliminate this assumption entirely.

The use of DIC will mitigate the problems that arise from these assumptions. The true stress and true strain may be accurately described by measuring strain across the width of the sample in addition to along the loading axis. Furthermore, DIC tracks hundreds of discrete points thereby enabling it to capture the existence and effects of localized strain.

2.1.4 DIC

Digital image correlation is a technique used to precisely map deformation fields over a sample during testing and is used in a variety of manners. In this case, DIC is used to measure the strain of a specimen during tensile testing. The process begins by applying a high contrast field, or 'speckling' to the sample of interest like that shown in Figure 2.7. Typically, the field is created using paint or ink on the surface of a sample. However, this is not an option for thin film paint specimens as DIC medium because adhering paint to the paint sample could affect the material properties of the samples and invalidate the test. As a solution to this obstacle, printer toner was used to speckle the samples. With this, the printer toner simply rests on the sample; held in place by surface tension, not adherence, which allows the high contrast field to be used for DIC without changing the behaviour of the sample.

Two 5-megapixel cameras placed near-normal to the sample with overlapping fields of view, similar to human eyes, can identify and track the points with perspective. For the tests performed, the frame rate of these cameras varied between approximately four and twelve frames per second depending on the material and strain rate being tested. More brittle materials and/or higher strain rates call for quicker frame rates. Using the cameras; the DIC testing apparatus, Zeiss GOM Aramis, is able to measure strain in the x- and y-directions as well as displacements in x, y, and z [16].

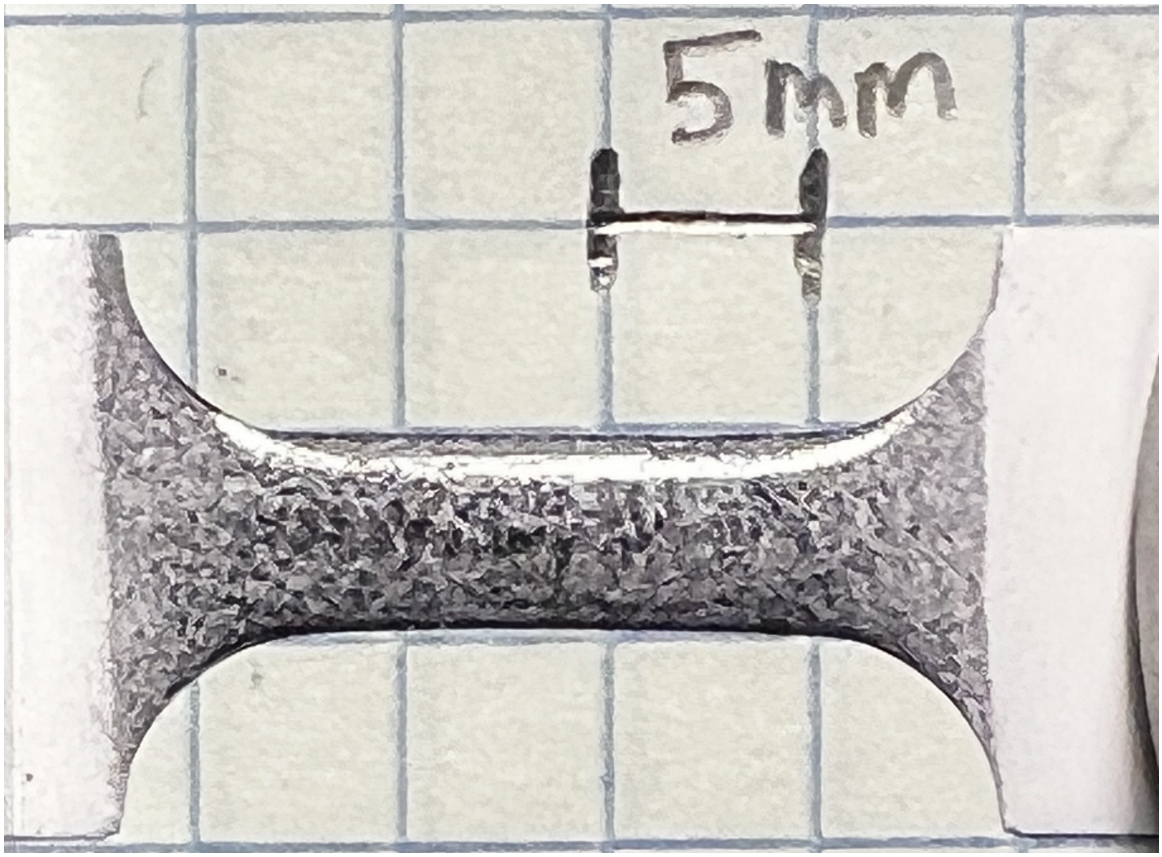


Figure 2.7: Tensile Specimen with a High Contrast Field for DIC

As the tests were run, three bands of contrast point data were analyzed and saved as shown Figure 2.8. One band is just inside the reduced area section on each end while the third band is located in the middle of the sample. This yields multiple ways to calculate the stress and strain of the sample:

- **Method A** uses the data from the center band exclusively and assumes strain in the x and z directions to be the same. This assumption stems from the thin film paints being assumed as isotropic and homogeneous. With Method A, no strain calculations are necessary as GOM Correlate Pro outputs the x and y strain directly.

- **Method B** is the same as Method A except for the fact that it does not assume the strain in x and z to be equivalent but rather, Method B calculates the strain in z using

the average deflection in z across the center band and initial thickness with equation 2.8.

- **Method C** is also very similar to Method A with the only difference being that it uses all three bands and returns the average values.

- **Method D** does not use any of the strains calculated by GOM correlate. Instead, Method D uses the deflections in x and y between the upper and lower bands to calculate the strain across the sample with equation 2.12 for x and 2.13 for y. There are three sub-methods within Method D based on the x strain calculation. Method D_C Calculates x-direction strain based on the center strip of contrast points, Method D_{TB} uses the top and bottom strips to calculate x strain, and Method D_{CTB} uses all three. Method D_{CTB} allows the user to vary weights on the center strain versus the top and bottom strain components of the total x strain. Top and bottom are weighted together while center is weighted separately. with this, a weight on the center x strain of $W_{DC} = \frac{1}{3}$ paired with a top and bottom weight of $W_{DTB} = \frac{2}{3}$ yields an even influence from each of the top, center, and bottom strips.

- **Method E** is a derivative of Method D, Method D_C specifically. The difference between Methods E and D is that, rather than using the average coordinate point to calculate y strain, Method E calculates the y strain between several pairs of coaxial points where the line they define is parallel to the loading axis. Once these strains are computed, the script evaluates and reports the average values for further analysis. The math between Methods D and E is identical, the difference is the order of operations. As such, Method E applies equation 2.14 to several points and then determines the average. The utility of using both methods is to check for oddities in the DIC results. If they are not nearly identical, then one or more of the points could be reading incorrectly.

- **Method F** is to Method C as Method B is to Method A. This method uses the average strains calculated by GOM Correlate Pro in x- and y- directions at the top, middle,

and bottom bands. It also determines the strain in z the same way as Method B (equation 2.8) at the top, middle, and bottom bands. Then, the average value of these three calculated z strains is used at each step.

The equations for stress and strain within the MATLAB script evaluate true strain and true stress at several time steps from testing. In the following equations, the subscript j denotes the j^{th} time step. All of these equations start from a datum of zero strain and zero load. In some cases, the strain needs to be shifted by a small amount to have the same datum as the load. The reason the initial datums for strain and load are not always at the same time step is because the TA Discovery HR-2 outputs load while the GOM Aramis apparatus outputs DIC and hence strain. The resulting time steps do not necessarily align with one another and so they are synchronized from failure backward. The load goes to zero and the DIC cameras capture the fracture when failure occurs. With this, the time at failure is set equal between the test machine and DIC cameras at failure and the corresponding time in the DIC videos is determined by way of subtracting the test machine's time steps from the time at failure until we reach zero time. Both apparatuses have discrete clocks so, there is potential for minor error which is corrected by shifting the strain. With this, we may move onto strain calculations. For Methods B and F, we have z strain:

$$\epsilon^T_{zz,j} = \epsilon^T_{zz,j-1} + ln\left(\frac{t_j}{t_{j-1}}\right) \tag{2.8}$$

where the instantaneous thickness may be determined by

$$dz_j = 2 \cdot (z_j - t_0) \tag{2.9}$$

$$t_j = t_{j-1} - 2 \cdot (dz_j - dz_{j-1}) \tag{2.10}$$

here, dz_j is a measure of total change in z from start to time step j and z_j is the average z-coordinate. The factor of 2 comes in because the cameras only track motion on one side of the sample and it is assumed that both side deform the same amount. Method B only uses the center strip for this analysis while Method F uses all three.

For the methods that do not use GOM's x and y strains but rather calculate them from

coordinates, we have the following equations. These build off of equation 2.3.

$$\epsilon_{yy,j}^{T,D} = \epsilon_{yy,j-1}^{T,D} + ln\left(\frac{Ygap_j^D}{Ygap_{j-1}^D}\right) \tag{2.11}$$

$$\textit{with: } Ygap_j^D = y_{mt,j} - y_{mb,j} \text{ for Method D}$$

$$\textit{or: } Ygap_j^D = y_{pt,j} - y_{pb,j} \text{ for Method E}$$

$$\textit{and}$$

$$\epsilon_{xx,j}^{T,D} = \epsilon_{xx,j-1}^{T,D} + ln\left(\frac{Xgap_j^D}{Xgap_{j-1}^D}\right) \tag{2.12}$$

$$\textit{with: } Xgap_j^D = x_{max_{band_k},j} - x_{max_{band_k},j}$$

where the y_{mt} is the mean y-coordinate of the top band and y_{mb} is the mean y-coordinate of the bottom band. Similarly, y_{pt} is the specific y-coordinate of interest in the top band and y_{pb} is its mate in the bottom band. While $x_{max_{band_k}}$ is the maximum x-coordinate of the k^{th} (top, middle or bottom) band and $x_{min_{band_k}}$ is the corresponding minimum coordinate. Equation 2.11 then simplifies to:

$$\epsilon_{yy,j}^{T,D} = \epsilon_{yy,j-1}^{T,D} + ln\left(\frac{(y_{mt} - y_{mb})_j}{(y_{mt} - y_{mb})_{j-1}}\right) \tag{2.13}$$

for Method D and

$$\epsilon_{yy,j}^{T,D} = \epsilon_{yy,j-1}^{T,D} + ln\left(\frac{(y_{pt} - y_{pb})_j}{(y_{pt} - y_{pb})_{j-1}}\right) \tag{2.14}$$

for Method E.

Calculating the stresses in the sample is a little simpler than the strains although strains are needed in order to calculate the instantaneous area of the sample. True stress is found per equation 2.1 where we may directly move to a true stress calculation by using our true strains from the DIC data. The instantaneous area is found using the true strains per equations 2.15 through 2.20 where x_0 and z_0 are the initial dimensions of the sample's narrowest cross section as shown in Figure 2.9.

$$A_0 = x_0 \cdot z_0 \tag{2.15}$$

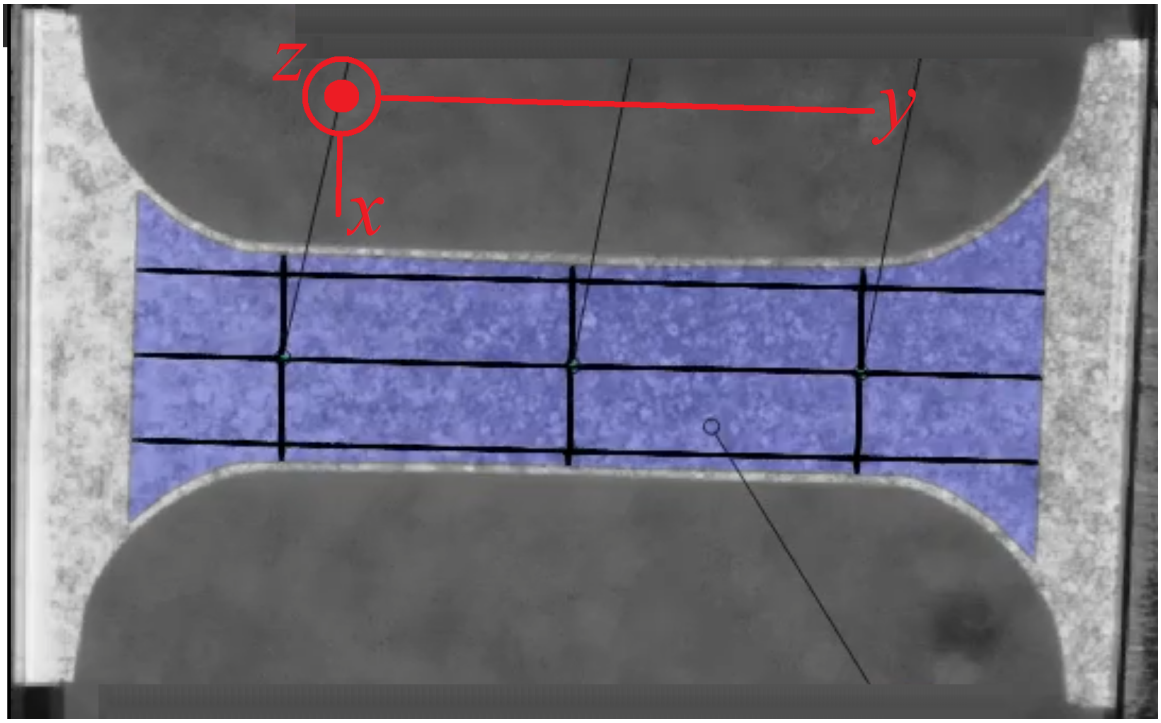


Figure 2.8: Setup and Unloaded DIC Tensile Specimen with Bands

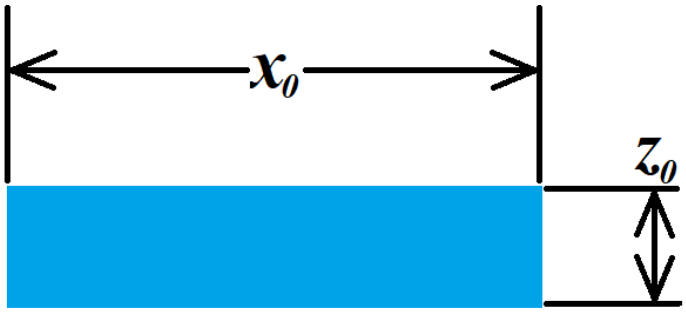


Figure 2.9: Initial Cross-sectional Area

$$
\begin{aligned}
dA &= xdz + zdx \\
\frac{dA}{A_i} &= \frac{dx}{x} + \frac{dz}{z} \\
\int_{A_0}^{A_i} \frac{dA}{A_i} &= \int_{x_0}^{x} \frac{dx}{x} + \int_{z_0}^{z} \frac{dz}{z}
\end{aligned} \tag{2.16}
$$

Assuming a homogeneous cross section, we may express the right-hand side of equation 2.16 as true strains in the x- and z- directions by leveraging equation 2.3. Thus:

$$
\begin{aligned}
\ln \frac{A_i}{A_0} &= \epsilon_{xx}^T + \epsilon_{zz}^T \\
A_i &= A_0 \cdot e^{(\epsilon_{xx}^T + \epsilon_{zz}^T)}
\end{aligned} \tag{2.17}
$$

Then having instantaneous area, we may calculate true stress at time step j per equations 2.1 and 2.17. Superscript M denotes any method while α denotes specific methods.

$$\sigma_j^{T,M} = \frac{P_j}{A_j^M} \tag{2.18}$$

$$A_j^\alpha = A_0 \cdot e^{2\epsilon_{xx,j}^{T,\alpha}} \; for \; \alpha = A, C, D_C, D_{TB}, E \tag{2.19}$$

$$A_j^\alpha = A_0 \cdot e^{\epsilon_{xx,j}^{T,\alpha} + \epsilon_{zz,j}^{T,\alpha}} \; for \; \alpha = B, F \tag{2.20}$$

$$\tag{2.21}$$

With true stress-strain curves, we are able to determine several useful material properties including Young's modulus, E; Poisson's ratio, ν; plastic Poisson's ratio, ν_{pl} (can be a single value or curve); yield stress, σ_y; true ultimate stress, σ_u; and true strain at failure, ϵ_f. σ_u and ϵ_f are rather simple and may be determine simply by looking at a stress versus strain chart as illustrated in Figure. Others are relatively straightforward and may be calculated using the equations below:

$$\nu = \frac{-\epsilon_{xx}^T}{\epsilon_{yy}^T} \text{ in the Elastic Regime} \tag{2.22}$$

$$\nu_{pl} = \frac{-\epsilon_{xx}^T}{\epsilon_{yy}^T} \text{ in the Plastic Regime} \tag{2.23}$$

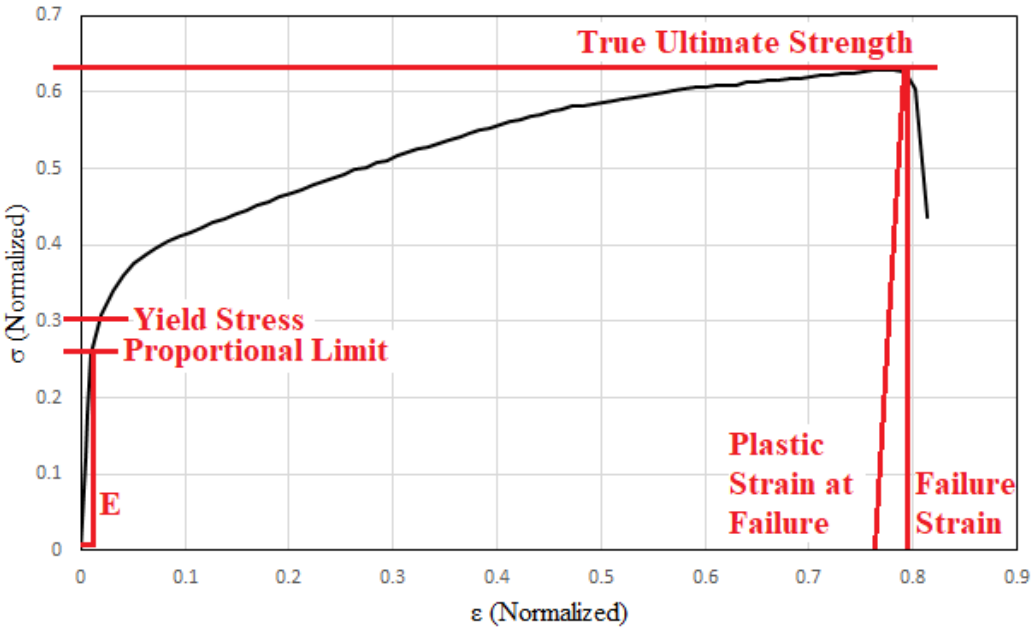


Figure 2.10: Some Useful Properties from the σ^T- ϵ^T Curve

Finally, some of these properties require a little more analysis to derive and understand. We'll begin with Young's modulus and the proportional limit. Young's modulus is defined as the slope of the linear-elastic region of a stress-strain graph. That is, the section below the proportional limit, σ_{prop}.

$$E = \frac{\sigma_{prop}}{\epsilon_{prop}} \tag{2.24}$$

To determine the proportional limit, an automated iterative process is used. Each iteration looks at the slope between the first recorded test point (ϵ, σ) from the datum plus an offset if deemed necessary $(0 \leq \mathit{offset} \in \mathbb{Z})$ and a point n steps away. Next, a linear regression is implemented through MATLAB to determine a line of best fit between the starting point and point of interest (inclusive of points between) and the coefficient of determination, R^2, for the fit line is compared to a critical value until $R^2 \leq R^2_{crit}$ [17]. If equal to, the upper bound of this line is the proportional limit and the slope is Young's modulus. If less than, the $n-1$ case is used.

An alternative method for determining Young's modulus is to use Excel as an E-Finder tool. With this method, the stress strain curve is plotted in Excel and a straight line is overlaid. The slope of the straight line is changed until it visually matches the linear section of the test. Here, the point where the curve and E-Finder line diverge is the proportional limit and the slope of the straight line is Young's modulus. This method is useful when there are not enough data points to determine the modulus accurately using the statistical method. This method is also a useful tool for intuitive validation of the statistical method.

The next property to calculate for our material model is the yield stress. For modelling and analysis purposes, the 0.2% offset yield stress is used. It's calculated by means of finding the intersection between two curves. The first curve is the stress-strain curve itself and the second is a line parallel to the linear region of the stress-strain curve and offset by 0.002 mm/mm. The MATLAB function 'polyxpoly' is used to determine the intersection point between these curves and this value is used (at least in part) as the yield strength of the material in FEM (see Section 3.1.2) [17].

The MATLAB code used for stress-strain curves by Methods A-F and material property determination is available in Appendix B.1.

2.2 Nanoindentation Testing

Most materials undergo uniaxial compression testing to generate compressive stress-strain curves from which a plastic curve in compression may be drawn in the same process used for uniaxial tension testing. However, this sort of test is not possible with thin film coupons such as the paints of interest due to buckling. With this in mind, an alternative approach is required: nanoindentation testing.

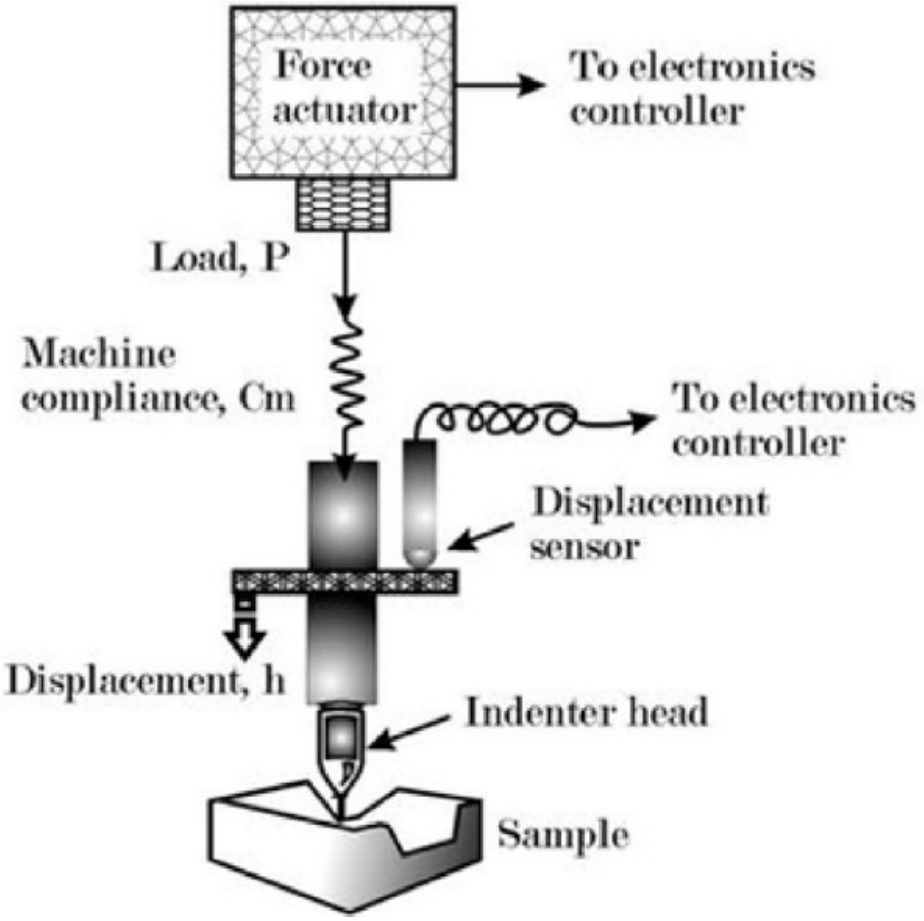


Figure 2.11: General Schematic of a Nanoindentation Test Machine [18]

A nanoindentation coupon consists of a substrate, made of aluminum in this case, to which the sample material is adhered. The testing machine has a sample bed on which the coupon is placed and an actuator that harbours the indenter tip. During the test, the tip is sunk into the sample where either the indentation depth, applied force, or both are on the nanoscale; hence the name of *nano*indentation. The test outputs a curve of load versus displacement including the loading, hold time, and unloading of the sample. Figure 2.11 illustrates the general layout of a nanoindentation test machine's components. From the figure, it is worth noting that machine compliance as well as the stiffness of the indenter tip will have a small impact on the test results. Another feature to consider is the blunting

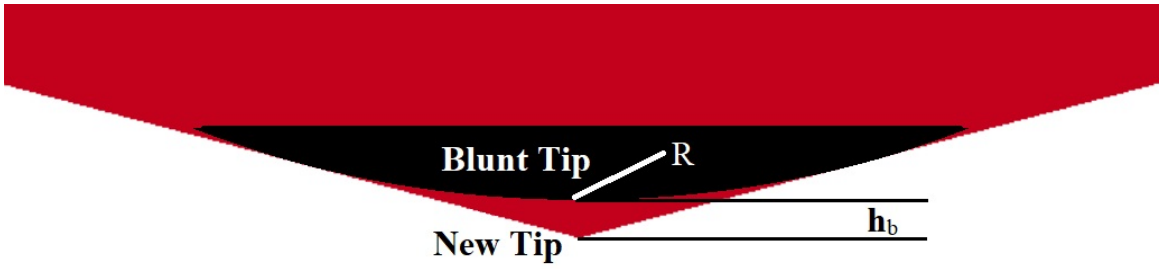


Figure 2.12: Visual Representation of Indenter Tip Blunting

of the indenter tip. As more and more tests are performed, the tip of the nanoindenter begins to blunt. This occurs even in diamond tips as the blunting is on the nanoscale itself. It has been shown in literature that the tip blunting effect is insignificant past depths of approximately 200 nm with regard to the shape of the load-displacement curve and contact area, [19], [20]. On the other hand, the indenter tip radius has a more appreciable effect on hardness and elastic modulus calculations than the raw curve. Nonetheless, this impact is most significant at smaller depths [21]. Furthermore, a sample test was performed to validate the lack of tip radius influence on the deeper portion of load-displacement curves which is described in Section 3.2.3.

Intuitively speaking, it is reasonable that the tip radius would have a smaller influence on the outcome of a nanoindentation test as the depth increases. As we can see in Figure 2.12, the contact area at depths of $h >> h_b$ is hardly influenced by the tip radius.

For the work presented herein, nanoindentation tests were performed at the University of Washington on either a Bruker Hysitron TI-980 (Topcoat and Primer) or Ubi-1 (Basecoat and Clearcoat) machine. Each sample was indented with a diamond Berkovich indenter tip which is a three sided self-similar pyramid shaped indenter with a total included angle of 142.3°. Berkovich indenter tips generally begin their lives with a tip radius around 100 nm and can be blunted to as much as $1,000$ nm although blunting is more commonly between 200-400 nm based on various literature [20], [21], [22]. One major goal of the nanoindentation tests is to obtain the Young's modulus and the plastic behaviour of each material in compression. The elastic modulus may be easily determined as

$$E_{comp} = \frac{1-\nu^2}{\left(\frac{1}{E_r} - \frac{1-\nu_I^2}{E_I}\right)} \tag{2.25}$$

where the subscript, I denotes properties of the indenter tip [19], [23].

But first, we need to know the reduced modulus, E_r, the sample's Poisson ratio, ν, and the indenter tip's stiffness and Poisson ratio. The former is output by the test machine which determines E_r with equation 2.26 [23].

$$E_r = \frac{S}{2}\sqrt{\frac{\pi}{A_c}} \tag{2.26}$$

$$\text{and } A_c = \pi \cdot tan^2\theta h_c^2 \tag{2.27}$$

Here, S is the slope of the unloading curve at maximum load as shown in Figure 2.13 and is known as the contact stiffness while A_c is the contact area. Poisson's ratio is determined for the sample by way of equation 2.22 from the tensile results. Lastly, the indenter tip properties are known for a given indenter.

It is important to mention that equation 2.27 assumes a perfectly pyramid shaped intenter with θ as the semi-angle at the tip apex and h_c as the contact depth. The Bruker machines at the University of Washington make use of the Oliver and Pharr ([23], [24]) model for tip bluntness which defines the contact depth as a function of indentation depth, h, applied load, P, contact stiffness, S, and geometric constant, ε [19], [23]. For a Berkovich or other pyramid shaped indenter $\varepsilon = 0.75$ [19].

$$h_c = h - \varepsilon\frac{P}{S} \tag{2.28}$$

As described, E_r and therefore A_c, h_c, and S are calculated by the test machine along with the sample's hardness ($H = \frac{P}{A_c}$) [23]. In order to determine these values, the machine keeps the intender tip geometry calibrated with the Oliver and Pharr ([23], [24]) tip area function of equation 2.29 rather than equation 2.27 to account for tip blunting and defects. In equation 2.29, C_0 through C_8 are constants that arise from curve fitting procedures [24].

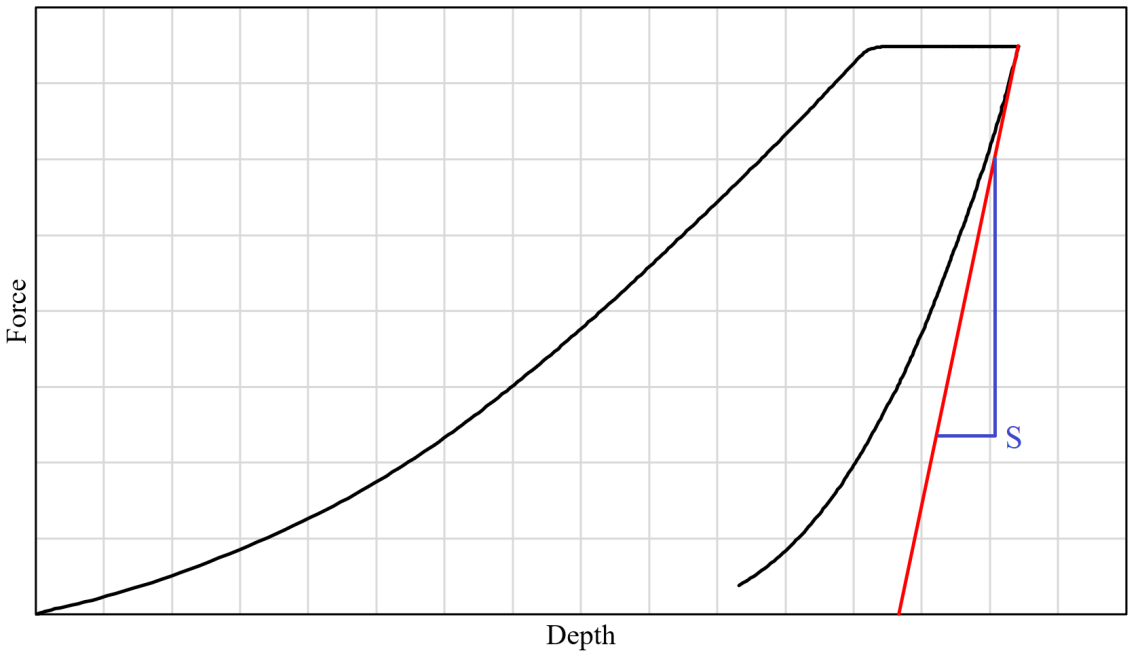


Figure 2.13: Example of Nanoindentation L-D Curve with Contact Stiffness

$$A_c = \sum_{n=0}^{8} C_n \cdot h_c^{2-n} = C_0 h^2 + C_1 h + C_2 h^{1/2} + ... + C_8 h^{1/128} \tag{2.29}$$

In the case of a perfect Berkovich indenter, only C_0 is needed and equation 2.29 will agree with equation 2.27 [24].

Now, with the theory behind the machine's calculations explained, we can move on to post-test processing. In addition to Young's modulus in compression (equation 2.25), we need to determine the plastic behaviour of the material. This is performed with curve matching techniques detailed in Section 3.2.3. In short, this involves recreating the nanoindentation tests with FEM models of the sample and indenter tip. Within the sample's material model, a curve of stress versus plastic strain is iterated until the resulting load displacement curve matches that of the physical test.

2.3 Lap Shear Testing

In the characterization of thin film coating systems, it is crucial to understand the behaviour of the interface between coating layers. Lap shear testing is a useful step toward obtaining this important understanding as it directly reveals the shearing strength of an interface. This mimics failure from Mode II (see Figure 2.14) crack propagation. Similarly, the lap shear test opens the door for an iterative curve matching process. Depending on the damage model of choice, this curve match allows the determination of critical distance for a tiebreak failure or the shear energy release rate of the failure. Another important use of lap shear testing is to predict whether a coating system is more likely to fail adhesively or cohesively. The difference between these failure modes is that adhesive failure is a delamination at the boundary between a coating with another coating or a coating with the substrate while a cohesive failure occurs in the material itself without evidence of delamination. This difference is depicted in Figure 2.15.

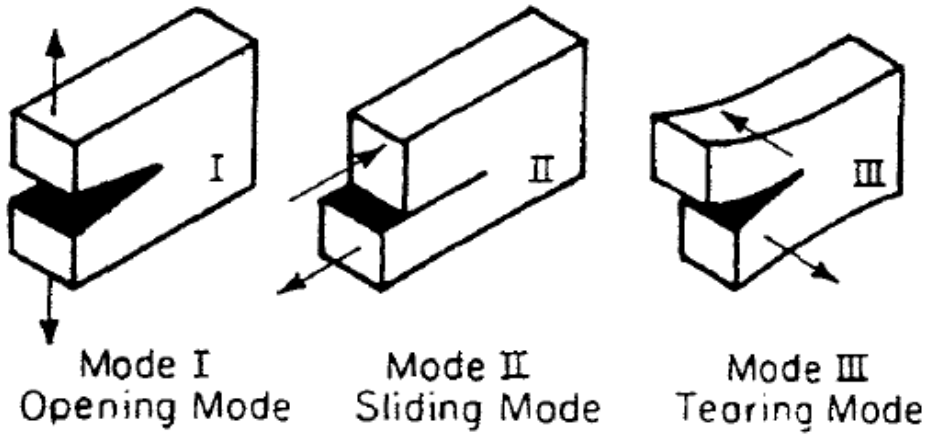


Figure 2.14: Crack Propagation Modes [15]

Similar to tensile testing, the lap shear test is a pull test. The tests discussed in this thesis were performed at Boeing on an Instron 5565 test machine. Sample preparation began with blank aluminum coupons about 0.050" thick. They were all painted with the Primer. At this time, half of the coupons were stuck together in a fixture with a predetermined gap

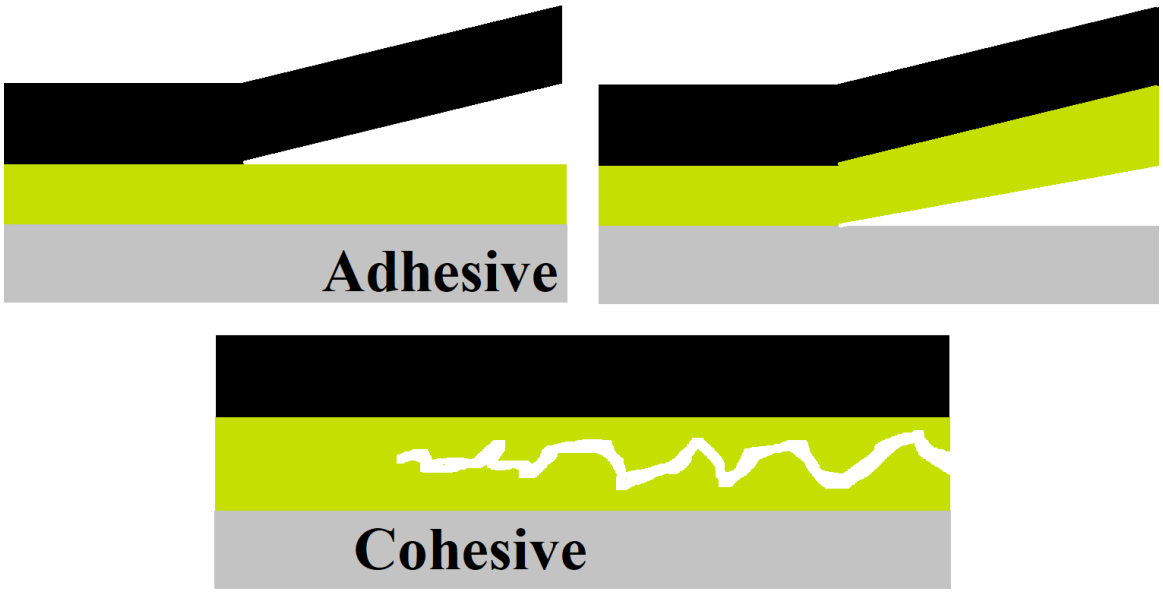


Figure 2.15: Adhesive versus Cohesive Failure

between the aluminum substrates and left to cure. The other half were allowed time to dry and next, they were painted with the Topcoat. Then, these samples were placed in a fixture and held to cure with a specific gap between the substrates. The resulting coupon configurations are shown in Figures 2.16 and 2.17.

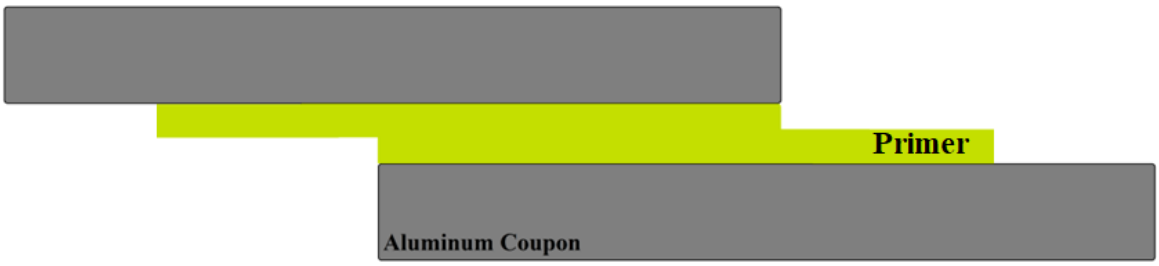


Figure 2.16: Primer Lap Shear Coupon

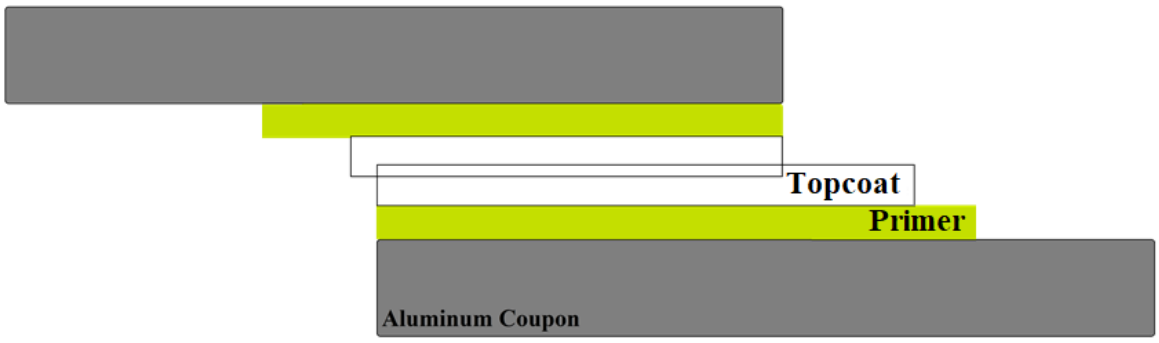


Figure 2.17: Topcoat over Primer Lap Shear Coupon

During the lap shear test, samples are gripped in the jaws at a distance of 1 in or 25.4 mm from the end of the overlapping substrate. As the machine's jaws are clamped to the substrates, they pull and apply a small preload to the system. Next, the machine's crosshead begins to move and load the sample to failure generating a load-displacement curve. With the curve, a stress-deflection curve may be found with equation 2.30.

$$\tau = \frac{P}{A_{shear}} \tag{2.30}$$

where l is the length of the bonded sample in the loading direction and A_{shear} is the shearing area. In these tests, the deformations are sufficiently small such that engineering stress and strain may be used without the need for true stress and strain calculations.

Chapter 3

MODELLING

3.1 Wind Turbine Blade Model

The first modelling performed in the structures side of this project, is a validation and comparison of Verma, Castro, Jiang, and Teuwen's work modelling the rain erosion of wind turbine blades (see reference [3]) which was discussed in Chapter 1.5.

3.1.1 Convergence Study

We begin with a discussion on remodelling Verma et al.'s convergence study. For this, Verma and associates clearly outlined the element size, basic types, and controls used although some details were not included such as the spacing of SPH particles within the droplet model. The aluminum plate convergence model was created as a near carbon copy of Verma's model which we can see in Figure 3.1. Both plates have a fine mesh in the center of 0.04×0.04 mm elements which enlarges to 0.4×0.04 mm elements and 0.4×0.4 mm elements on the outer sides and corners, respectively. The plate is 20 mm square by 1 mm tall and modelled as aluminum using the *MAT_PIECEWISE_LINEAR_PLASTICITY keyword in LS Dyna. The density, Young's modulus, Poisson's ratio, and the yield stress were all taken from Verma's paper (see Table 1.1) [3]. A value was not provided from Verma et al. to use as the tangent modulus, although a value for shear modulus was provided. In this case, the tangent modulus is irrelevant because plastic deformation of the aluminum plate was not a concern. The droplet in this recreation was modelled as 2.7 mm with an impact velocity of 2.67 m/s which one can intuitively determine will not cause high stresses in the aluminum plate. This was later verified after running the simulation to find a maximum stress below 1 MPa. In addition to its size, the droplet was modelled using between 47,000 and $158,000$ particles for the convergence study and 137,000 once converged. The SPH particles were distributed evenly throughout the sphere of the droplet.

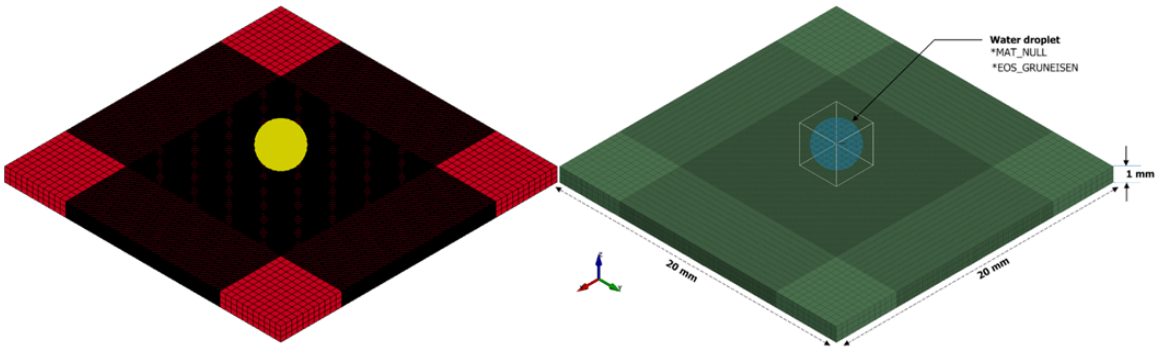


Figure 3.1: Author's Recreation (Left) of Verma's Aluminum Plate (Right) [3]

3.1.2 Gelcoat Model

The Gelcoat model is the recreation of the full four layer model from Verma and associates as shown in Figure 1.4. The paper did not disclose as much information about this model as with the convergence model hence, some of the properties and relationships had to be calculated, approximated, or assumed. Save for the raindrop, each part was modelled with constant stress (ELFORM = 1) solid elements per the paper. From the top down, we begin with the rain droplet. The droplet was modelled using 137,000 SPH particles based on the mesh convergence which aligned with Verma's. Since the purpose of this recreation is for methodological validation in LS-Dyna, it was deemed unnecessary to recreate every scenario the paper used. Thus, the droplet was modelled with a 2.74 mm diameter, 107.6 m/s velocity, and 90° impact angle. The Verma paper provided critical details to model the rain drop with *MAT_NULL and *EOS_GRUNEISEN cards (Table 1.3.)

Modelling the gelcoat layer is when a few obstacles made themselves clear. First, this mesh was not quite the same as in the aluminum plate convergence model. To mitigate this issue, Figure 1.4 was printed out and the relative distances were measured with a pair of Mitutoyo calipers. The size of the elements was then calculated with by

$$\frac{l_{1,print}}{l_{1,model}} = \frac{l_{2,print}}{l_{2,model}} \tag{3.1}$$

after which the larger element size was verified as 0.2022 mm by counting while the smaller was assumed to be correct by the equation as it matched the 0.04 mm element size of the

convergence model. The final dimension, thickness, was given by Verma et al. as 0.3 mm for the gelcoat and each layer was confirmed to be the same thickness with the caliper method.

The next step in modelling the gelcoat layer was the material model. From Table 1.1, we have density, stiffness, Poisson's ratio, and yield strengths. The gelcoat's material card, *MAT_PLASTICITY_COMPRESSION_TENSION, also intakes load curves of effective yield stress versus plastic strain in both tension and compression as well as scaling curves to describe the strain rate effect. Such curves were given by Verma and company per Figures 3.2 and 3.3 [3]. Since LS-Dyna requires curves defined by data points, the plot digitizer, *Web Plot Digitizer* was used to obtain .csv files for each curve at the rate of $10^{-1}s^{-1}$ as well as the both scaling curves [25]. The resulting curves were then input into the LS-Dyna material model.

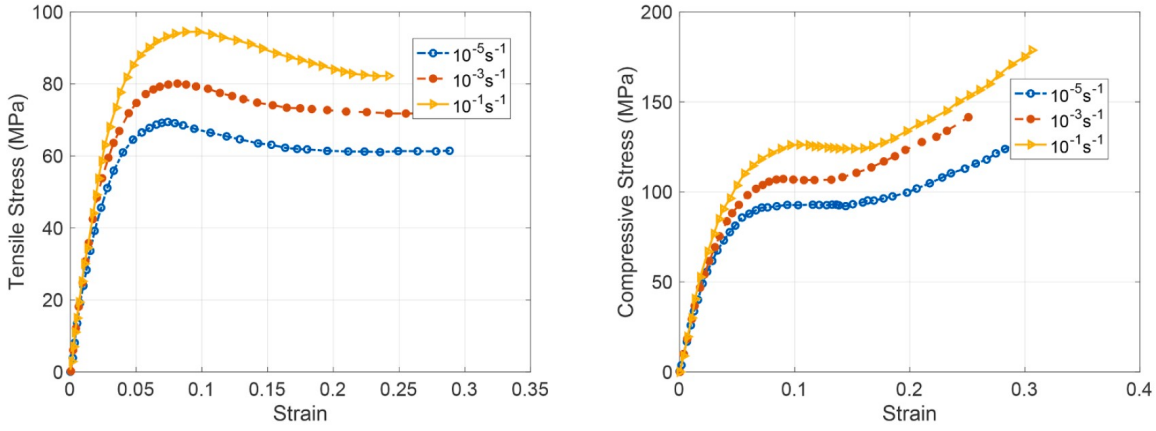


Figure 3.2: Gelcoat Tension (Left) and Compression (Right) σ-ϵ Curve [3]

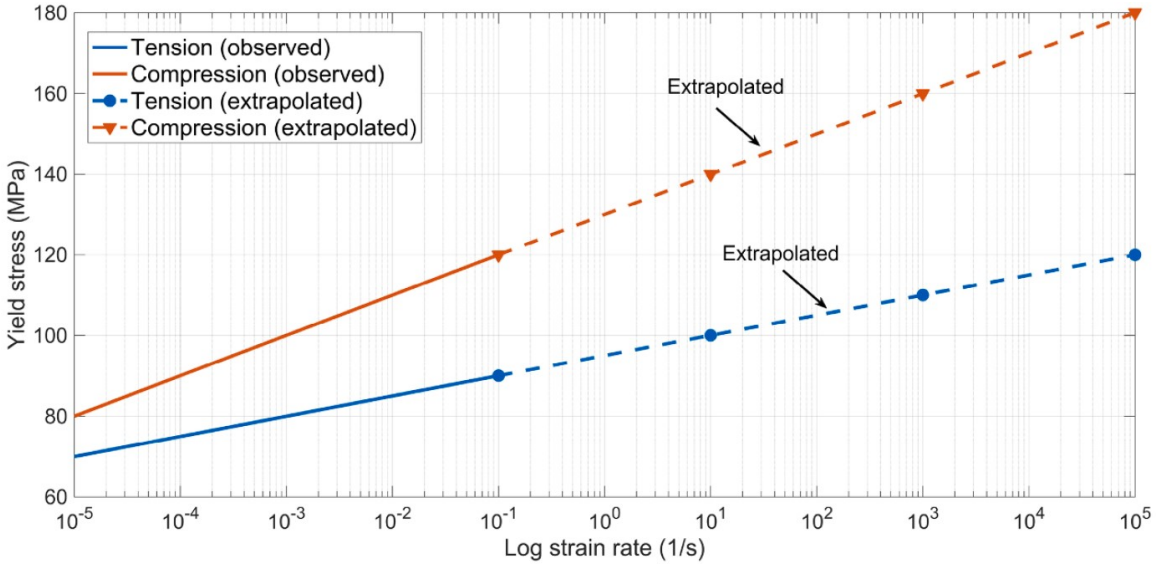


Figure 3.3: Strain Rate Effect on the Gelcoat [3]

The chopped strand mat was fairly straightforward with the mesh already figured out. All of the material parameters were given (see Table 1.1) along with the material model: *MAT_PLASTIC_KINEMATIC.

Finally, we have GFRP. The +45 and -45 degree plies of this material were individually modelled with the *MAT_COMPOSITE_FAILURE_SOLID_MODEL keyword for which critical parameters were given and are tabulated in Tables 1.1 and 1.2 while other parameters were left as the LS-Dyna defaults [3].

With all materials modelled, the next step was to define contacts. As with Verma and associates, the contact keywords *CONTACT_AUTOMATIC_NODES_TO_SURFACE and *CONTACT_AUTOMATIC_SURFACE_TO_SURFACE_TIEBREAK were invoked. Here, the former describes the contact between the droplet and gelcoat layer while the latter was implemented to describe the cohesive contacts between each layer. The droplet to gelcoat contact made use of soft penalty constrain formulation with a scaling factor of 0.25. The soft constrain computes the interface stiffness based on nodal mass and the global time step [11]. In this contact, the droplet was set as the slave with the gelcoat as the master. No data for friction was given and related inputs were left as default. The remaining three

contacts took the top surface as the slave and bottom as the master with OPTION = 6. This option activates tiebreak for nodes initially in contact and acts as a surface to surface contact after failure [11]. Failure can be defined by the normal and shear interface failure stresses, *NFLS* and *SFLS* were only given for GFRP. For the purpose of this model, it was deemed acceptable to use these values for all three tiebreak contacts. These contacts left friction and other non-given parameters as default.

Lastly, this model incorporated an hourglass control with an hourglass coefficient of 0.12 and uses standard LS-Dyna viscous form. The addition of the hourglass control is part of the Verma recreation and acts to avoid zero-energy deformation [3]. In addition to the hourglass, the *CONTROL_TIMESTEP card was included with a time step scaling factor of TSSFAC = 0.4 [3]. With the model ready for analysis, it should be noted that only the single impact case was recreated.

3.2 Material Validation and Test Recreation

3.2.1 Material Model

The tensile material profile for each material is built from the tensile stress strain results as described in Section 2.1. The elastic behaviour in compression is determined by calculating Young's modulus per equation 2.25 while the plastic compressive profile is found as part of the recreation detailed in Section 3.2.3. There are two material cards of interest for the thin films in this research: *MAT_SAMP-1 (also referenced as *MAT_187) and *MAT_PLASTICITY_COMPRESSION_TENSION (*MAT_124). Their primary functionalities as applicable to this work are summarized in Table 3.1. Both models have other capabilities, this table only includes those most relevant to the materials and service conditions of this thesis.

*MAT_187	*MAT_124
Able to consider plastic behaviour in tension, compression, and shear.	Able to consider plastic behaviour in tension and compression.
Plastic behaviour is input as load curves of effective yield stress versus plastic strain.	Plastic behaviour is input as load curves of effective yield stress versus plastic strain.
Strain rate dependence in accounted for by using tables of load curves where each curve corresponds to a different strain rate.	Strain rate dependence is accounted for by scaling or with the Cowper-Symonds relationship 3.2 [26].
Able to consider strain rate dependence of Young's modulus as a load curve.	Able to consider two independent values for Young's modulus: one in compression and one in tension.
Intakes load curve defining plastic strain at failure as a function of yield stress.	Intakes load curve defining plastic strain at failure as a function of yield stress.
Able to consider plastic Poisson effects	
Able to consider damage as a function of equivalent plastic deformation.	

Table 3.1: SAMP-1 versus Plasticity Compression Tension

Both of these materials are viable candidates and should be able to capture the behaviour of any of the four materials. A few advantages to using *MAT_187 are:

- With limited testing, tables describing plastic strain as a function of yield stress at select strain rates are more useful in the beginning than using empirical or scaled relationships.

- Once sufficient testing has taken place, the tables can be extrapolated to encompass higher strain rates using empirical relationships such as Cowper-Symonds or scaling.

- This material model's ability to consider plastic Poisson effects will be useful in a fatigue study.

Similarly, *MAT_124 has advantages of its own:

- It is easier to incorporate difference in Young's modulus in tension versus compression with *MAT_124 than with *MAT_187 so long as it does not vary significantly at different strain rates.

- Verma et al. have already demonstrated this material card's ability to model rain erosion.

- Cowper-Symond parameters (C and P) may be directly fed into the model rather than having to extrapolate them in tables which implies smaller storage requirements for *MAT_124.

$$\sigma = \sigma_0(\epsilon)\left[1 + \left(\frac{\dot{\epsilon}}{C}\right)^{1/P}\right] \tag{3.2}$$

3.2.2 Tensile Recreation

Building a robust material model requires a validation of the tension tests performed. Tensile recreation models are full 3D solid models using constant stress solid elements (ELFORM = 1) as inspired by Verma and associates [3]. The simulation is meant to recreate the actual test as precisely as reasonable. To achieve this, the validation coupon is modelled after ASTM D1708 just like the actual test samples and the thickness of coupon depends on the material being tested.

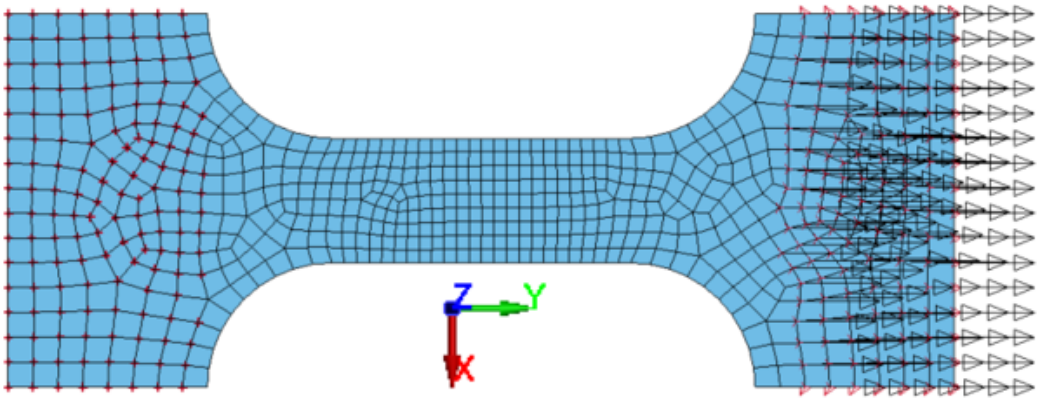


Figure 3.4: Tensile Validation Coupon (Dimensions per ASTM D1708 [14])

As we can see in Figure 3.4, nodes at the top (right in figure) are prescribed motion in the positive y direction. This motion is prescribed either as displacements from a physical test, or as a velocity to mimic the actual test. Similarly, the lower grip zone of the sample is fixed in all degrees of freedom. From here, the velocity or displacement is varied based on which strain rate is being recreated and the simulation is allowed to run. Being such a simple model, this is able to run relatively quickly on a personal computer.

Mesh Convergence

As an important part of model validation, the mesh to be used must undergo a convergence study. For the tensile coupons, the mesh convergence study looked specifically at the mesh in the reduced area section of the sample. These elements varied in size from 1.0 to 0.1 mm and it was determined that 0.5 mm elements adequately captured the tensile behaviour.

3.2.3 Nanoindentation Recreation

The nanoindentation recreation model serves not only as a validation of the material profiles by comparison with physical tests but also is used for the determination of the material's plastic behaviour through iteration. This section begins with an overview of the general nanoindentation recreation model and proceeds to detail the processes used in plastic curve determination.

Type	Material	ρ $(tonnes/mm^3)$	$E(GPa)$	ν	Radius (Assumed)
Berkovich	Diamond	3.51×10^{-18}	1140	0.07	100 nm

Table 3.2: Indenter Tip Properties

Model Overview

Each nanoindentation recreation model consists of an indenter tip and a sample where the bottom of the sample is fixed in all degrees of freedom. The tip is modelled after the actual Berkovich tip used in testing and its relevant properties are outlined in Table 3.2. A perfectly sharp indenter tip from Mo A on grabcad was brought into SolidWorks and shortened to just the tip [27]. Then, in LS-Dyna, a 100 nm radius was applied to all three edges to create the geometry. In the LS-Dyna Model, the indenter tip is modelled as a rigid body (*MAT_RIGID keyword in LS-Dyna) comprised of Belytschko-Tsay (ELFORM = 2) elements auto-meshed at 30 nm.

The modelling for the sample was inspired by Verma et al. [3]. It is modelled with constant stress solid elements (ELFORM = 1), as was Verma's model of the gelcoat, and has a biased mesh. About the point of indentation, elements are 50×50 nm ant step up to 1000×1000 nm, then to 1500 nm squared, followed by squares of 2,000, 3,000, 7,000, and 12,000 nm. These outermost elements likely have no impact on the results as the load is focused at the center. The boarders for the fine elements in the center were determined by the reach of the outer fringe. Initially set large, the area was shrunk to hold the displacement fringe as shown Figure 3.5. This figure likewise details the fine in-plane mesh toward the center of the sample. Similarly, we may observe the vertical mesh biasing in Figure 3.6. It is important to note here that the large aspect ratios toward the bottom of the sample do not impact the model as the applied strains are extremely localized. The model also makes use of the same type of hourglass control as used by Verma et al.

The indenter tip and material sample contact each other with the *AUTOMATIC_SURFACE_TO_SURFACE keyword where the sample is surface A and the indenter is surface B. In addition to this primary contact card, the *CONTACT_ERODING_SURFACE_TO_

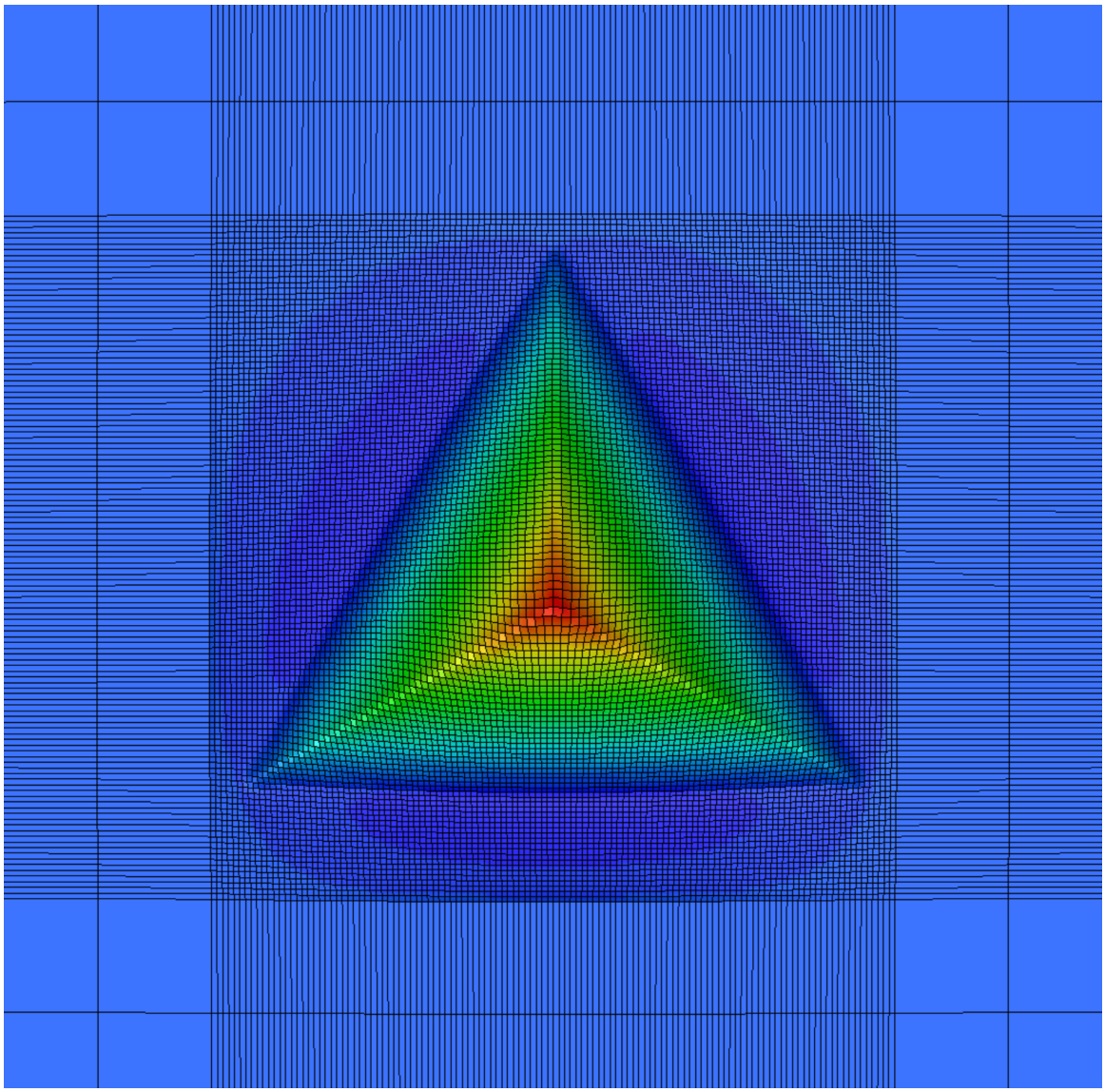


Figure 3.5: Displacement Fringe of Nanoindentation Recreation

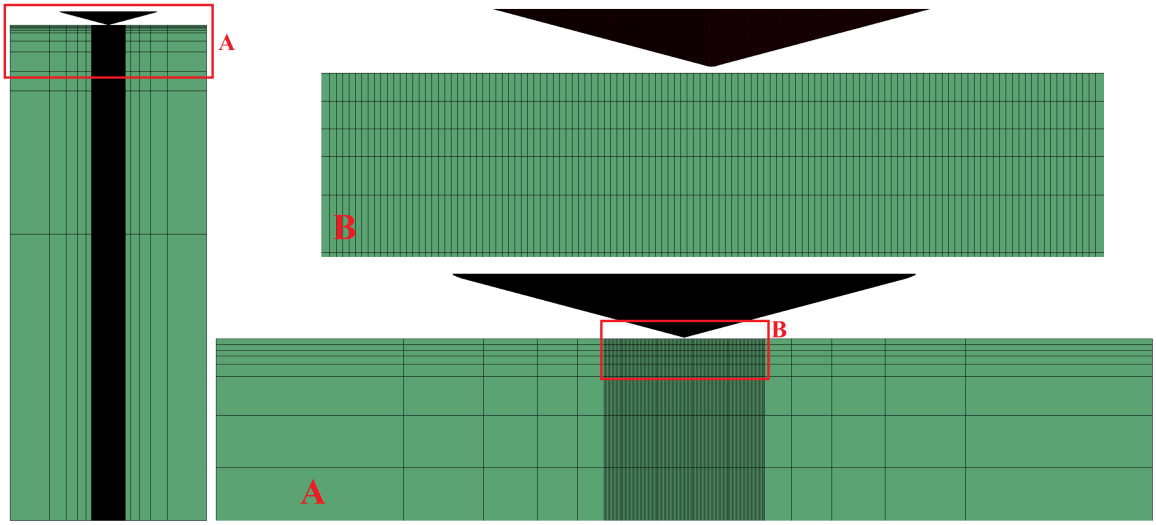


Figure 3.6: Displacement Fringe of Nanoindentation Recreation

SURFACE keyword is applied in this model to ensure that if any elements erode due to negative volume, the newly exposed elements beneath them will not penetrate the indenter tip. Lastly, these contacts were modelled as frictionless which was initially an assumption based intuition that was later verified with FEM when static and dynamics coefficients of friction were applied as $\mu_s = 0.5,\ \mu_d = 0.4$, and the resulting contact force did not change a meaningful amount as we can see by the lighter gray plot in Figure 3.7.

Mesh Convergence

As with the tensile test validation, a mesh convergence study was performed. For the simulation of nanoindentation tests, the convergence study was run on the element size in the plane normal to the loading action of the tip. Specifically, the force at the beginning of hold time was compared for elements sized from $500 \times 500\ nm$ down to $10 \times 10\ nm$. This was run with the Topcoat as the material and the force was found to converge at 50×50 nm elements.

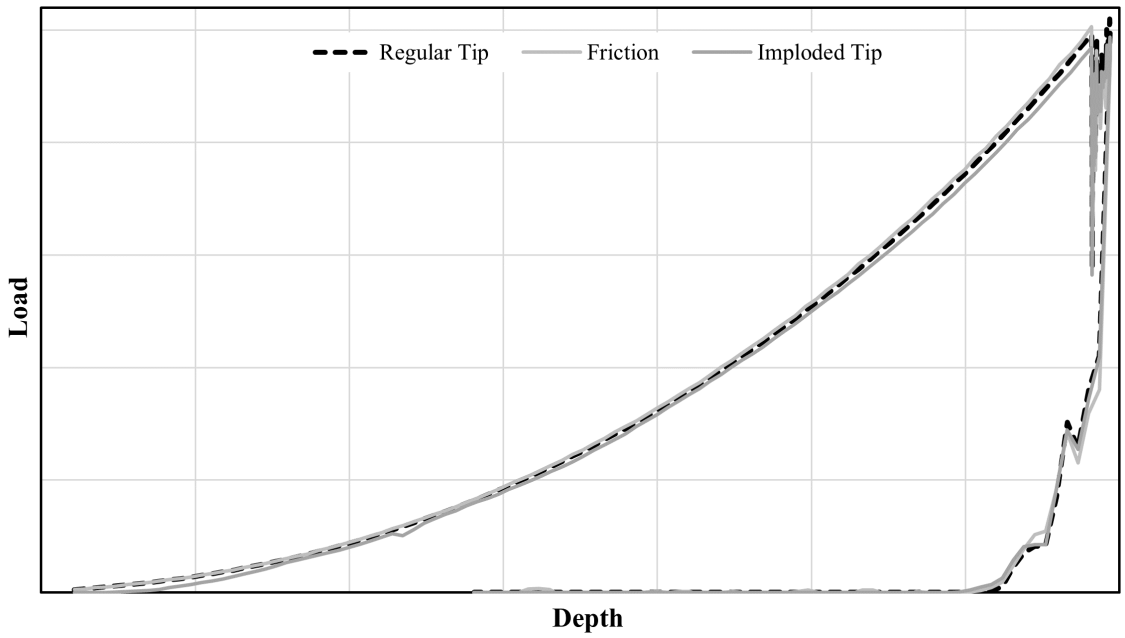


Figure 3.7: Friction and Imploded Tip Check on Nanoindentation Model

Effect of the Tip Radius

The tip underwent several iterations of radius before it was eventually deemed that the tip radius was irrelevant for purposes of deriving the plastic curve of the Topcoat, Primer, and Clearcoat. This was determined based on the indentation depth of these materials being substantially more than the depth shown in literature where tip radius has a paramount effect on contact area. This was then confirmed with an imploded tip model. As we can see in Figure 3.8, the imploded tip model takes the regular nanoindentation recreation model with a 100 nm tip radius and displaces several elements at and around the tip to the inside of the tip at some odd angle effectively creating a hole in the tip with an angled plane behind it. These elements were first sent backward (+y) by 100 nm before being translated in the +x and +z directions. Then, going back to Figure 3.7 and looking at the darker gray plot, we see no significant difference from the regular model.

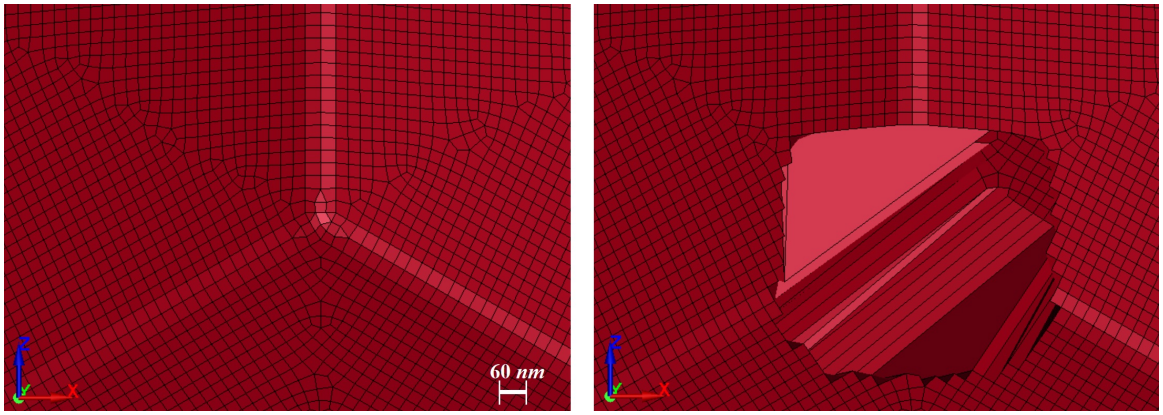


Figure 3.8: Regular 100 nm Tip (Left) versus Imploded Tip (Right)

Plastic Curve Determination

Plastic curve determination in a particularly iterative process. It begins by defining the material with the *MAT_PLASTICITY_COMPRESSION_TENSION keyword with tensile properties from any one strain rate tested. Since the reduced modulus, E_r is known from the test machine, the calculated stiffness in compression, E_{comp} is applied to the material card as well. Recall equation 2.25.

$$E_{comp} = \frac{1-\nu^2}{\left(\frac{1}{E_r} - \frac{1-\nu_I^2}{E_I}\right)} \tag{2.25}$$

Additionally, a curve of stress versus plastic strain in compression is guessed for this first iteration. From here, the curve is modified each iteration until the resulting load-displacement curve closely resembles the curves from testing. This method does not currently consider strain rate effects which is on the horizon.

3.3 Flat Plate Model

The Flat Plate Model is a 16×16 mm plate. Earlier models consist only of the coating layup with the bottom nodes of the primer fixed in all degrees of freedom. Later models include an aluminum substrate below the primer whose bottom nodes are fixed in all degrees

of freedom. Thus far, the two layer coating model has been modelled while the three layer is forthcoming.

Following the example set by Verma et al., the substrate is connected to the Primer with the *CONTACT_AUTOMATIC_SURFACE_TO_SURFACE_TIEBREAK keyword and the Primer connects to the Topcoat in the same Fashion. These contacts are modelled with OPTION = 6 which activates tiebreak for nodes initially in contact and failure initiates once the failure criterion is met. If a value is provided for CCRIT (value for PARAM in the contact card with OPTION = 6), then failure is fully realized once the distance between node initially in contact reaches this value [28]. Otherwise, delamination is governed with the failure criterion of

$$\left(\frac{|\sigma|}{NFLS}\right)^2 + \left(\frac{|\tau|}{SFLS}\right)^2 \geq 1 \tag{3.3}$$

where σ is the normal stress acting on the interface, τ is the shear stress acting on the interface, *NFLS* is the normal failure stress of the interface, and *SFLS* is the shear failure stress of the interface [3], [28]. Current models only consider *SFLS* as it may be obtained from the lap shear tests, at least for the critical interface. This is sufficient for the single impact case as the interface damage is expected to be below any threshold of significant damage however, the fatigue model will require more in-depth data derived from further testing. Because of this, OPTION = 6 will be replaced by a more robust option utilizing more fracture toughness data in the future.

3.3.1 Mapping CFD Data

In order to apply (LS-Dyna) the stressed found from the CFD team (ANSYS Fluent), a MATLAB script named "GetStress" was written. First, a solid model needs to be created and the nodal and elemental keyword data needs to be copied into separate files. In a nutshell, the script reads in several text files:

- **elements.txt** is a file containing the element keyword data (element ID, corresponding part ID, IDs of each node about the element) and is copied directly out of the LS-Dyna keyword file.

- **nodes.txt** is a file of nodal data (ID and coordinates) which is also copied directly from the keyword file.

- **coordinate_data_for_MATLAB.txt** is a file containing the coordinates for load points from the CFD team's analysis in ANSYS Fluent. Here, each row represents one coordinate point.

- **pressure_data_for_MATLAB.txt** is a file of the pressure exerted on the plate. In this file the rows and columns represent coordinate points and time steps, respectively.

- **tau_xy_data_for_MATLAB.txt** is just like the pressure data file but holds the applied shear stress data in the x-y direction.

- **tau_xz_data_for_MATLAB.txt** is just like the shear x-y file but holds the x-z direction of shear.

With these text files, the script determines the centroid of each element in the LS-Dyna keyword file and which points from the fluids coordinate data are the closest. It then maps pressures directly to element centroids to be applied on the top surface of the model and shear stresses are calculated at the element centroid and then broken apart into four components as

$$F_{node} = \frac{1}{4} \cdot \tau_{applied} A_{element} \tag{3.4}$$

and applied at the corners of the top surface elements. The forces from x-y and x-z shears are superimposed as are the equivalent nodal forces from neighboring elements. Since the CFD and FEM element locations may or may not perfectly align, linear interpolation is used as needed to determine the stress acting over an element's centroid.

When interpolating, GetStress checks for alignment between the FEM centroid and CFD data points in the y- and z- directions. Alignment in both directions means that the FEM centroid and CFD data point fall at the same location and thus, no interpolation is needed. If there is alignment in one direction, then the interpolation is one dimensional and equation 3.5 simplifies to equation 3.6. If there is no alignment, the interpolation is two dimensional

and follows equation 3.5 [29]. Let (y_c, z_c) denote the coordinates of the element centroid in LS-Dyna and let y_1, z_1, y_2, and z_2 denote the nearest y and z coordinates on either side of the centroid that have an applied stresses from CFD. Then we have variables Finally, let σ_{ij} denote the applied stress where $\sigma_{11} = \sigma_P = \textit{Pressure}$; $\sigma_{12} = \tau_{xy}$; $\sigma_{13} = \tau_{xz}$; and no other combinations of i and j are considered.

$$\sigma_{ij}(y_c, z_c) = \frac{\begin{Bmatrix} y_2 - y_c \\ y_c - y_1 \end{Bmatrix}^T \begin{bmatrix} \sigma_{ij}(y_1, z_1) & \sigma_{ij}(y_2, z_1) \\ \sigma_{ij}(y_1, z_2) & \sigma_{ij}(y_2, z_2) \end{bmatrix} \begin{Bmatrix} z_2 - z_c \\ z_c - z_1 \end{Bmatrix}}{((y_2 - y_1) \cdot (z_2 - z_1))} \tag{3.5}$$

$$\begin{gathered} \sigma_{ij}(y_c) = \frac{\sigma_{ij}(y_2) - \sigma_{ij}(y_1)}{y_2 - y_1}(y_c - y_1) + \sigma_{ij}(y_1) \\ \text{or} \\ \sigma_{ij}(z_c) = \frac{\sigma_{ij}(z_2) - \sigma_{ij}(z_1)}{z_2 - z_1}(z_c - z_1) + \sigma_{ij}(z_1) \end{gathered} \tag{3.6}$$

With pressures and representative shear forces, it is time to apply them to the model. GetStress generates a load curve of pressure versus time for each element centroid and load curve force versus time for each node. The code is then able to output these load curves with the proper syntax to be copied directly into the LS-Dyna keyword file. The pressures are applied with the *LOAD_SEGMENT keyword which requires a load curve of pressure versus time and a set of nodes to outline the boundary of pressure application. In this case, the boarder nodes are the nodes outlining the top surface of each element. GetStress outputs load curves in the correct syntax for the *DEFINE_CURVE keyword.

The equivalent shear forces are applied by way of the *LOAD_NODE_POINT keyword. This card requires load curves, defined with the *DEFINE_CURVE keyword, for each node to be loaded.

At the end of the day, GetStress outputs several text files for inspection purposes and one main file called “copy_to_keyword.txt” which contains load curves for each element and node in the syntax of the *DEFINE_CURVE keyword, as well as the fully defined *LOAD_SEGMENT and *LOAD_NODE_POINT keywords.

The full code is available in Appendix B.2.

Chapter 4

RESULTS AND ANALYSIS

4.1 Wind Turbine Blade Model Recreation

The first results to share are those of the A.S. Verma et al. SPH validation recreation. From Figure 4.1, we see that convergence is reached shy of 140,000 particles. The converged recreation model slightly under predicts the force while Verma's slightly over predicts the force. This variance is acceptable as all the last four points are all within 1% of the experimental data. The convergence results in this paper differ from those of Verma et al. primarily due to the differences in SPH modelling that were not disclosed in the Verma paper such as friction and particle distribution. Likewise, this explains difference in droplet morphology between this work and Verma et al.

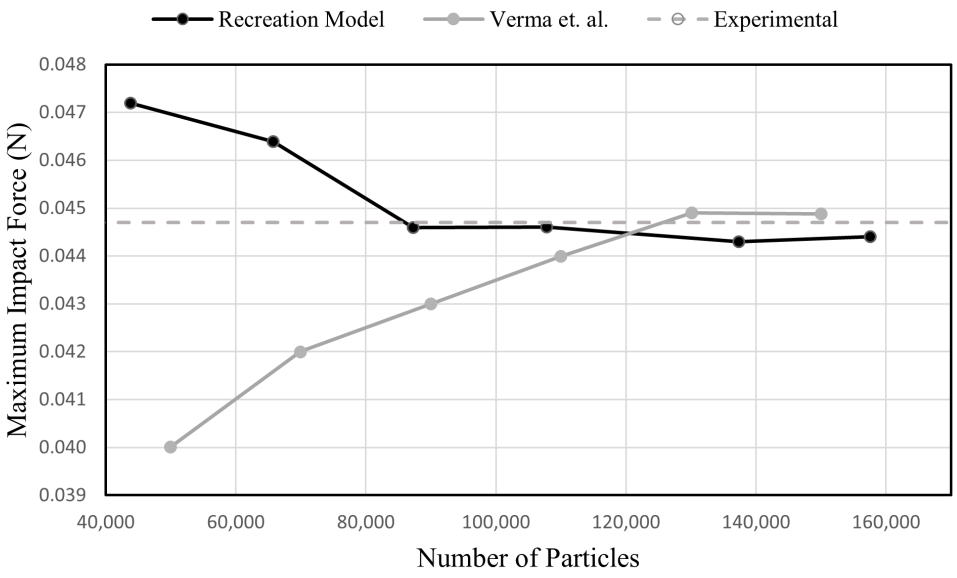


Figure 4.1: SPH Convergence (2.7 mm diameter at 2.67 m/s)

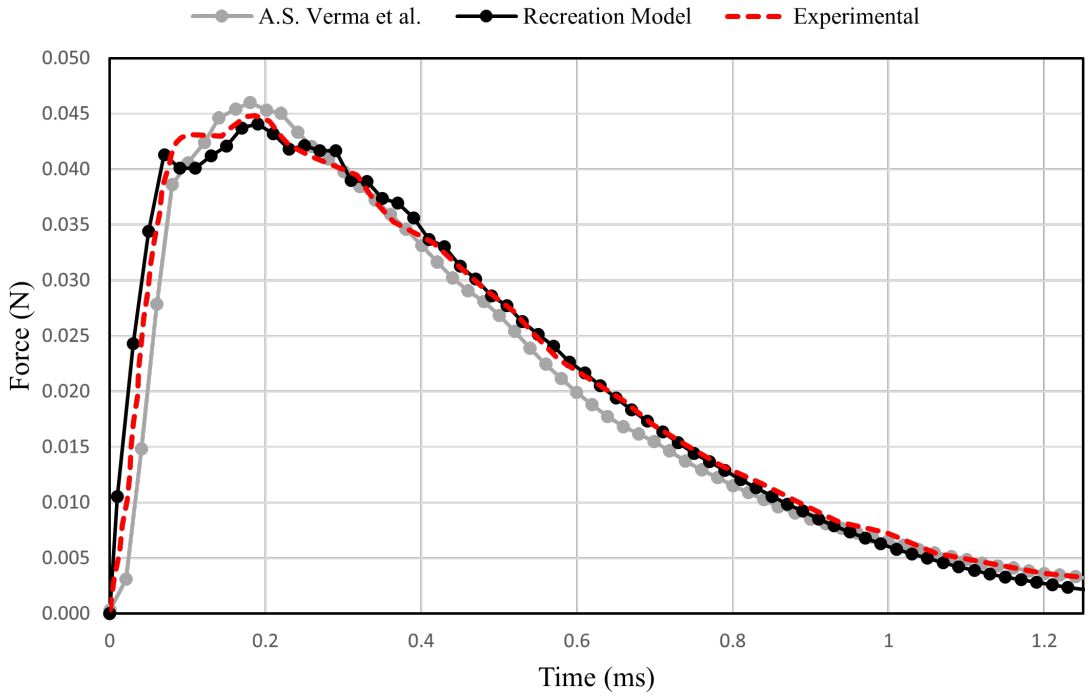


Figure 4.2: Aluminum Plate Model Comparison (2.7 mm diameter at 2.67 m/s)

Next, we may compare Verma's aluminum plate convergence model with its recreation. Looking at Figure 4.2, we first notice that the recreation model is more similar to the experimental data than the Verma model for most of the simulation. Then, in Figure 4.3, we see the droplet morphology from Verma et al. as well as the recreation and real water. Here, we notice that both the Verma study and recreation match the actual droplet's morphology decently well.

The final model recreation result to observe is the single impact gelcoat model. The impact force versus time for a 2.76 mm droplet impacting at 106.7 m/s for the original and recreated Gelcoat model is shown in Figure 4.4. Here, we can see similar but not identical behaviour. Both plots have close maxima at 93.3 N for Verma and 92.2 N for the recreation. Further, the overall shape of the decay in force over time is very similar with the only major difference between the two plots being the early spike in the Verma model that is not present

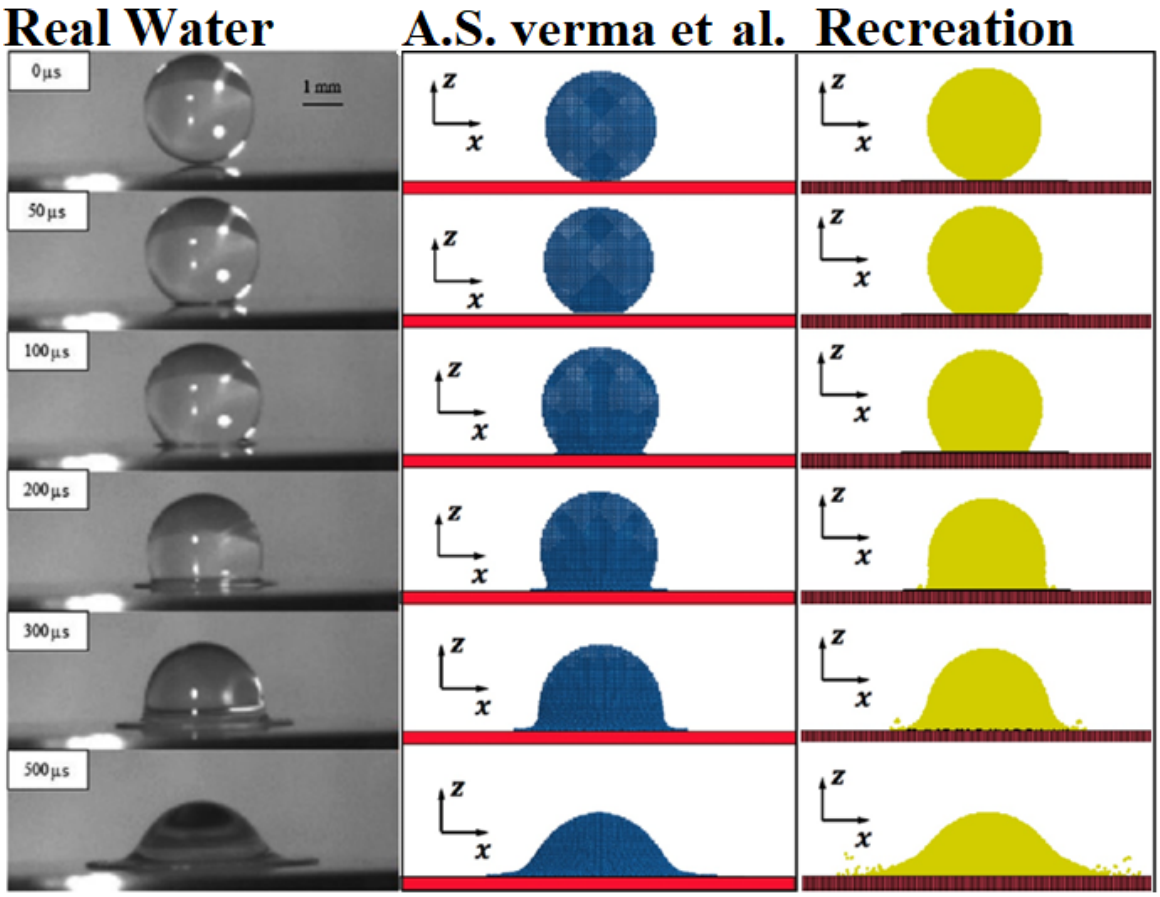


Figure 4.3: Aluminum Plate Droplet Morphology (2.7 mm diameter at 2.67 m/s)

at such severity in the recreation. Differences like that can be attributed to the fact that this is a highly dynamical simulation with some differences between the models. Highly dynamical simulations will often produce a different result when run on different CPU's. Another model variation which likely contributed to the differences is friction which was not input into any of the contact cards and thus, not fully described.

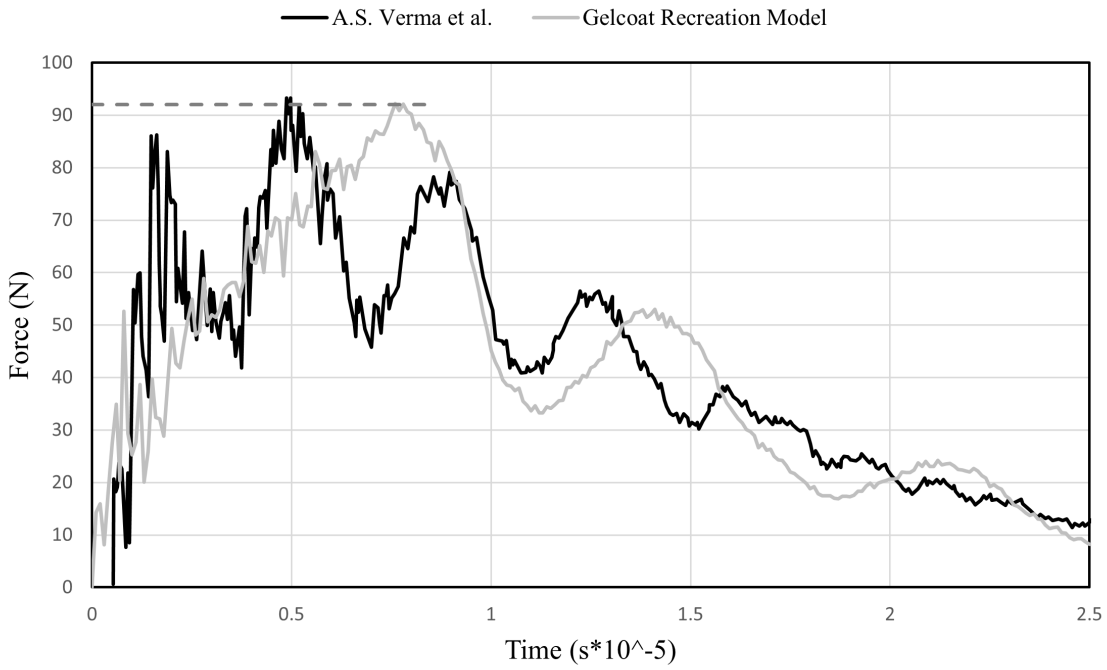


Figure 4.4: Aluminum Plate Model Comparison (2.74 mm diameter at 106.7 m/s)

4.2 Tensile Tests and Validation

The tension tests have come a long way in this project. They began with plain rectangular samples and evolving into DIC with accurate LS-Dyna modelling. These results are critical in shaping our understanding of the tensile material behaviour.

4.2.1 Non-DIC Test Results

We may begin our tensile test analyses with the samples run without DIC. Recall equations

$$\sigma^E = \frac{P}{A_0} \tag{2.1}$$

$$\epsilon^E = \frac{\Delta L}{L_0} \tag{2.2}$$

$$\epsilon^T = ln(1 + \epsilon^E) \tag{2.4}$$

$$\sigma^T = \sigma^E \cdot (1 + \epsilon^E) \tag{2.7}$$

used to calculate engineering stress and strain from the machine's transducer load, crosshead displacement, and sample's initial area and then to approximate true stress and strain for all samples up to and excluding DIC.

From the First Samples

The first results to discuss are from the first samples. As detailed in section 2.1.2, these were simple rectangles cut by a razor. Looking at Figures 4.5, 4.6, and 4.7, several observations stand out:

1. **Jagged Curves:** Every curve in these three charts at rates greater than 10 microns per second, with the exception of the Topcoat at 100 microns per second, resemble a staircase. One likely explanation for this is that the square samples could have slipped in the test machine's jaws during testing. Slippage would explain the staircase as it would result in sudden changes in force. Another potential cause, particularly for the 1000 micron per second cases, is that the machine's frame rate is too slow to fully capture the material's strain under load, even moreso when the sample stretches a lot. For instance, this behaviour is most noticeable in the Topcoat's run at 1000 microns per second, which strained three times as much as the Clearcoat did for the same rate.

2. **Strain Rate Dependence:** All three of these charts show clear evidence of strain rate dependence for their material when considered alone. The Topcoat's runs at 10 and 100 microns per second have a similar elastic stiffness to start but their plastic

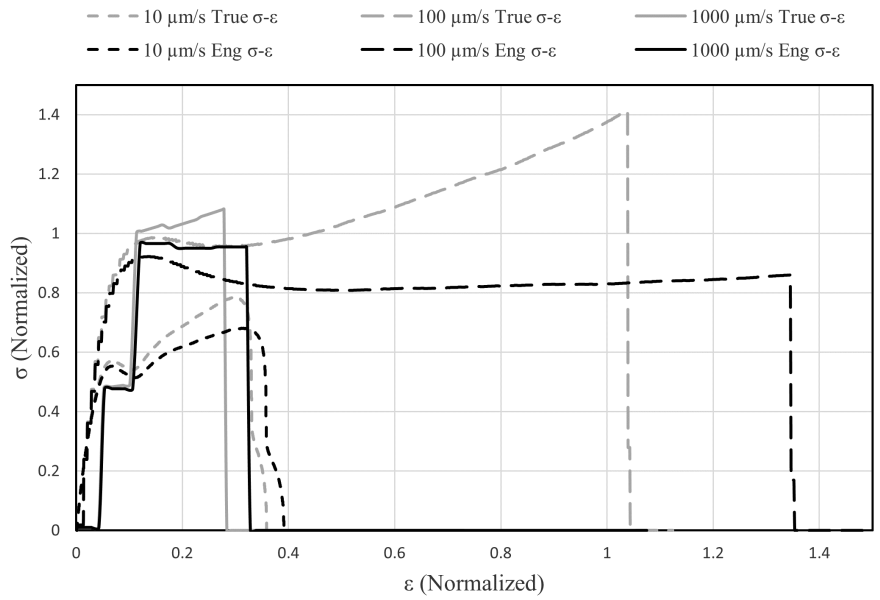


Figure 4.5: σ-ϵ for the Topcoat's First Tests

behaviour is vastly different. The 100 micron per second run has a much higher yield stress, failure stress, and failure strain. Similarly, the Clearcoat's 100 and 10 micron per second runs produced unique curves. The Clearcoat is noticeably stiffer at 100 microns per second with a higher yield stress and failure stress but smaller failure strain. Finally, the Primer's curves exhibit vastly different behaviour at the two rates.

3. **Only True Stress for the Primer:** Only the true stress is reported here for the primer because it is virtually identical to the engineering stress due to such small stains.

4. **Overall Poor Results:** All things considered, these are not quality test results largely due to the fact that they are from hand cut rectangular samples and tested with the machine's crosshead displacement for strain measurement.

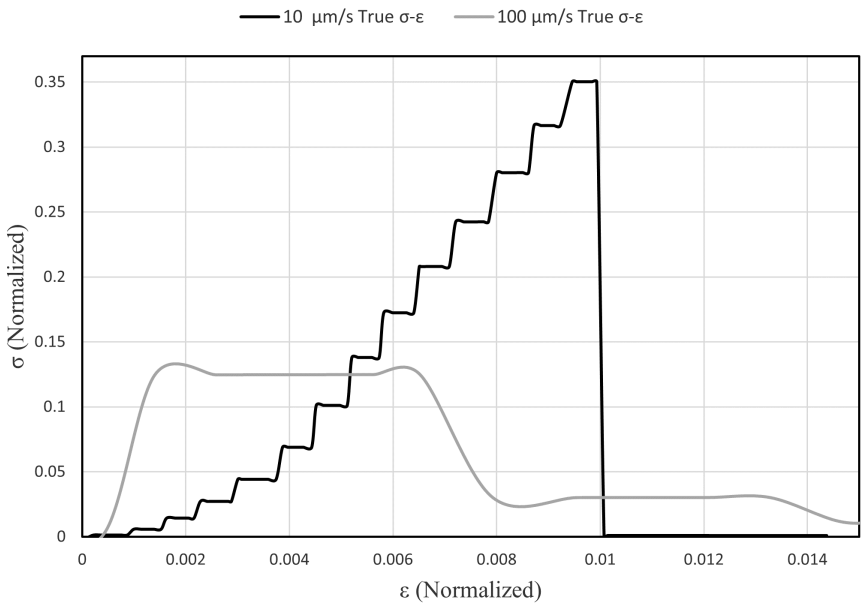


Figure 4.6: σ-ϵ for the Primer's First Tests

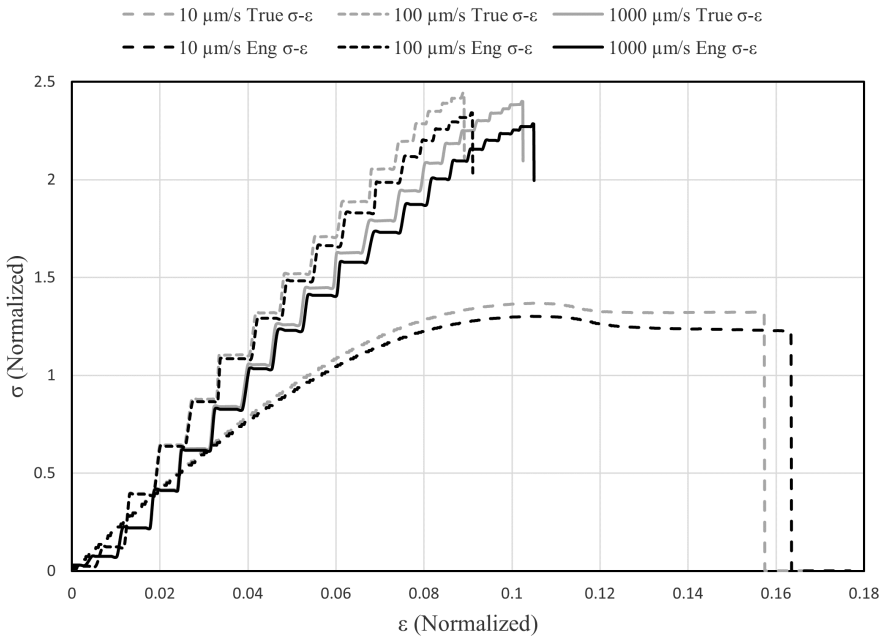


Figure 4.7: σ-ϵ for the Clearcoat's First Tests

D1708 and the Microscope

The next set of test results to analyze is the set of dotted dogbones for the Primer and Topcoat for which only the true stress versus true strain is reported. These samples behaved much better than our first set with more apparent consistency between rates. From Figure 4.8 we notice that the stiffness and yield stress of the Topcoat is close to the same at each rate with the largest difference between rates being the failure strain which may be explained by stress concentrations introduced into the samples during preparation. Figure 4.10 (Right) is a good example of sample failure at a stress concentration point. When this sample was prepared, a small cut around the radius veering inward was introduced thereby creating a singularity in the sample and leading to premature failure from a crack that initiated at this point. Nonetheless, these results were very promising compared to the first samples and allowed for the creation of a preliminary material model.

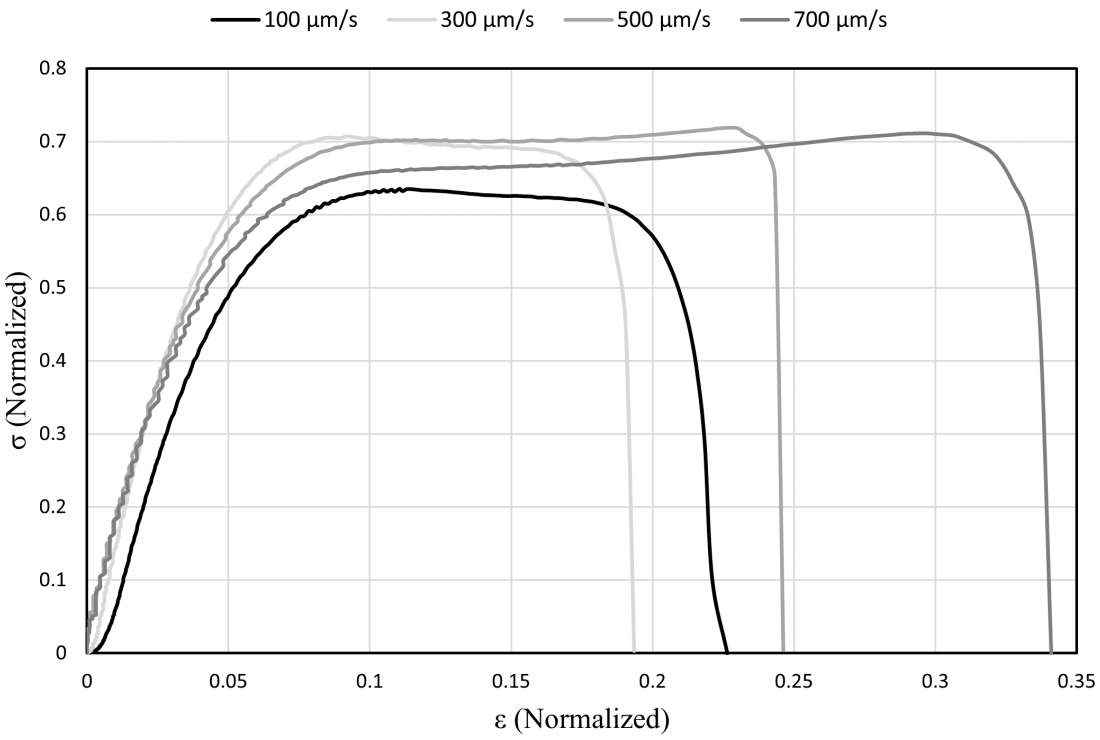


Figure 4.8: True σ-ϵ for the Topcoat with D1708 Using the Microscope Method

Now considering the Primer, Figure 4.9 exhibits substantially better quality results than the first samples in Figure 4.6. With the dogbone samples, the Primer demonstrates similar stiffnesses across all four samples with varying failure strains. The chief cause for this variance in failure strains is due to stress concentrations from sample preparation as with the Topcoat.

Being the thinnest and most brittle of all four materials, the Primer did not respond well to being cut. Several samples had to be remade during the testing process due to their crumbling apart before testing. The samples that did make it had very rough edges. Despite the edges, some samples managed to fail as expected like Sample 3 (500 $\mu m/s$) while others failed as a clear result of a stress riser.

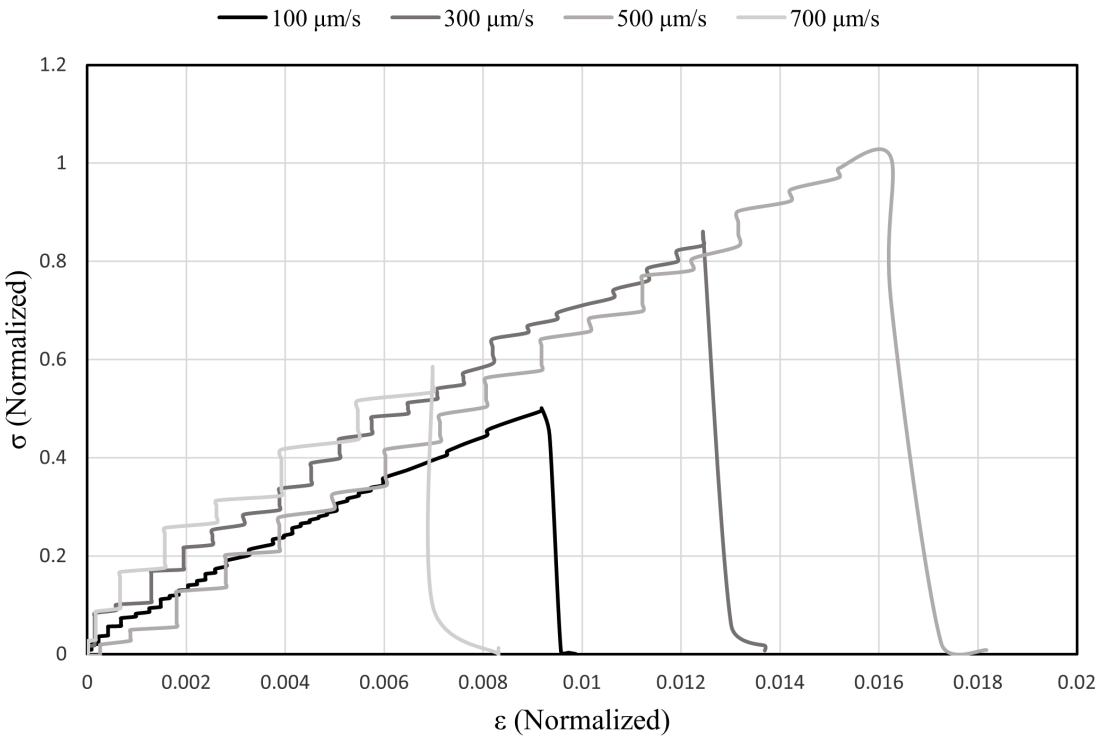


Figure 4.9: True σ-ϵ for the Primer with D1708 Using the Microscope Method

in addition to the stress concentrations from cutting, these results retain some drawbacks.

First and foremost, the true stress and true strain still assume conservation of volume. Despite using the microscope to measure the cross sectional width and other initial dimensions of the samples, equations 2.4 and 2.7 are used to determine the true stress and strain of the samples for three main reasons: First, the resolution of the test videos was not sufficient to accurately measure the width of the cross section to a degree of repeatability. Second of all, trying to manually track the width of the cross section in ImageJ through the testing videos is excruciatingly time consuming for the level of precision it offers. And finally, DIC is a better option and was scheduled to happen in the near future. Thus, the extra effort required to marginally improve these curves was not worth it. All things considered; we have learned some valuable lessons from these samples that we may carry with us into our DIC analysis. For instance, The stress concentrations that were introduced while preparing the cut samples were quite pronounced. From this, we may consider that brittle materials like the Primer could have stress risers introduced in future sample preparation even if the method is more robust.

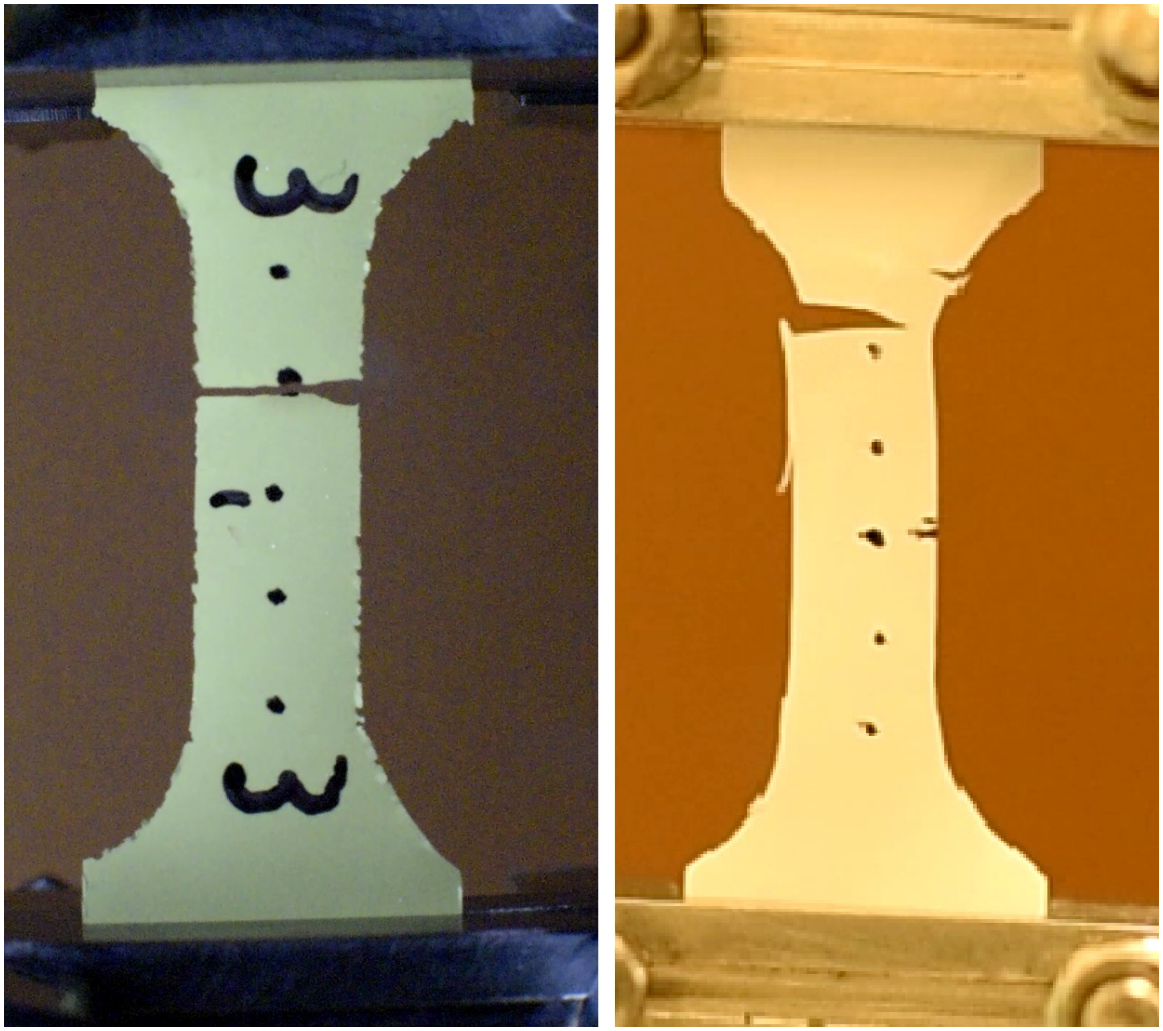


Figure 4.10: Failed Primer (Left) and Topcoat (Right) Samples at 500 and 100 $\mu m/s$, respectively.

4.2.2 DIC Results

The best test results came from digital image correlation. We will begin by discussing results obtained with Methods B and F. As described in section 2.1.4, these methods use the x and y direction strains calculated within GOM Correlate Pro as well as the z coordinates to calculate z strain. It was quickly realized that Methods B and F would not work very well. As shown in Figure 4.11, the methods that utilize z coordinates to calculate z strain have

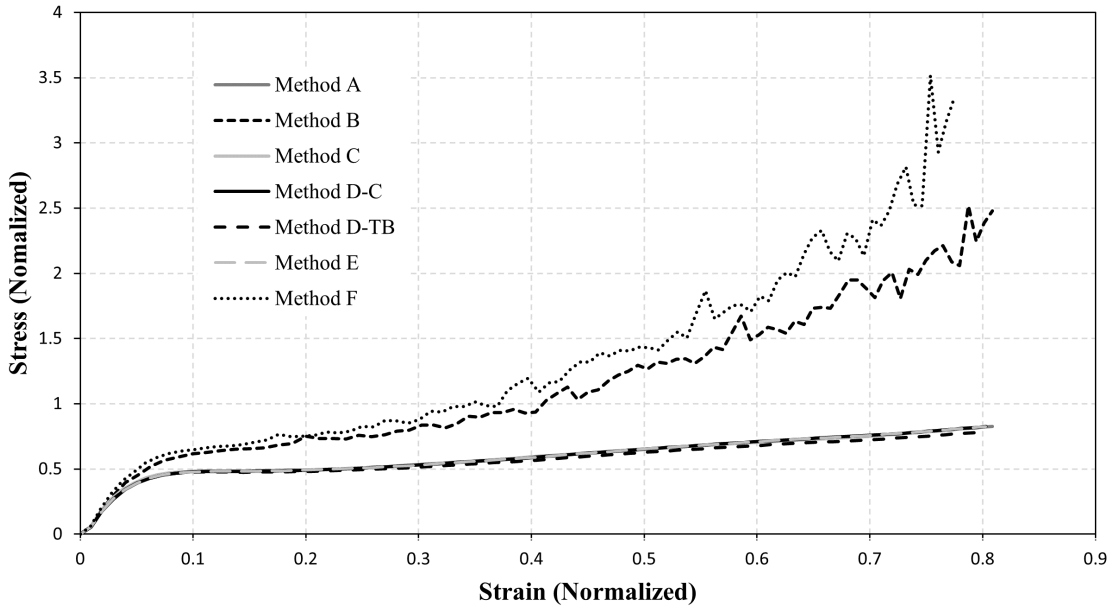


Figure 4.11: Topcoat Sample 9 (500 $\mu m/s$) All Methods

very rough, jagged curves that grossly over predict stress as compared to the other methods. If these were closer to the other methods and smoother curves, they would be interpreted as more accurate since they have one less assumption associated with them. However, this is not the case. It is worth noting that at a slower strain rate for the same material, Methods B and F are more reasonable which we can see in Figure 4.11 however, Methods B and F still result in very rough curves and this inconsistency between samples suggests that Methods B and F and unreliable. Furthermore, we may take a look at the results of one of the primer samples in Figure 4.13. With this example, the sample is so thin that even more error is introduced into Methods B and F than with the Topcoat.

One likely cause for these poor results of z strain in Methods B and F comes from sample bowing. Figure 4.14 (recall that z is the out of plane direction) illustrates this phenomenon as we advance in the test, one can observe that the z coordinates in the center of the sample move one direction while the out of plane motion at the edges of the sample is in the opposite direction. This was present to some extent in every sample and more pronounced in some

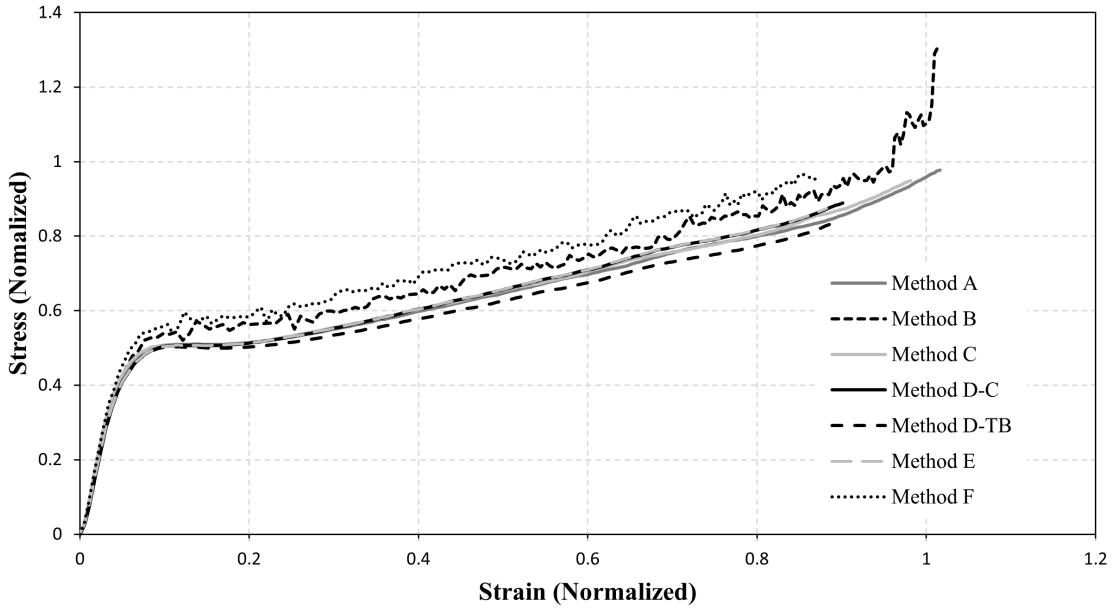


Figure 4.12: Topcoat Sample 6 (300 $\mu m/s$) All Methods

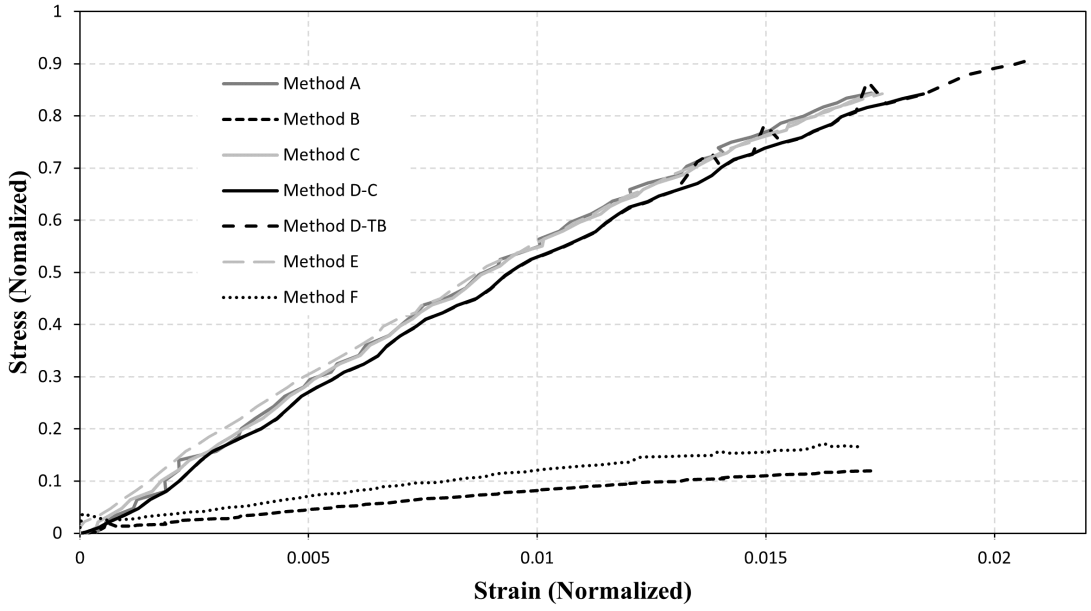


Figure 4.13: Primer Sample 4 (50 $\mu m/s$) All Methods

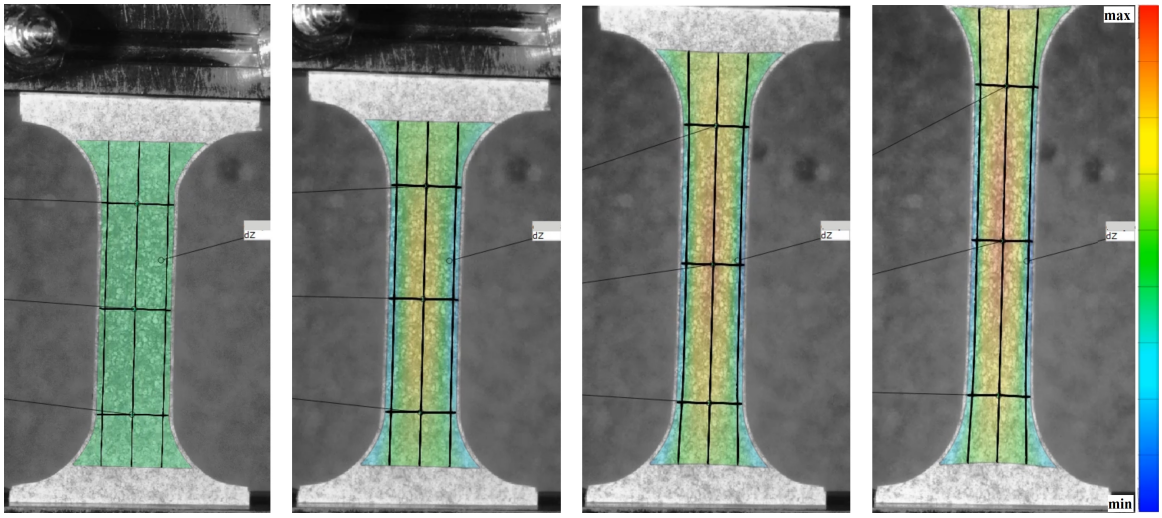


Figure 4.14: dz of Topcoat Sample 9 During Testing

samples than others. Between the inconsistent and obscure z strain results and the sample bowing, it is conclusive to say that the dz and z strain calculations are unreliable.

With this, we move forward with a focus on the other methods. Method A is the primary method of interest. The method is simple in that it uses strains output by GOM, only requiring calculations of instantaneous area and stress from that with equations

$$\sigma_j^T = \frac{P_j}{A_j} \tag{2.18}$$

$$A_j = A_0 \cdot e^{2\epsilon_{xx,j}^T} \tag{2.19}$$

to develop a stress-strain curve. Since Method A utilizes the center band of high contrast points across the sample, it is the best method to account for necking. Most samples will be evaluated with Method A as a primary measure of stress and strain henceforth. Other methods will be considered in the background as well, in part to validate Method A. Some samples may be evaluated and reported with other methods if an inconsistency with Method A arises.

Now having the stress-strain methodology sorted out, we move on to the Topcoat by Method A. There was a total of five Topcoat samples tested with DIC. Sample 6 was tested

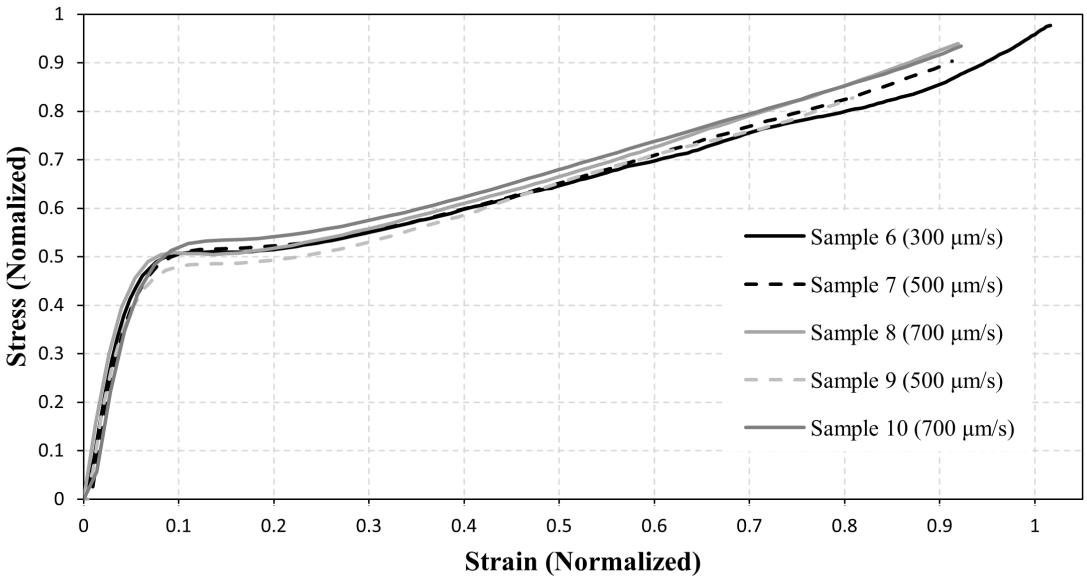


Figure 4.15: Topcoat σ^T- ϵ^T Curves by Method A

with a crosshead rate of 300 $\mu m/s$, Samples 7 and 9 were tested at 500 $\mu m/s$, and Samples 8 and 10 ran at 700 $\mu m/s$. Looking at Figure 4.15, we see there is some strain rate dependence present which is particularly noticeable toward the end of the plastic regime. All samples share an elastic response and although their plastic responses differ, they are fairly similar for all five giving a failure strain of about 0.9 and true stress at failure also about 0.9.

Moving on to the Primer, one major problem is immediately noticeable: only two samples provided quality data. Samples 4 and 9, both run at 50 $\mu m/s$. The other samples all failed prematurely which could be due to stress risers in the form of small nicks on the edges of the sample, thickness variations, or clamping force effects. Being the thinnest and most brittle material, the Primer is extra sensitive to geometric imperfections. More testing is needed to better understand what exactly is happening.

Now, looking at the Basecoat, we notice some strain rate dependence most prevalent in Sample 4, the only sample run at 300 $\mu m/s$. Samples 7 and 9 were run at 500$\mu m/s$ with Samples 8 and 10 at 700 $\mu m/s$. Samples 7 and 8 have similar behaviour with minimal strain rate dependence between the faster two rates.

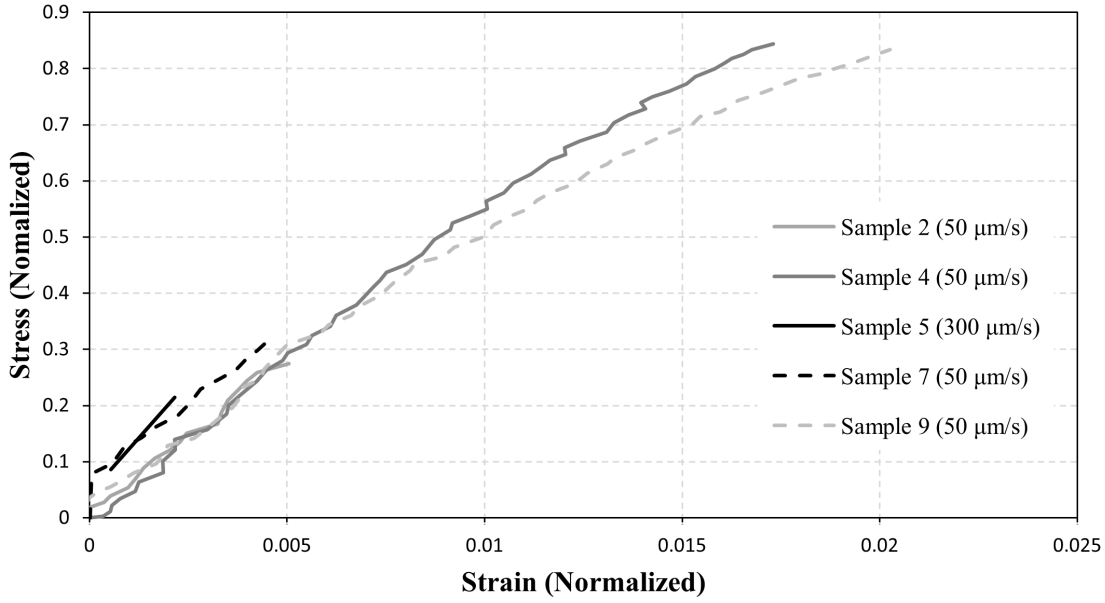


Figure 4.16: Primer σ^T- ϵ^T Curves by Method A

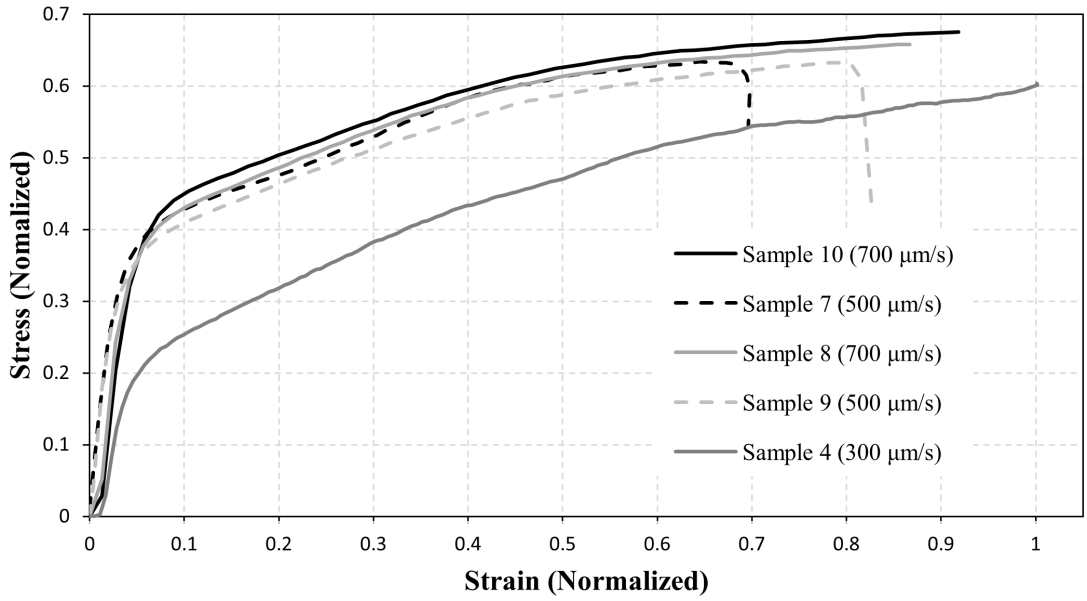


Figure 4.17: Basecoat σ^T- ϵ^T Curves by Method A

Material	E_T	E_C	ν	ν_{pl}	σ_y	σ_u	$\epsilon_{f,pl}$
Topcoat	10.2	26.9	0.37	0.44	0.41	0.9	0.84
Primer	66.8	67.2	0.35	0.35	0.38*	0.83	0.01**
Basecoat	10.1	-	0.49	0.44	0.18-0.37	0.63	0.88

∗The Primer exhibits fully elastic behaviour when a 0.2% offset is used. This was chosen by inspection.

∗∗Some iteration involved

Table 4.1: Normalized Material Properties from Tension Tests

4.2.3 Model Calibration

With all of our test data, we may look at the calibration of the tensile models in LS-Dyna. Using some of the values from Table 4.1 and curves of stress versus plastic strain found by subtracting the elastic recovery from the full stress strain curve. We start with the Topcoat:

The Topcoat's material model uses the *MAT_SAMP-1 keyword with three plastic tension curves, one for each strain rate tested, and one plastic compression curve (see Section 4.3). Since SAMP-1 relies on plastic curves, no yield strength in input into the model. The recreations were all velocity controlled and as we can see in Figure 4.18, the model is calibrated, matching up well at the three strain rates.

The Primer is modelled with the *MAT_PLASTICITY_COMPRESSION_TENSION keyword and strain rate dependence is not considered. Like SAMP-1, this material card relies on plastic curves and no yield stress value is necessary. Also like the Topcoat, the Primer's model uses a plastic curve in compression that was found through nanoindentation recreation iterations. Looking at Figure 4.19, we see that the Model is able to describe the Primer with respect to both Sample 4 and Sample 9. This model may change as more testing is completed in the future.

Now, we may discuss the Basecoat. Similar to the topcoat, the Basecoat was modelled with three plastic tension curves in the *MAT_SAMP-1 keyword. Of all the validations, the Basecoat's 300 and 500 $\mu m/s$ runs have the most error. The is likely due to the fact that the validation simulations were velocity controlled and the strain rates may have varied

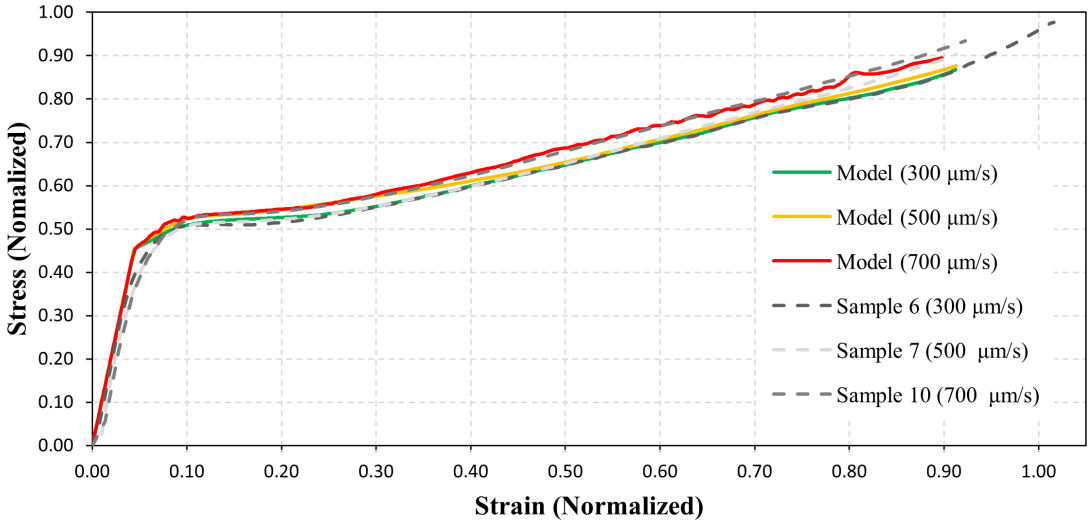


Figure 4.18: Topcoat Model Calibration

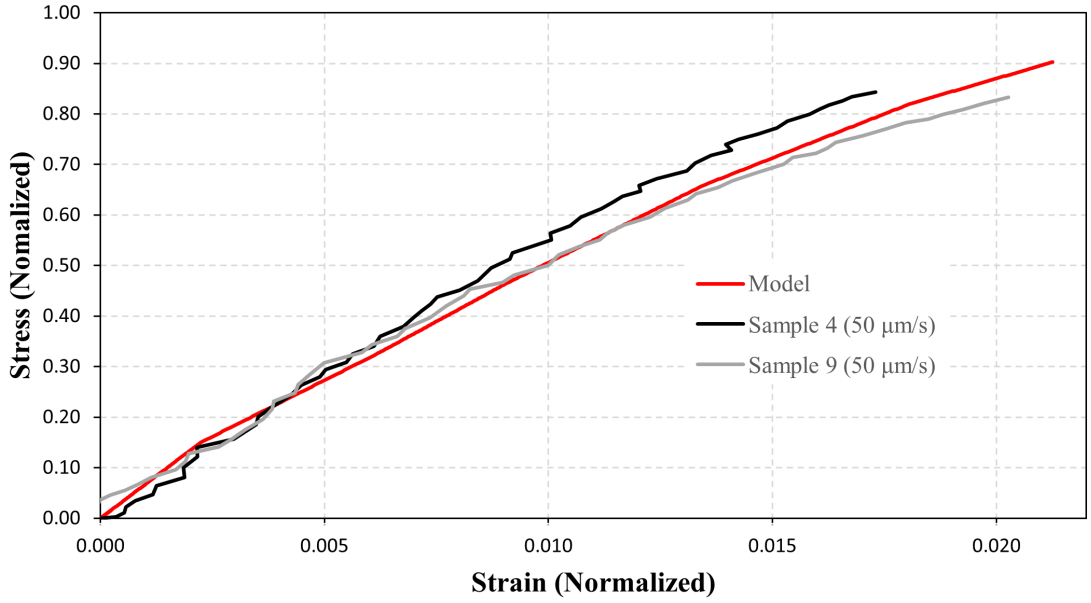


Figure 4.19: Primer Model Calibration

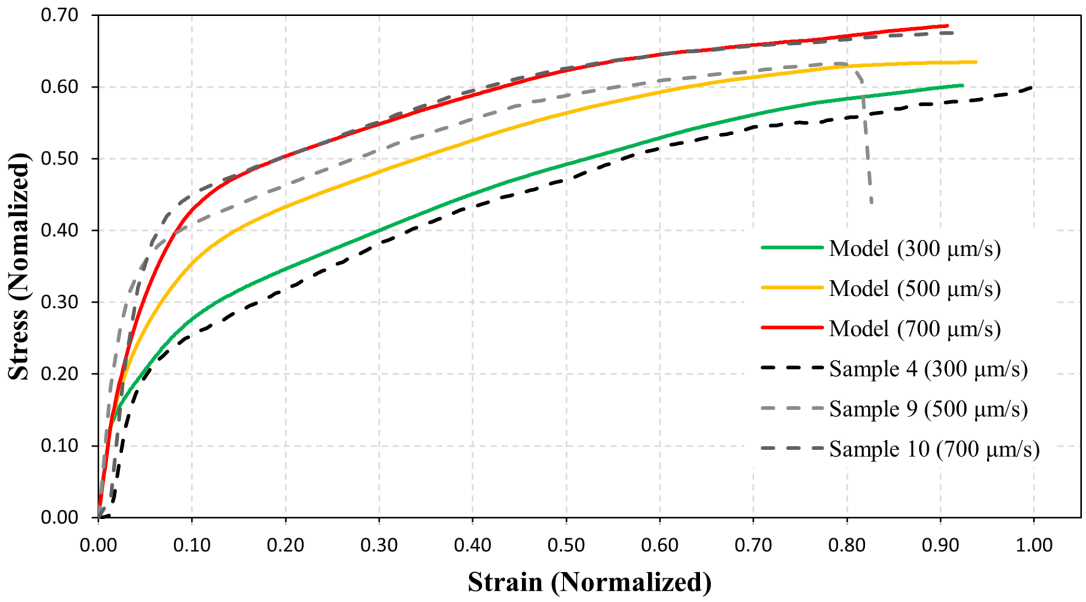


Figure 4.20: Basecoat Model Calibration

as compared to the instantaneous strain rates during physical testing. Even with the small errors, we can see that the material model aligns well with each corresponding sample.

The Clearcoat's DIC tension data has not yet been released.

4.3 Nanoindentation Recreation and Results

The next set of tests to discuss are the Nanoindentation tests for which we begin with the physical test results. The chart legends read "Run-x (Ramp up in seconds - hold time in seconds - ramp down in seconds)." Looking at Figure 4.21, we notice very little evidence of strain rate dependence as only one curve falls out of the larger cluster and all of the curves exhibit a similar amount of creep, evidenced by the flat at the top of curve where load controlled tests had to increase depth to sustain the load. Comparing that to Figure 4.22, we see some clear evidence of strain rate dependence as all of the 10 s runs experienced creep while the quicker ones did not. Nonetheless, the overall curve shapes do not significantly differ. Similarly, the Basecoat (see Figure 4.23) shows substantial strain rate dependence

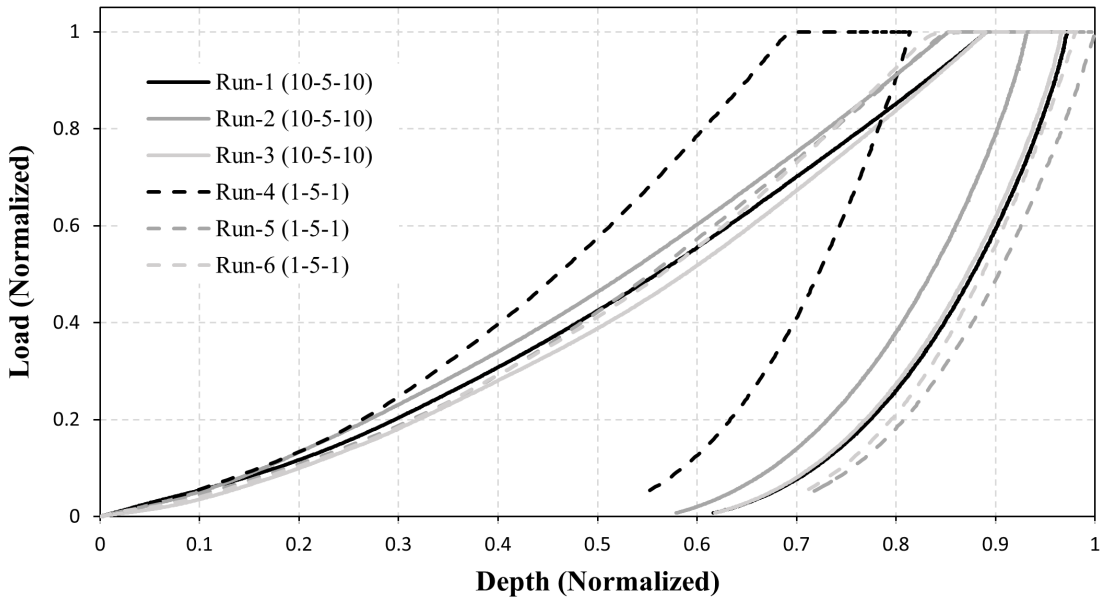


Figure 4.21: Topcoat Nanoindentation Results

between the two rates as the loading, hold, and unloading sections are all distinctly different between rates. This is in stark contrast to the Clearcoat which shows almost no strain rate dependence with a close cluster of curves for every test run.

We may more easily compare and contrast all four materials in Nanoindentation with Figure 4.25. Here, we observe that the Clearcoat and Topcoat have similar stiffnesses at lower depth and diverge around 0.6 units deep. The Primer is a bit less stiff than both the Clearcoat and Topcoat but much stiffer than the Basecoat, which is the least stiff of them all, at either rate.

Now, we may move on to mesh convergence. In Figure 4.26 we see the convergence plot of the load at the beginning of hold time versus the number of elements in the model with data labels at each point denoting the element size in microns. This study was conducted on the Topcoat with a first iteration curve guess, which is why it does not converge to the Topcoat's actual load. From the plot, 50 nm elements were determined to be an appropriate size that was small enough to obtain accurate results and large enough that the model would

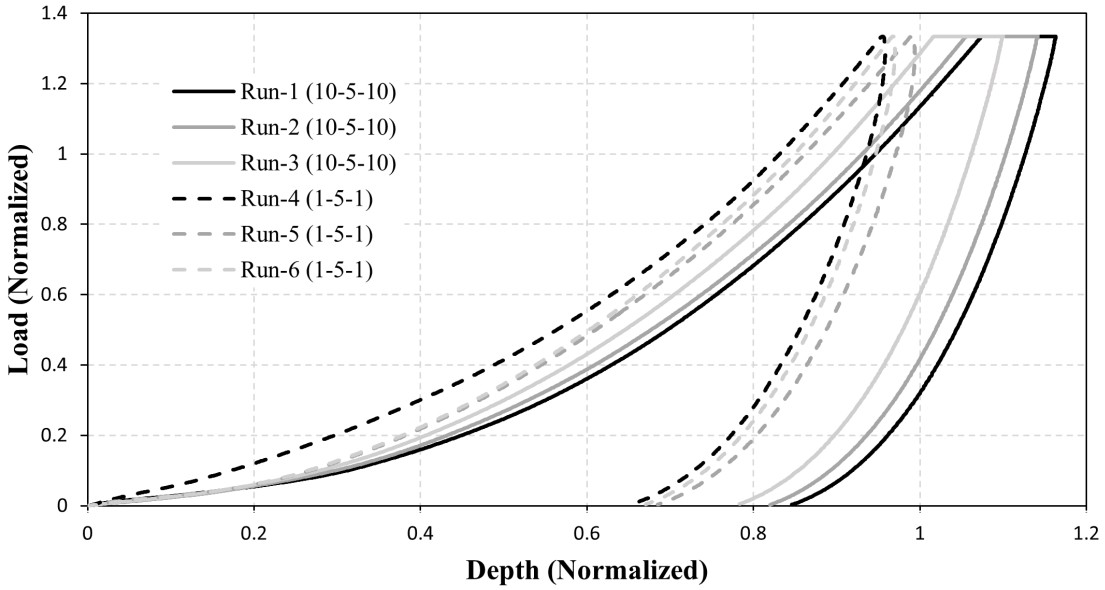


Figure 4.22: Primer Nanoindentation Results

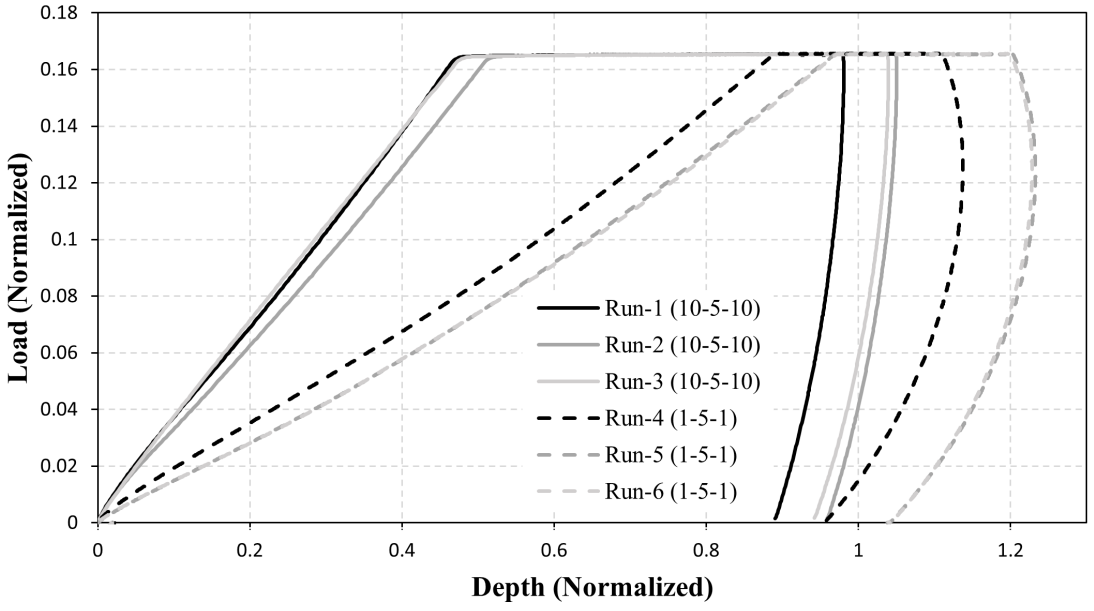


Figure 4.23: Basecoat Nanoindentation Results

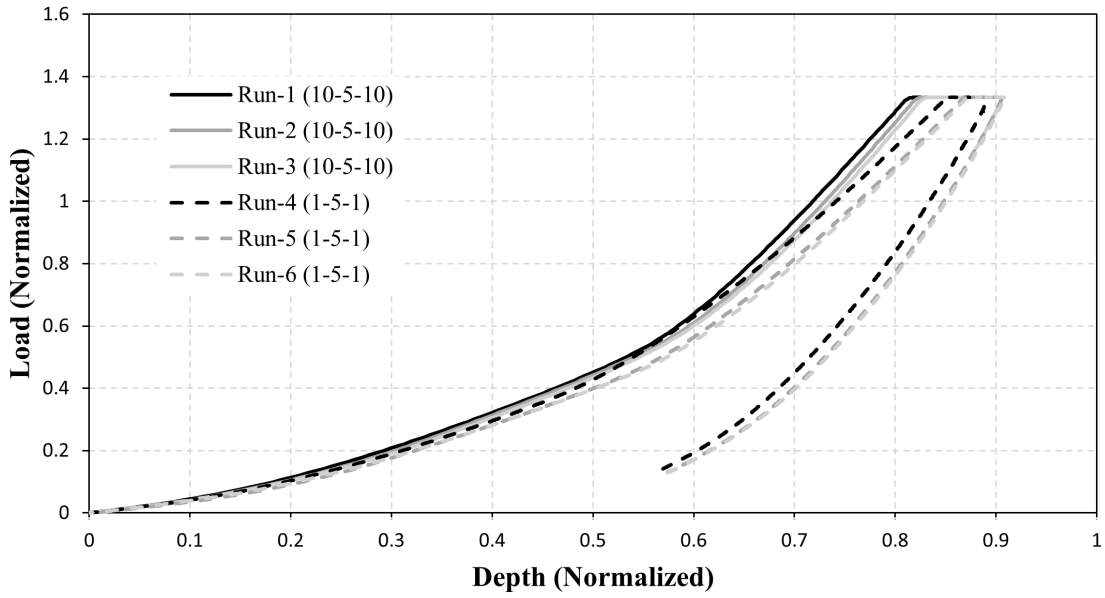


Figure 4.24: Clearcoat Nanoindentation Results

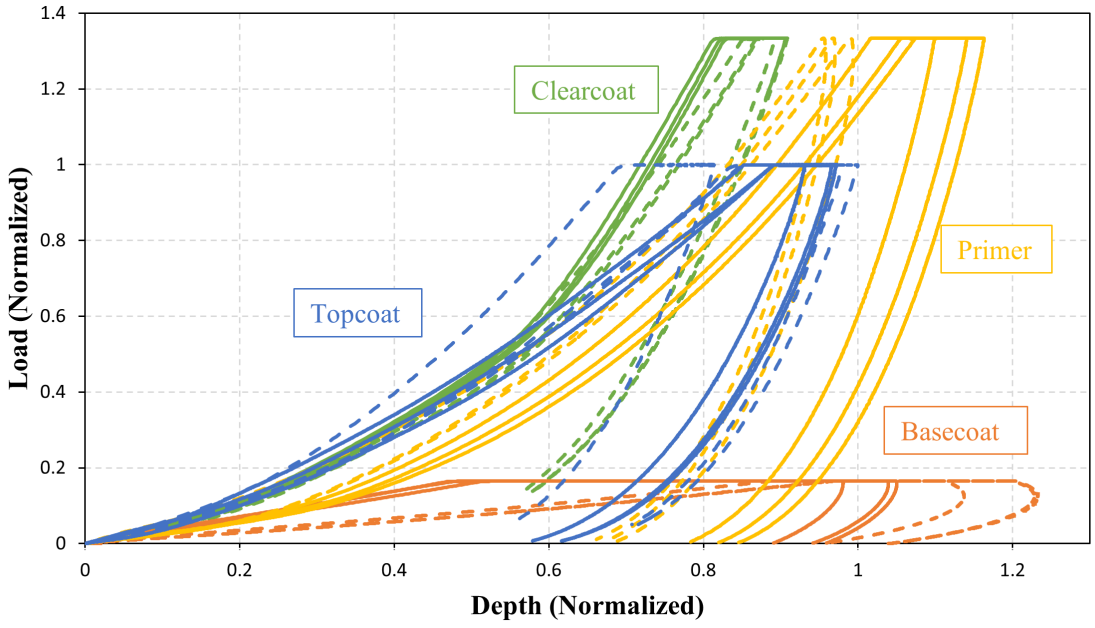


Figure 4.25: All Material's Nanoindentation Results Together

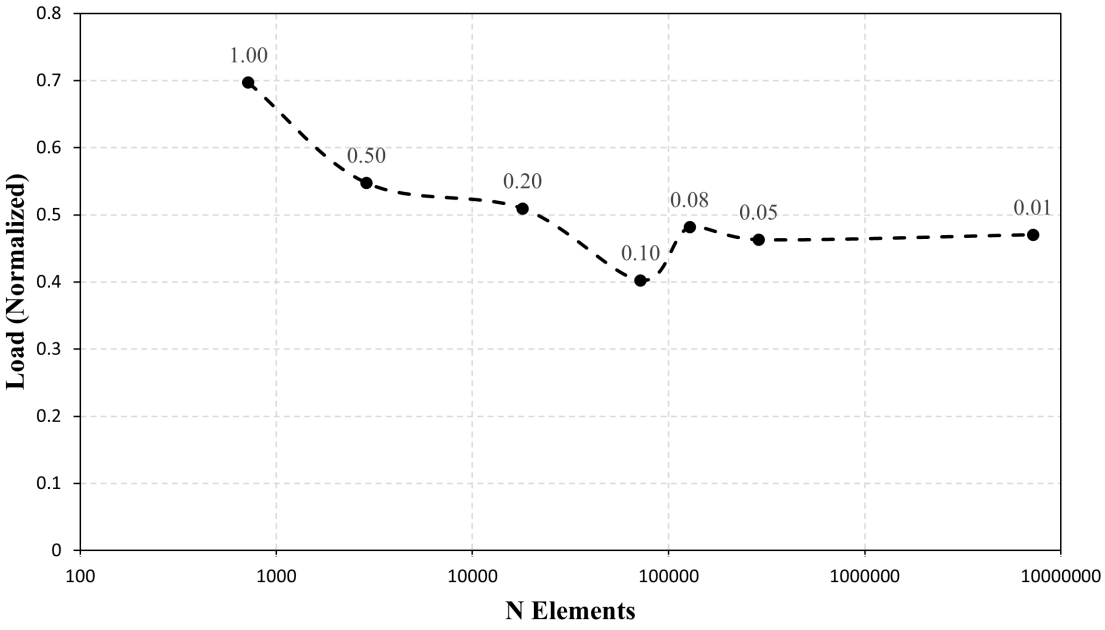


Figure 4.26: Nanoindentation Load at Hold Mesh Convergence

run in a reasonable length of time.

Another useful comparison is that of the plastic indentation left behind in a real nanoindentation test with a simulation. In Figure 4.27, we can see several tests on the Topcoat adjacent to the calibrated recreation model. Here we notice that plastic indentations have consistent shapes. Taking this a step further, Figure 4.28 overlays one of the actual test indents with that of the simulation for an easier comparison showing consistent shapes.

With a converged mesh and valid indentation, we may take a look at plastic curve results. The Topcoat has been characterized with one representative curve in compression and the Primer has had one of two curves characterized so far. The Clearcoat is next on the radar and Basecoat is further in the future because an appreciable portion of its load-displacement curve may depend on indenter tip radius. Figure 4.29 showcases the Topcoat model's ability to replicate the nanoindentation tests. In determining a plastic curve through iteration, the ramp up is the most important part of the curve to consider as the materials compressive plasticity has a large influence on this section and a very small effect on the

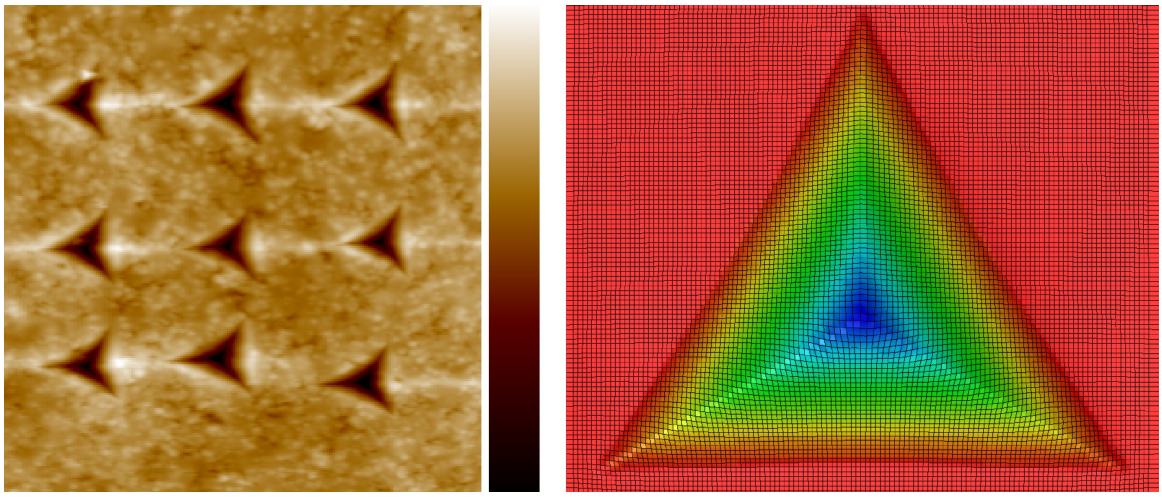


Figure 4.27: Topcoat Test (Left) and Recreation (Right) Plastic Indents

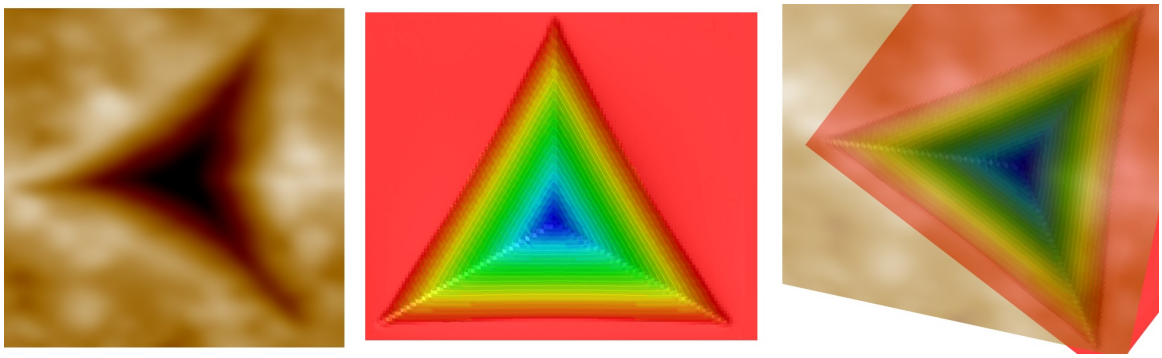


Figure 4.28: Overlay of Simulation Indentation with Actual Test Sample

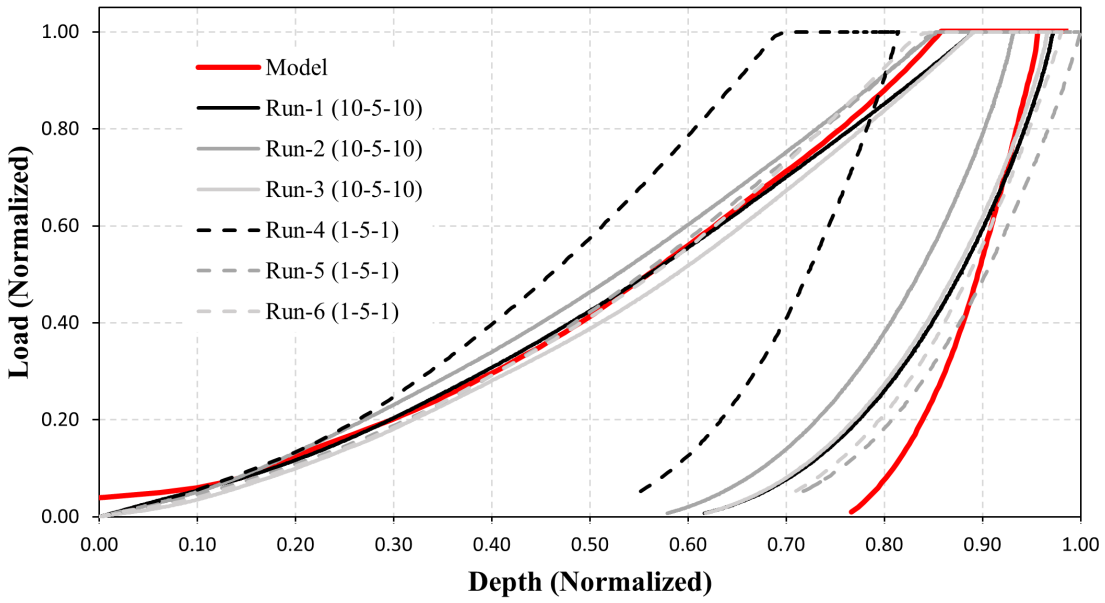


Figure 4.29: Topcoat Model, Calibrated in Nanoindentation

shape of the others. The plastic curve that reached this convergence was the 80^{th} iteration of the Topcoat's curve and model. This particular material took so long as it was the first one and there was a lot of learning involved. Earlier iterations played with the indenter tip radius until it was realized that this had a negligible effect on the majority of nanoindentation tests analyzed herein. The final plastic compression curve, as shown in Figure 4.30 is significantly stiffer than the Topcoat's plastic tension curves at smaller strains and about twice as stiff as the Topcoat's plastic tension curves for larger strains. Additionally, the compression curve reaches a higher stress. The Primer only took 15 iterations to find a well matching curve for the 1 s load case as shown in Figure 4.32. The Topcoat in compression is much more similar to itself in tension that the Primer is. As we can see by Figure 4.31, the plastic regime is compression is quite extensive and the effective yield stress versus plastic strain curve has a very shallow stiffness for the majority of it. Furthermore, the Primer's plastic curve in tension was nonexistent for a 0.2% yield and resulting tensile curve may assume too low of a yield stress.

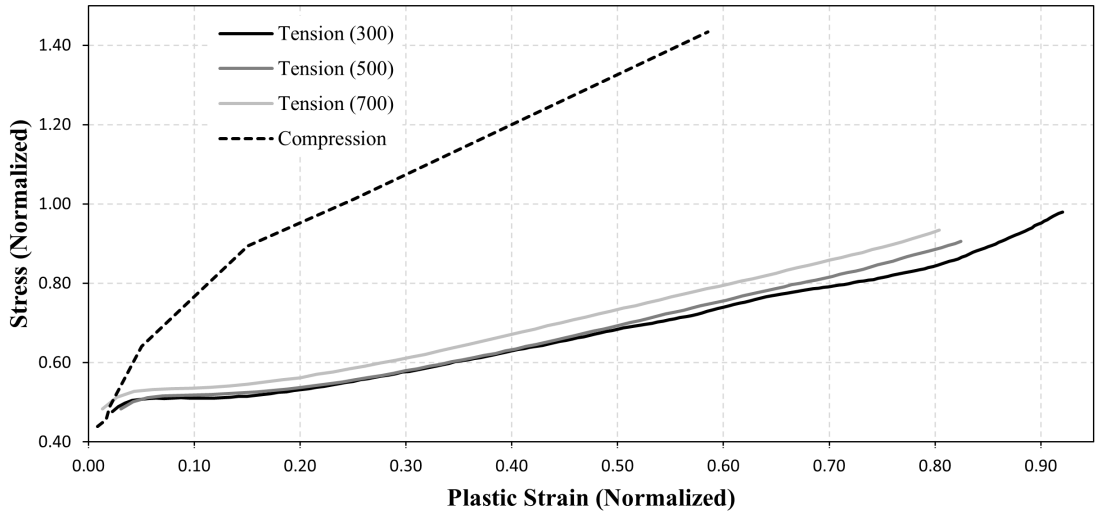


Figure 4.30: Plastic Curves of the Topcoat

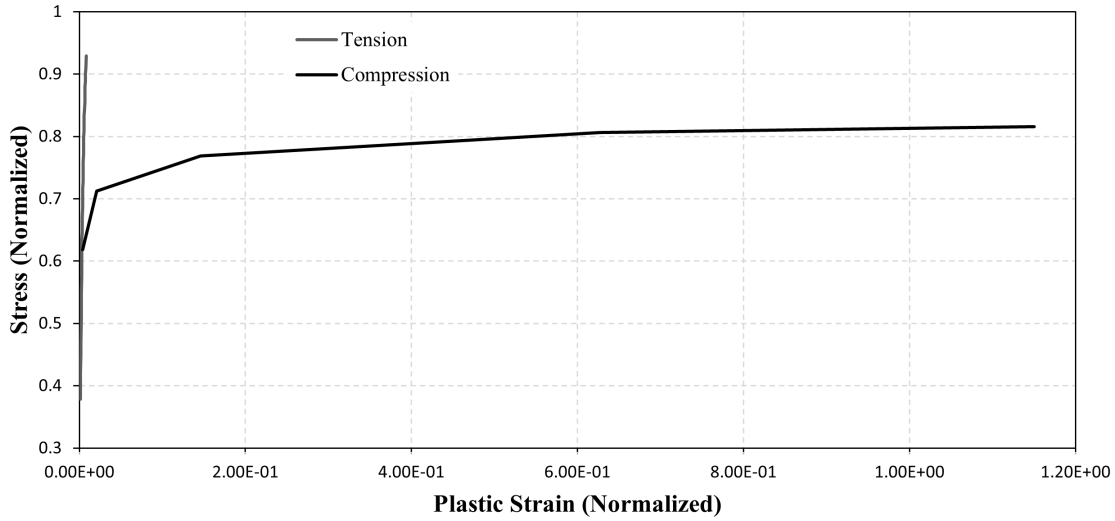


Figure 4.31: Plastic Curves of the Primer

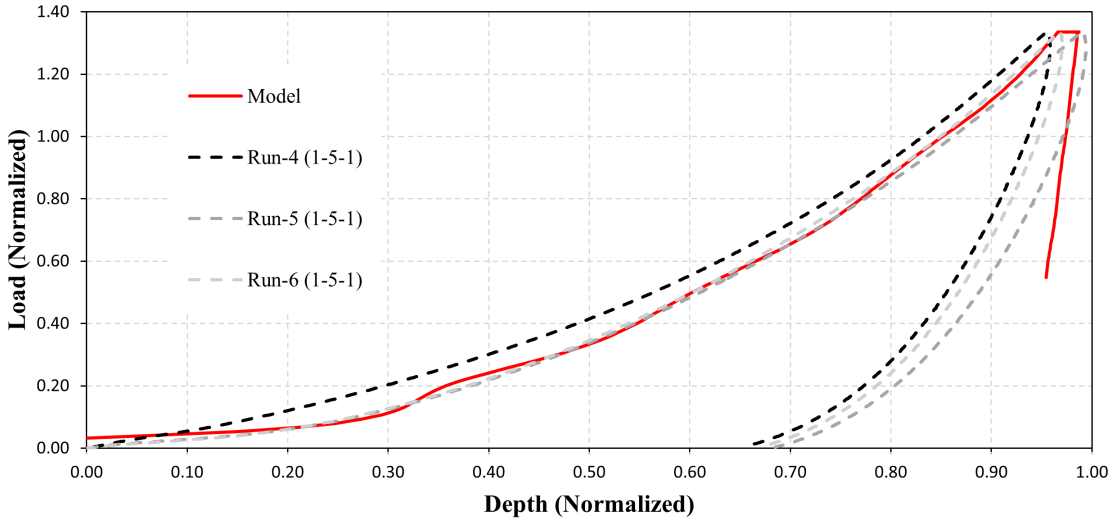


Figure 4.32: Primer Model, Calibrated in Nanoindentation

4.4 Lap Shear

Moving on to lap shear results, we have shear stress versus deflection curves for the Primer by itself (Figure 4.33) as well as the Topcoat over Primer layup (Figure 4.34. The first thing we notice looking at the curves is that the Primer's strength in shear is just under half of that in tension as one would expect. Similarly, the Topcoat over Primer layup fails at just about half of the Topcoat's ultimate strength except for Sample 11 which fell a little short. We also notice that all of the results are slightly inconsistent. The Primer's Samples 2 and 4 have the same stiffness in lap shear and Samples 4 and 5 fail at a similar stress level but the deflections are widely spread. This is also reflected in Figure 4.34. Here, we see the curves all diverge though, they stay close to one another. Samples 9 and 10 have similar failure stresses as do Samples 8 and 12. Then Samples 9 and 11 failed at similar deflections as did Samples 10 and 12. Bearing all of this in mind, these initial lap shear tests are decently repeatable and tell us that both materials have interfacial shear strengths just under half of the true tensile strengths. From the fracture surfaces (images have not yet been released), one may conclude that every Primer sample failed cohesively while each Topcoat over Primer sample exhibits adhesive failure at the substrate.

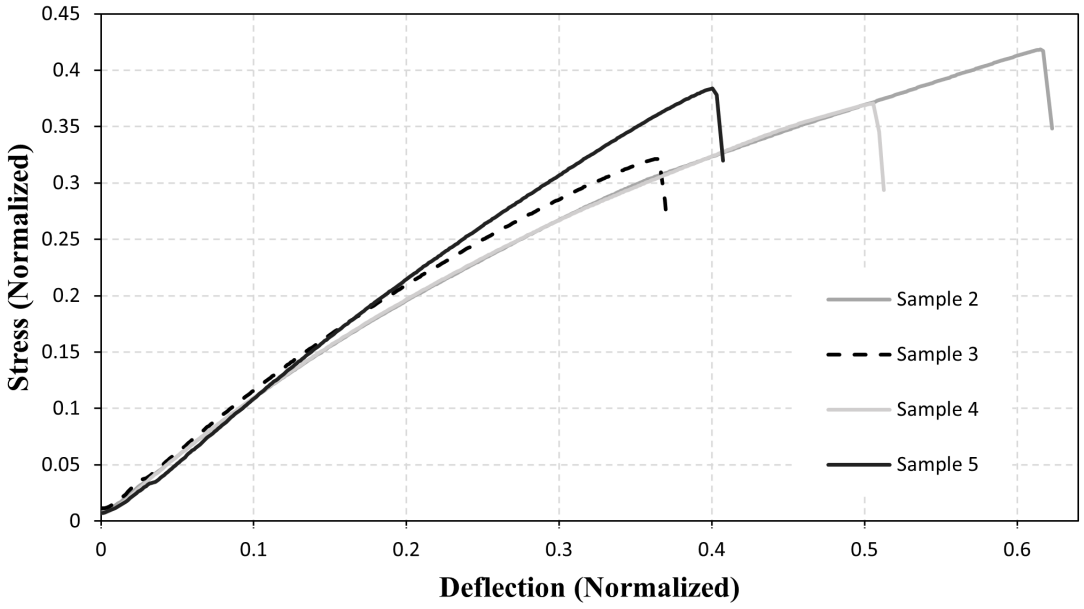


Figure 4.33: Primer in Lap Shear

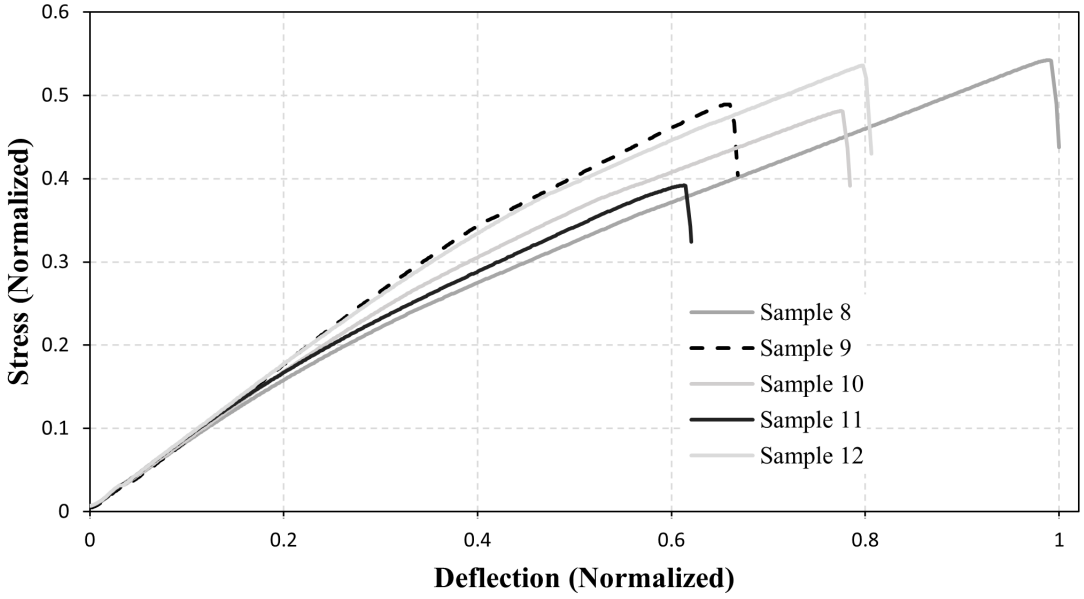


Figure 4.34: Topcoat over Primer in Lap Shear

4.5 Flat Plate Model

Marrying everything together is the Flat Plate Model. This model of the Topcoat over Primer over aluminum substrate layup tells us the material response of the two layer coating system to a rain droplet impact using the applied stresses found with CFD. Looking at Figures 4.35 and We can see small stresses propagating outward from the point of impact as the water droplet moves about the plate. In the final image, the water droplet is gone and these are residual stresses let behind after the impact. We notice that these residual stresses resemble the same shape as the initial impact stresses, simply on a smaller scale. Zooming the fringe out to just under 10% of the Topcoat's true ultimate strength in Figures 4.37 and 4.38, we see that most of the stress waves through the materials are very small.

From all four figures, one can see that the von-Mises stress transfers through the Topcoat to the Primer almost in full meaning that the interface is taking a hefty load which begs the question: "when will the interface fail?" and solidifies that a fatigue study into crack propagation at the interface is an important part of future work. Additionally, this tells us that the Topcoat absorbs some of the impact energy and translates most of it to the Primer.

Continuing with the single droplet impact, the maximum von-Mises stress seen in the plate is just shy of 0.12 (Topcoat normalized) and is only sustained for an instant. Smaller normal and shear stresses exist throughout the simulation following the majority of the droplet impacting shortly after 0.2 ms. As expected, this lower-than-yield force does not cause any plastic deformation in the model and thus the plastic strain remains at zero from start to finish.

Most importantly, these flat plate results demonstrate the capability to model the material response to a (non-spherical) rain droplet impact with FEM. From this, we have the groundwork to create predictive erosion models for any coating system and almost everything we need to model the two layer system from birth to failure.

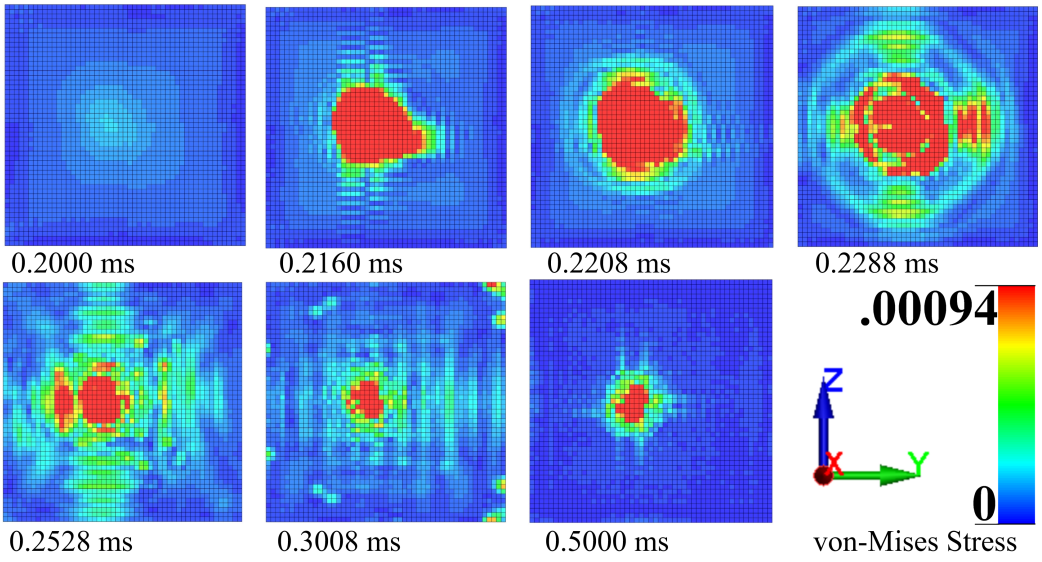


Figure 4.35: von-Mises Stress Over the Topcoat with a Fine Fringe

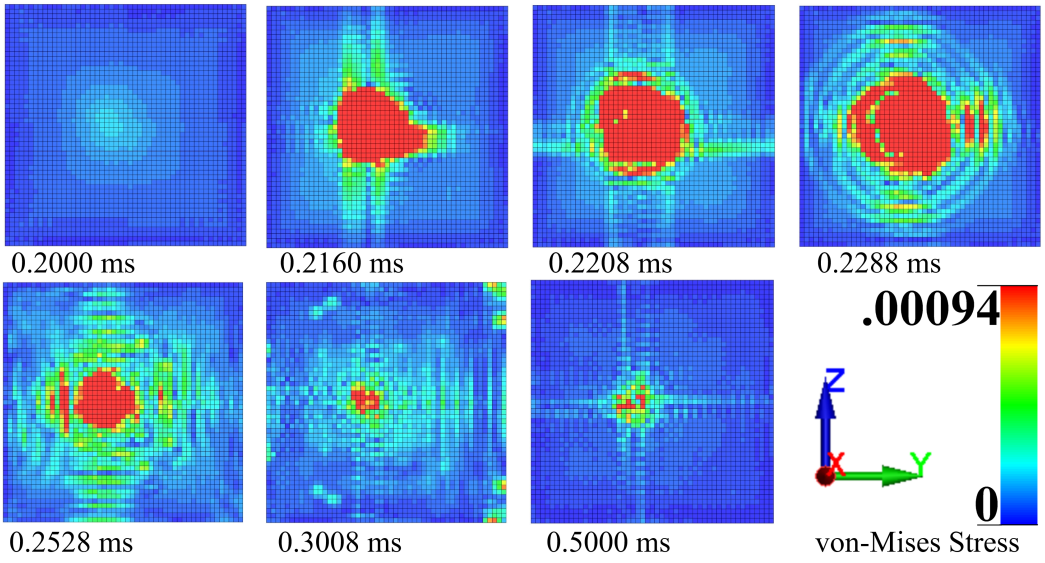


Figure 4.36: von-Mises Stress Over the Primer with a Fine Fringe

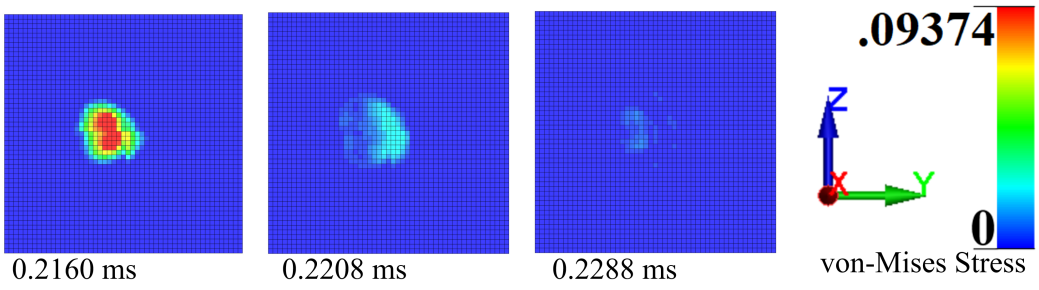


Figure 4.37: von-Mises Stress Over the Topcoat with a Wider Fringe

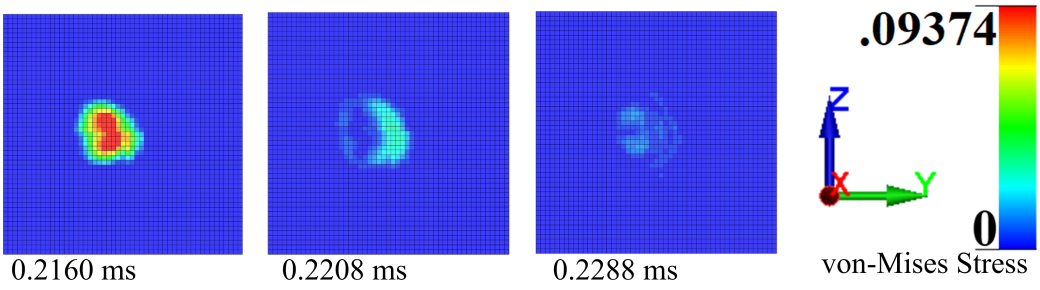


Figure 4.38: von-Mises Stress Over the Primer with a Wider Fringe

Chapter 5

CONCLUSIONS AND OUTCOMES

5.1 Summary of Experimentation and Modelling

5.1.1 WTB Recreation

The project began with the recreation of A.S. Verma et al.'s WTB erosion model. This recreation involved their aluminum plate SPH convergence model as well as single impact full gelcoat model. As aspects that were recreated matched the expected results reasonably well and taught the author some valuable lessons to carry into LS-Dyna.

5.1.2 Tension Testing and Validation

Taking place at Boeing on a TA Discovery HR-2, Tension tests began as cut out rectangles being pulled and the true curves estimated. They evolved into full DIC analysis with several different methods to analyze the strain. DIC testing made use of printer toner to create benign high contrast fields over the surface of each sample tested. From these analyses true stress and strain curves (without the conserved volume assumption) that account for necking were output. Using these, material properties were identified and material models were built and calibrated in LS-Dyna.

5.1.3 Nanoindentation Testing and Recreation

Nanoindentation testing was performed with a Berkovich indenter tip on one of two Bruker machines at the University of Washington. The test machines output a reduced modulus in compression from which the material's Young's modulus is calculated. These tests also give us curves of applied load versus indentation depth. By recreating the tests in LS-Dyna, a stress by plastic strain curve may be determined. Here, the nanoindentation recreation model provides validation through iteration.

5.1.4 Lap Shear Testing

Lap shear tests were performed on a Boeing Instron 5565. From these, we know the Mode II strength of the interface and have a curve that could be iteratively match in a similar process to the nanoindentation validation.

5.1.5 Flat Plate Model

The Flat Plate Model elegantly combines the CFD simulations with material characterization to realize the material response of a coating system to rain droplet impact. In order to apply the CFD results to the FEM model, a comprehensive MATLAB script was developed that maps pressures and shear stresses from the CFD mesh to the FEM mesh. It then outputs a text file to be copied directly into the LS-Dyna keyword file. This model is able to determine the stresses and strains present in each layer from applied surface pressure and shear. It found a decent portion of the droplet's energy is transferred from the Topcoat to the Primer.

5.2 Conclusions Thus Far

This thesis studied the rain erosion of aircraft coating systems with a multi-physical model incorporating novel CFD droplet morphology with tensile characterization and novel plastic curve extraction from nanoindentation. From the testing and modelled discussed herein, the following conclusions are made:

1. Rain droplets do not impact aircraft as spheres. The wave propagation discussed in the introduction still applies as Rayleigh and shear waves propagate from the point of impact.

2. In an impacted two layer coating system, a substantial fraction of the impact energy is transferred to the primer. As such, the interface between the primer and topcoat experiences high stresses.

3. The Flat Plate Model is able to apply the CFD results to actual Boeing materials and predict their response.

4. Most of the materials tested exhibit some form of strain rate dependence which the LS-Dyna models are able to accommodate.

5.3 Sources of Error

It is important to recognize where any noise or error could come from. Potential sources for error in this work include:

1. Measurement error. When measuring the initial cross section of a tension specimen, there exists a possibility of error due to limitations of both the measuring tool itself and of the operator.

2. Difficulty loading brittle samples into the test machine. Extra care had to be taken with the setup of brittle tensile specimens. It is entirely possible that once pure samples could have cracked during setup.

3. Potential micro defects in the dogbone die or thickness variations in the samples. Defects and plastic deformation of samples could result in inconsistent and/or premature failure strains.

4. Plastic deformation of the samples at the edges where the die cuts. This could be responsible for z axis warpage during testing.

5. Assumption error. When creating all of the models used throughout this project, assumptions had to be made. Although every assumption is thought out and justified, they are still assumptions which could be incorrect or too simple.

6. Surface preparation. Some nanoindentation samples had to undergo mild surface preparation as they had too rough of a surface for precise testing.

5.4 Future Work

5.4.1 Continued Material Characterization

Tensile Testing Continuation and Improvements

Future iterations of tensile testing need to include more samples tested at each strain rate for each material. These rates likewise must increase to recreate more realistic service conditions. Another improvement to tensile testing would be a way to measure the out of plane strain to a higher degree of accuracy. As discussed in Section 4.2.2, the samples bowed during testing rendering the out of plane measurements useless. Perhaps this could be accomplished with a 3 or 4 camera setup. Third of all and also related to the cameras, future testing should be performed with a higher camera frame rate. The elastic region of most of these samples had five or less data points which introduces extra uncertainty into calculations like Young's modulus and Poisson's ratio.

Nanoindentation Testing and Modelling Continuation and Improvements

The nanoindentation recreations performed thus far have been on tests with displacements sufficiently deep such that the size effect at the tip radius was insignificant in determining plastic behaviour. In the future, a 3D scan of the indenter tip to precisely determine the geometry will be useful with characterizing materials that do not indent as deeply. This is also important for characterizing the Basecoat at the 1.0 s load rate as a decent portion of that curve falls into a gray area where the tip radius may have a significant effect on the load-displacement curve. Similarly, the Topcoat was more recently tested a higher load of $\frac{4}{3}$ the maximum load that was used to calibrate the model. This higher load resulted in a smaller depth. With this in mind, perhaps the Topcoat's result at a normalized load of 1 could be used to calibrate the indenter tip model at the normalized load of $\frac{4}{3}$. Another future consideration for nanoindentation modelling is the implementation of strain rate dependence. Some materials exhibit sufficient strain rate dependence in compression such that it needs to be considered in the model to fully describe the material. This will involve extracting plastic curves for each strain rate one by one and then combining the curves into one model

to validate the FEM material model.

5.4.2 Interfacial Property Characterization

As well as characterizing the materials themselves, it is important to understand how they interact with each other, this is where we study the interfacial properties. The first step toward understanding the interface is by recreating the lap shear tests in LS-Dyna. This will reveal if the interfacial shear properties accurately describe the model and open the door for curve matching. Another way the interface is to be characterized is by using laser spallation to determine its fracture energy. Laser spallation introduces stress waves into a sample through rapid high energy impacts [30]. It can help to characterize adhesive and interfacial properties with primary advantages of non-contact loading, high stress (GPa scale), and high strain rates [30].

5.4.3 Fatigue Studies

The next major step in this project will be a fatigue study once the interfacial properties are determined. This study will allow for the prediction of coating failure, both in time and mechanism.

5.4.4 The Service Environment

The final future work to discuss with respect to modelling is the service environment. The work in this project so far has only considered mechanical rain erosion with no temperature effects. This is very important in building a baseline erosion model however, the aircraft surfaces of interest see more than ideal rain. Research into the service environment and conditions such as chemically corrosive erosion by acidic rain and temperature effects are forthcoming.

5.4.5 Physical Validation

Finally, upon the completion of a robust failure predicting model, the FEM results will be verified with physical testing before it is used as part of the qualification process.

BIBLIOGRAPHY

[1] Omid Gohardani. Impact of erosion testing aspects on current and future flight conditions. *Progress in Aerospace Sciences*, 47(4):280–303, 2011.

[2] Boeing Commercial Airplane Company. *Aircraft Surface Coatings: Energy Efficient Transport Program*. NASA, 1982.

[3] Amrit Shankar Verma, Saullo G.P. Castro, Zhiyu Jiang, and Julie J.E. Teuwen. Numerical investigation of rain droplet impact on offshore wind turbine blades under different rainfall conditions: A parametric study. *Composite Structures*, 241:112096, 2020.

[4] Antonino Ferrante and Marco Salviato. *Internal Report*. University of Washington, 2022.

[5] JR. GEORGE F. SCHMITT. *FLIGHT TEST-WHIRLING ARM CORRELATION OF RAIN EROSION RESISTANCE OF MATERIALS*. US Air Force, 1968.

[6] Baals, D.D., and Corliss, W.R. *Wind Tunnels of NASA*. NASA, 1981.

[7] K. Pugh and M. M. Stack. Rain erosion maps for wind turbines based on geographical locations: A case study in ireland and britain. *Journal of Bio- and Tribo-Corrosion*, 7(1), January 2021.

[8] Cameron Mackie, David H. Nash, Dean Boyce, Matthew Wright, and Kirsten Dyer. Characterisation of a whirling arm erosion test rig. *2018 Asian Conference on Energy, Power and Transportation Electrification (ACEPT)*, pages 1–6, 2018.

[9] E.F. Tobin, T.M. Young, D. Raps, and O. Rohr. Comparison of liquid impingement results from whirling arm and water-jet rain erosion test facilities. *Wear*, 271(9):2625–2631, 2011. 18th International Conference on Wear of Materials.

[10] Daryl L. Logan. *A First Course in the Finite Element Method*. PWS-KENT, 2nd edition, 1992.

[11] ANSYS LS-Dyna. *LS-Dyna Keyword User's Manual Volume I*. LIVERMORE SOFTWARE TECHNOLOGY (LST), AN ANSYS COMPANY, Livermore, California, 2021.

[12] Mira Tipirneni and Antonino Ferrante. *Internal Report*. University of Washington, 2022.

[13] ASTM D638. *Test Method for Tensile Properties of Plastics*. ASTM International, 2014.

[14] ASTM D1708. *Test Method for Tensile Properties of Plastics by Use of Microtensile Specimens*. ASTM International, 2018.

[15] Comer, J. J., Bannantine, J.A., and Handrock, J. L. *Fundamentals of Metal Fatigue Analysis*. Prentice Hall, 1989.

[16] GOM Metrology. GOM correlate pro.

[17] MATLAB. *version R2022b*. The MathWorks Inc., Natick, Massachusetts, 2022.

[18] Abdulaziz Alaboodi and Zahid Hussain. Finite element modeling of nano-indentation technique to characterize thin film coatings. *Journal of King Saud University*, 31:61–69, 01 2019.

[19] Gaylord Guillonneau, Guillaume Kermouche, Jean-Michel Bergheau, and Jean-Luc Loubet. A new method to determine the true projected contact area using nanoindentation testing. *Comptes Rendus Mécanique*, 343(7):410–418, 2015.

[20] Christian Saringer, Michael Tkadletz, Markus Kratzer, and Megan J. Cordill. Direct determination of the area function for nanoindentation experiments. *Journal of Materials Research*, 36(11):2154–2165, February 2021.

[21] D. Torres-Torres, J. Muñoz-Saldaña, L Guevara, Abel Hurtado-Macias, and Michael Swain. Geometry and bluntness tip effects on elastic–plastic behaviour during nanoindentation of fused silica: Experimental and fe simulation. *Modelling and Simulation in Materials Science and Engineering*, 18:075006, 09 2010.

[22] Suresh Sagadevan and Priya Murugasen. Novel analysis on the influence of tip radius and shape of the nanoindenter on the hardness of materials. *Procedia Materials Science*, 6:1871–1878, 2014. 3rd International Conference on Materials Processing and Characterisation (ICMPC 2014).

[23] W.C. Oliver and G.M. Pharr. An improved technique for determining hardness and elastic modulus using load and displacement sensing indentation experiments. *Journal of Materials Research*, 7(6):1564–1583, June 1992.

[24] Warren Oliver and G. Pharr. Measurement of hardness and elastic modulus by instrumented indentation: Advances in understanding and refinements to methodology. *Journal of Materials Research - J MATER RES*, 19:3–20, 01 2004.

[25] Ankit Rohatgi. Web Plot Digitizer.

[26] Paul Meißner, Jens Winter, and Thomas Vietor. Methodology for neural network-based material card calibration using LS-DYNA MAT_187_SAMP-1 considering failure with GISSMO. *Materials*, 15(2):643, January 2022.

[27] Mo A. Berkovich and flat punch indenter tips, Jan 2015.

[28] ANSYS LS-Dyna. *LS-Dyna Keyword User's Manual Volume II*. LIVERMORE SOFTWARE TECHNOLOGY (LST), AN ANSYS COMPANY, Livermore, California, 2021.

[29] n.a. Bilinear interpolation. *Wikipedia*, May 2023.

[30] Junlan Wang, Richard L. Weaver, and Nancy R. Sottos. A parametric study of laser induced thin film spallation. *Experimental Mechanics*, 42(1):74–83, March 2002.

[31] ANSYS LS-Dyna. *LS-DYNA*. LIVERMORE SOFTWARE TECHNOLOGY (LST), AN ANSYS COMPANY, Livermore, California, 2022.

[32] Karim Gadelrab and Matteo Chiesa. Influence of nanoindenter tip radius on the estimation of the elastic modulus. *MRS Proceedings*, 1297, 01 2011.

Appendix A

CFD SIMULATION

The CFD flat plate simulation is setup as shown in Figure A.1. The left boundary is the velocity inlet which has a uniform velocity of $U_\infty = 156\ m/s$. The right boundary is wall (this is the where the flat plate is) meaning a zero velocity case. The other four boundaries are pressure outlets set to a defined gauge pressure of zero. The model is first solved in Fluent for the steady state solution with no droplet. Once complete, the 2 mm droplet is added to the domain and initialized with U_∞ as well as a small relative velocity representing the terminal velocity of a falling rain droplet [12].

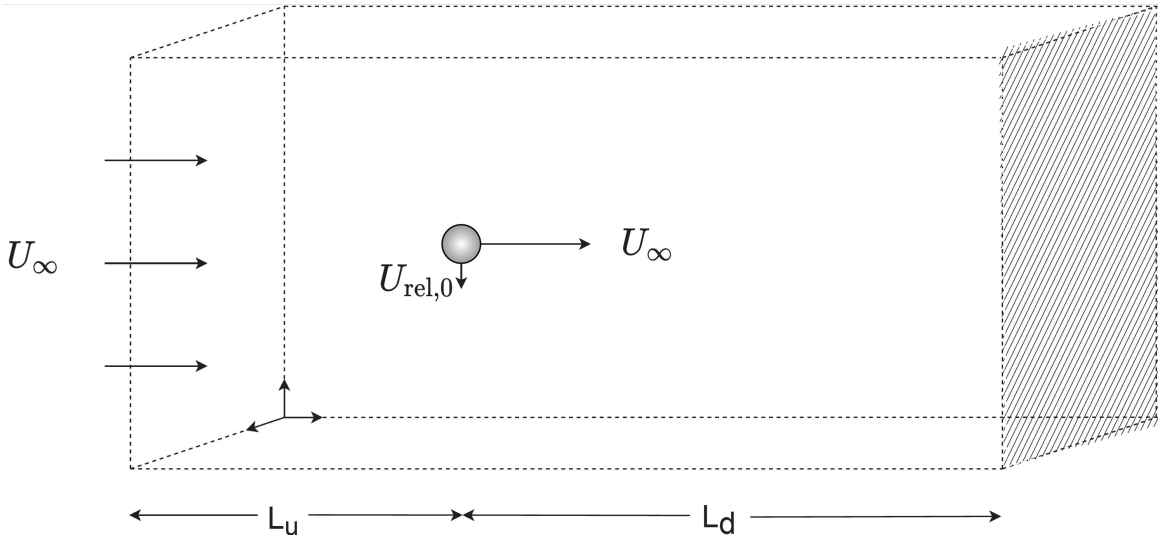


Figure A.1: CFD Initial Conditions [12]

Appendix B

MATLAB CODE

B.1 Material Property Determination

Note: Some of the typed MATLAB characters may not display correctly.

```
DIC - True Sections

Read in Data
Input the desired directory below, best to start from the hadrive so that it
    works in any folder. This code is a constant work in progress and can
    changes based on the testing of different materials. There are several
    commenet-toggled plots throughout to make evaluate intermitten things.
    This code has some room for optimization and some redundancy which is
    accpetable in its current work in progress state. It is also not yet fully
     commented. For any questions please email cman002@uw.edu.
clear all


%% Inputs PART 1. See section 'Material Properties' for more inputs.

sampleNumber = [redacted]; Material = "[redacted]";
directory = [redacted]
directoryOut = [redacted] + Material + "_Output\"

tmil = [redacted] % initial thickness in thousants of an inch
w = 5 % initial sample width in mm
strainScaleFactor = 0.0001 % if needed to go from micro strains to regular
    strains or what have you

calcMethodB = true
shift = true
```

```
overrideE = true; % Use if young's mod is getting weird
Ein = [redacted] % Override value in MPa
usealtE = false; % set true to use E2
% these cannot both be true. At least one must be false

normalizeTo = [redacted] % [stress (MPa), strain]
%% End Inputs

files = dir(directory + "*.csv")

if sampleNumber <10
    sampleWrite = "-0" + sampleNumber
else
    sampleWrite = "-" + sampleNumber
end
sampleName = Material + sampleWrite

t = tmil/1000*25.4 % initial thickness in mm
numFiles = length(files);
names = string({files.name});
file = cell(numFiles-2,1);
L = length(file);
load = zeros(L,1); % initalize vector of load at each state
time = zeros(L,1); % initialize test time vector
dirname = 'directory and filename'; % changes in loop

% specific to format of DIC output files
for i = 2:numFiles-1
    dirname = directory + names(i);
    file{i-1} = table2array(readtable(dirname));
    currentLoad = xlsread(dirname, 1,'A7:A7');
    if isempty(currentLoad)
        currentLoad = 0;
    end
    load(i-1) = currentLoad;
    currentTime = xlsread(dirname, 1,'C4:C4');
```

```
    time(i-1) = currentTime;
end

% Find plane divisions
divLine = zeros(L,3); % index of NaN's or 0's whichever will work best
loopdivLine = [0;0;0]; %initialize
for i = 1:L
   inloopdiv = find((file{i}(:,1)==0));
   if length(inloopdiv) > 3
       disp "Greater than 3"
   elseif length(inloopdiv) < 3
       less1 = "Only two. Look at data to verify problem. It is most likely
           that the top has gone out of frame"
       less2 =  "Since the top is probably out of frame the first zero denotes
            the start of center"
       loopdivLine = [0; inloopdiv];
   else
       loopdivLine = inloopdiv;
   end
   divLine(i,:) = transpose(loopdivLine);
end

Method A: Center, Îţx = Îţz
For this method we assume the strain in x and z is the same. We take the DIC
    points from the center to calculate strain in the center and instantaneous
     area in the center of the sample. Strain in y is taken as average across
    the center, strain in x is taken as the average acorss the center as well,
     and strain in z is taken as the same as x
%% Reduce To Center Plane
centerStrain = cell(L,1);
centerCoords = cell(L,1);
for i = 1:L
    centerStrain{i} = file{i}([divLine(i,2):(divLine(i,3)-2)],[6,7])*
        strainScaleFactor;
    centerCoords{i} = file{i}([divLine(i,2):(divLine(i,3)-2)],[2:4]);
end
```

```
%% Create Strain Curve
% For average strain in the cross section, we calculate the average strain
% at each stage. This average at each stage is then used to generate a
% strain curve based on average strain across the center of the sample. The
% max and min are compared as well as the standard deviation to evaluate if
% averaging across the center of the sample is useful
% this is performed for both x and y

% initialize variables
ystrain = zeros(L,1);
ymaxes = zeros(L,1);
ymins = zeros(L,1);
ystdevs = zeros(L,1);
ypercentdiffs = zeros(L,1);

xstrain = zeros(L,1);
xmaxes = zeros(L,1);
xmins = zeros(L,1);
xstdevs = zeros(L,1);
xpercentdiffs = zeros(L,1);
for i = 1:L
    ystrain(i) = mean(centerStrain{i}(:,2));
    ymaxes(i) = max(centerStrain{i}(:,2));
    ymins(i) = min(centerStrain{i}(:,2));
    ystdevs(i) = std(centerStrain{i}(:,2));
    ypercentdiffs(i) = (ymaxes(i)-ymins(i))/ystrain(i);

    xstrain(i) = mean(centerStrain{i}(:,1));
    xmaxes(i) = max(centerStrain{i}(:,1));
    xmins(i) = min(centerStrain{i}(:,1));
    xstdevs(i) = std(centerStrain{i}(:,1));
    xpercentdiffs(i) = (xmaxes(i)-xmins(i))/xstrain(i);
end
yreview = vpa([ystrain, ystdevs, ymaxes, ymins, ypercentdiffs],4)
xreview = vpa([xstrain, xstdevs, xmaxes, xmins, xpercentdiffs],4)
```

```
datum = find((load~=0), 1, 'first') -1
if isempty(datum)
    datum = 1
end
if isnan(ystrain(datum))
    shift = false
end
if shift == true
% Shift strain to (0,0)
d1plus = false
if ystrain(datum) == 0
    if ystrain(datum+1) == 0
        dplus1 = true
        while dplus1 == true
            warning('y-strain at datum + 1 = 0')
            ystrainOld = ystrain
            for i = 1:length(ystrain)-1
                ystrain(i) = ystrainOld(i+1)
            end
            ystrain(length(ystrain)) = 0
            clear ystrainOld
            if ystrain(datum+1) ~= 0
                dplus1 = false
            end
        end
    else
    end
else
    ystrain = ystrain - ystrain(datum)
end
if xstrain(datum) == 0
    if xstrain(datum+1) == 0
        dplus1 = true
        while dplus1 == true
            warning('x-strain at datum + 1 = 0')
            xstrainOld = xstrain
```

```
            for i = 1:length(xstrain)-1
                xstrain(i) = xstrainOld(i+1)
            end
            xstrain(length(xstrain)) = 0
            clear xstrainOld
            if xstrain(datum+1) ~= 0
                dplus1 = false
            end
        end
    else
    end
else
    xstrain = xstrain - xstrain(datum)
end
end

%% Instant Area
% using x strains calculated above and assuming z strains to be the same
Area = zeros(L,1);
stress = zeros(L,1);
A0 = t*w % mm^2
for i = 1:L
    Area(i) = A0*exp(xstrain(i)+xstrain(i));
    stress(i) = abs(load(i)/Area(i));
end

% Write to file
fileOut = sampleName + "_Method_A_ÏČ–Îţ_and_t.csv";
writematrix([ystrain, stress, time],directoryOut+fileOut);
close all

plot(ystrain,stress)
title(sampleName + " ÏČ–Îţ")
ylabel("ÏČ (MPa)")
xlabel("Îţ (mm/mm)")
grid on
```

```
xlim([0 max(ystrain)*1.05])
ylim([0 max(stress)*1.05])


ystrainA = ystrain;
xstrainA = xstrain;
stressA = stress;
poissonA = -xstrainA./ystrainA

Method B: Center, average dz
This method is the same as Method A excpet we do not assume the strain in x
    and z are equal. Instead, we use the average dz across the cross section
    to calculate z strain and instantaneous area. dz is found using z
    coordinate data. As a sanity check, Îţz should be very close to Îţx if the
     data and code are both good. The strain matrices from Method A will be
    used in this and so it should be run as well.
if calcMethodB == true
    xstrainB = xstrain
    dzcenter = zeros(L,1);
    zinst = zeros(L,1);
    zstrain = zeros(L,1);
    zref = mean(centerCoords{datum}(:,3))
    for i = 1:L
        dzcenter(i) = 2*(mean(centerCoords{i}(:,3)) - zref);
    end
    plot(dzcenter)
    ylim([-1,1])
    shift = dzcenter(datum+1)/tmil*2
    zstrain_eng = -dzcenter/tmil+shift
    hold off

    zinst(datum) = tmil
    for i = 1+datum:L
        zinst(i) = zinst(i-1) - (dzcenter(i)-dzcenter(i-1));
        if zinst(i)>zinst(datum)%zinst(i) > zinst(i-1)
            zinst(i) = zinst(datum)%zinst(i) = zinst(i-1);
```

```
        end
        zstrain(i) = zstrain(i-1) + log(zinst(i)/zinst(i-1));%-dzcenter(i)/
            zinst(i-1);
    end

    figure(4)
    plot(time,zstrain)
    hold on
    plot(time,zstrain_eng)
    plot(time,xstrain)
    legend("\epsilon_z_z","z_{eng}","\epsilon_x_x")
    ylim([redacted])
    hold off

% Shift strain to (0,0)
if shift == true
d1plus = false
if ystrain(datum) == 0
    if ystrain(datum+1) == 0
        dplus1 = true
        while dplus1 == true
            warning('y-strain at datum + 1 = 0')
            ystrainOld = ystrain
            for i = 1:length(ystrain)-1
                ystrain(i) = ystrainOld(i+1)
            end
            ystrain(length(ystrain)) = 0
            clear ystrainOld
            if ystrain(datum+1) ~= 0
                dplus1 = false
            end
        end
    else
    end
else
    ystrain = ystrain - ystrain(datum)
```

```
end
if xstrainB(datum) == 0
    if xstrainB(datum+1) == 0
        dplus1 = true
        while dplus1 == true
            warning('x-strain at datum + 1 = 0')
            xstrainOld = xstrainB
            for i = 1:length(xstrainB)-1
                xstrainB(i) = xstrainOld(i+1)
            end
            xstrainB(length(xstrainB)) = 0
            clear xstrainOld
            if xstrainB(datum+1) ~= 0
                dplus1 = false
            end
        end
    else
    end
else
    xstrainB = xstrainB - xstrainB(datum)
end
if zstrain(datum) == 0
    if zstrain(datum+1) == 0
        dplus1 = true
        while dplus1 == true
            warning('z-strain at datum + 1 = 0')
            zstrainOld = zstrain
            for i = 1:length(zstrain)-1
                zstrain(i) = zstrainOld(i+1)
            end
            zstrain(length(zstrain)) = 0
            clear zstrainOld
            if zstrain(datum+1) ~= 0
                dplus1 = false
            end
        end
```

```
    else
    end
else
    zstrain = zstrain - zstrain(datum)
end
end

    %% Instant Area
    % using x strains calculated above and assuming z strains to be the same
    Area = zeros(L,1);
    stressB = zeros(L,1);
    A0 = t*w % mm^2
    for i = 1:L
        Area(i) = A0*exp(xstrainB(i)+zstrain(i));
        stressB(i) = abs(load(i)/Area(i));
    end
    hold off
    figure(5)
    %plot(ystrain,stressB)
    ystrainB = ystrain; zstrainB = zstrain;
    poissonB = -xstrainB./ystrainB
    poissonBz = -zstrainB./ystrainB
  % plot(ystrainB,stressB)
else
    ystrainB = zeros(L,1)
    stressB = zeros(L,1)
end

Method C: Center, Bottom, and Top when there; Îţx = Îţz

bottomStrain = cell(L,1);
bottomCoords = cell(L,1);
topStrain = cell(L,1);
topCoords = cell(L,1);
for i = 1:L
    fileEnd = length(file{i});
```

```
    if divLine(i,1) == 0
        bottomStrain{i} = file{i}([divLine(i,3):(fileEnd)],[6,7])*
            strainScaleFactor;
        bottomCoords{i} = file{i}([divLine(i,3):(fileEnd)],[2:4]);
    else
        bottomStrain{i} = file{i}([divLine(i,3):(fileEnd)],[6,7])*
            strainScaleFactor;
        bottomCoords{i} = file{i}([divLine(i,3):(fileEnd)],[2:4]);
        topStrain{i} = file{i}([divLine(i,1):(divLine(i,2)-2)],[6,7])*
            strainScaleFactor;
        topCoords{i} = file{i}([divLine(i,1):(divLine(i,2)-2)],[2:4]);
    end
end
(topStrain)
(bottomStrain)

% initialize variables
clear ystrain xstrain
ystrain = zeros(L,1);
xstrain = zeros(L,1);
for i = 1:L
    % use the length of each vector to create a weight so that each
    % point is evenly evaluated
    w1 = length(bottomStrain{i});
    w2 = length(centerStrain{i});

    str1y = mean(bottomStrain{i}(:,2));
    str2y = mean(centerStrain{i}(:,2));
    str1x = mean(bottomStrain{i}(:,1));
    str2x = mean(centerStrain{i}(:,1));
    if divLine(i,1) == 0
        tot = w1+w2;
        W1 = w1/tot;
        W2 = w2/tot;

        ystrain(i) = str1y*W1+str2y*W2;
```

```
        xstrain(i) = str1x*W1+str2x*W2;
    else
        w3 = length(topStrain{i});
        tot = w1+w2+w3;
        W1 = w1/tot;
        W2 = w2/tot;
        W3 = w3/tot;
        str3y = mean(topStrain{i}(:,2));
        str3x = mean(topStrain{i}(:,1));

        ystrain(i) = str1y*W1+str2y*W2+str3y*W3;
        xstrain(i) = str1x*W1+str2x*W2+str3x*W3;
    end
end

% Shift strain to (0,0)
if shift == true
d1plus = false
if ystrain(datum) == 0
    if ystrain(datum+1) == 0
        dplus1 = true
        while dplus1 == true
            warning('y-strain at datum + 1 = 0')
            ystrainOld = ystrain
            for i = 1:length(ystrain)-1
                ystrain(i) = ystrainOld(i+1)
            end
            ystrain(length(ystrain)) = 0
            clear ystrainOld
            if ystrain(datum+1) ~= 0
                dplus1 = false
            end
        end
    else
    end
else
```

```
    ystrain = ystrain - ystrain(datum)
end
if xstrain(datum) == 0
    if xstrain(datum+1) == 0
        dplus1 = true
        while dplus1 == true
            warning('x-strain at datum + 1 = 0')
            xstrainOld = xstrain
            for i = 1:length(xstrain)-1
                xstrain(i) = xstrainOld(i+1)
            end
            xstrain(length(xstrain)) = 0
            clear xstrainOld
            if xstrain(datum+1) ~= 0
                dplus1 = false
            end
        end
    else
    end
else
    xstrain = xstrain - xstrain(datum)
end
end

%% Instant Area
% using x strains calculated above and assuming z strains to be the same
Area = zeros(L,1);
stress = zeros(L,1);
A0 = t*w % mm^2
for i = 1:L
    Area(i) = A0*exp(xstrain(i)+xstrain(i));
    stress(i) = abs(load(i)/Area(i));
end

% Write to file
fileOut = sampleName + "_Method_C_ÏČ–Îţ_and_t.csv";
```

```
writematrix([ystrain, stress, time],directoryOut+fileOut);
close all

plot(ystrain,stress)
ystrainC = ystrain;
xstrainC = xstrain;
stressC = stress;

poissonC = -xstrainC./ystrainC

% Percent Difference of C from A
AC_strainDiff = (ystrainA-ystrainC)./ystrainA*100


Method D: Strain Over Sample, top to bottom
This one will use coordinates instead of strain to calculate the strain
    between the top and bottom planes in Y and compare that to center and such
    .
% measure each coordinate (top and bottom) and subtract dat shit. IF top
% gets cut off, use middle mayhaps
ygap0 = mean(topCoords{datum}(:,2))-mean(bottomCoords{datum}(:,2))
xgapTB0 = 0.5*(max(topCoords{datum}(:,1))+max(bottomCoords{datum}(:,1)))- ...
    0.5*(min(topCoords{datum}(:,1))+min(bottomCoords{datum}(:,1)))
xgapC0 = max(centerCoords{datum}(:,1)) - min(centerCoords{datum}(:,1))
ystrainD = zeros(L,1); xstrainDC = zeros(L,1); xstrainDTB = zeros(L,1);
ygap = zeros(L,1); xgapTB = zeros(L,1); xgapC = zeros(L,1);
ygap(datum) = ygap0; xgapTB(datum) = xgapTB0; xgapC(datum) = xgapC0;
ystrain_eng = zeros(L,1)

%ystrainD(datum+1) = (-ygap0 + (mean(topCoords{datum+1}(:,2))-mean(
    bottomCoords{datum+1}(:,2))))/ygap0
for i = datum+1:L
    if isempty(topCoords{i}) == true
        break
    end
    ygap(i) = mean(topCoords{i}(:,2))-mean(bottomCoords{i}(:,2));
```

```
    dy = ygap(i) - ygap(i-1);
    %ystrainD(i) = ystrainD(i-1) + dy/ygap(i-1);
    ystrainD(i) = ystrainD(i-1) + log(ygap(i)/ygap(i-1));
    xgapC(i) = max(centerCoords{i}(:,1)) - min(centerCoords{i}(:,1));
    xgapTB(i) =  0.5*(max(topCoords{i}(:,1))+max(bottomCoords{i}(:,1)))- ...
    0.5*(min(topCoords{i}(:,1))+min(bottomCoords{i}(:,1)));
    dxC = xgapC(i) - xgapC(i-1); dxTB = xgapTB(i) - xgapTB(i-1);
    xstrainDC(i) = xstrainDC(i-1) + log(xgapC(i)/xgapC(i-1));
    xstrainDTB(i) = xstrainDTB(i-1) + log(xgapTB(i)/xgapTB(i-1));
    ystrain_eng(i) = dy/ygap0 + ystrain_eng(i-1);
end
xstrainDCTB = (1/3)*xstrainDC + (2/3)*xstrainDTB;
zgap0 = tmil/2
zgap = zeros(L,1); zgap(datum) = zgap0
zlow = zeros(L,1); zmed = zeros(L,1); zhigh = zeros(L,1); z30 = zeros(L,1);
zref = mean(centerCoords{datum}(:,3))
for i = datum+1:L
    zgap(i) = zgap0 -(zref - mean(centerCoords{i}(:,3)));
    zhigh(i) = max(centerCoords{i}(:,3));
    zmed(i) = mean(centerCoords{i}(:,3));
    zlow(i) = min(centerCoords{i}(:,3));
    z30(i) = (centerCoords{i}(30,3));
end

% Shift strain to (0,0)
if shift == true
d1plus = false
if ystrainD(datum) == 0
    if ystrainD(datum+1) == 0
        dplus1 = true
        while dplus1 == true
            warning('y-strain at datum + 1 = 0')
            ystrainDOld = ystrainD
            for i = 1:length(ystrainD)-1
                ystrainD(i) = ystrainDOld(i+1)
            end
```

```
            ystrainD(length(ystrainD)) = 0
            clear ystrainDOld
            xstrainDCOld = xstrainDC
            for i = 1:length(xstrainDC)-1
                xstrainDC(i) = xstrainDCOld(i+1)
            end
            xstrainDC(length(xstrainDC)) = 0
            clear xstrainCOld
            xstrainDTBOld = xstrainDTB
            for i = 1:length(xstrainDTB)-1
                xstrainDTB(i) = xstrainDTBOld(i+1)
            end
            xstrainDTB(length(xstrainDTB)) = 0
            clear xstrainDTBOld
            if ystrainD(datum+1) ~= 0
                dplus1 = false
            end
        end
    else
    end
else
    ystrainD = ystrainD - ystrainD(datum)
    xstrainDC = xstrainDC - xstrainDC(datum)
    xstrainDTB = xstrainDTB - xstrainDTB(datum)
end
end

%% Instant Area
% using x strains calculated above and assuming z strains to be the same
AreaC = zeros(L,1);
stressDC = zeros(L,1);
AreaTB = zeros(L,1);
stressDTB = zeros(L,1);
AreaCTB = zeros(L,1);
stressDCTB = zeros(L,1)
A0 = t*w % mm^2
```

```
for i = 1:L
    AreaC(i) = A0*exp(xstrainDC(i)+xstrainDC(i));
    stressDC(i) = abs(load(i)/AreaC(i));
    AreaTB(i) = A0*exp(xstrainDTB(i)+xstrainDTB(i));
    stressDTB(i) = abs(load(i)/AreaTB(i));
    AreaCTB(i) = A0*exp(xstrainDCTB(i)+xstrainDCTB(i));
    stressDCTB(i) = abs(load(i)/AreaCTB(i));
end

poissonDC = -xstrainDC./ystrainD
poissonDTB = -xstrainDTB./ystrainD
poissonDCTB = -xstrainDCTB./ystrainD

% Write to file
fileOut = sampleName + "_Method_D-C_ÏČ-Îţ_and_t.csv";
writematrix([ystrainD, stressDC, time],directoryOut+fileOut);
close all
fileOut = sampleName + "_Method_D-TB_ÏČ-Îţ_and_t.csv";
writematrix([ystrainD, stressDTB, time],directoryOut+fileOut);
close all

plot(ystrainD,stressDC)
hold on
plot(ystrainD,stressDTB)
plot(ystrainD,stressDCTB)
title(sampleName + " ÏČ-Îţ")
ylabel("ÏČ (MPa)")
xlabel("Îţ (mm/mm)")
grid on
xlim([0 max(ystrain)*1.05])
ylim([0 max(stress)*1.05])
hold off
```

Method E: Point by Point Strain Over Sample, Top to Bottom

```
Method E is similar to method D and in fact, is the same in the x direction
    strain as DC but the y direction takes individual points across the
    section, calcualtes the sttrain using those two points, and then
    calculates the average at the end.
% First, we find the minimum difference between coordinate points at the
% top and bottom of the sample
corrBots = dsearchn(topCoords{datum+1}(:,1),bottomCoords{datum+1}(:,1))
if isempty(topCoords{L-3})
    empty = true;
    iii = 4
    while empty == true
        if isempty(topCoords{L-iii})
            iii = iii + 1
        else
            empty = false
        end
    end
    iii = iii + 7
else
    iii = 3
end
corrBotsEnd = dsearchn(topCoords{L-iii}(:,1),bottomCoords{L-iii}(:,1))
% Next, we pick ten points to use spaced evenly
pointsDesired = 10;
pointsAcross = length(corrBots)
searching = true;
j = 0;
offsetAcross = floor(pointsAcross/pointsDesired);
nFound = 0;
found = zeros(pointsDesired,1);
foundOne = false;
trySome = zeros(offsetAcross,2);
trySomeIndex = zeros(offsetAcross,2);
tryCount = 0;
while searching == true
    if foundOne == true
```

```
        j = nFound*offsetAcross;
    else
        j = j+1;
    end
    if corrBots(j) == corrBotsEnd(j)
        foundOne = true;
        nFound = nFound+1;
        found(nFound) = j;
        trySome = trySome*0;
        tryCount = 0;
    else
        tryCount = tryCount+1
        initialDiff1 = bottomCoords{datum+1}(j,1) - topCoords{datum+1}(
            corrBots(j),1);
        finalDiff1 = bottomCoords{L-iii}(j,1) - topCoords{L-iii}(corrBots(j)
            ,1);
        initialDiff2 = bottomCoords{datum+1}(j,1) - topCoords{datum+1}(
            corrBotsEnd(j),1);
        finalDiff2 = bottomCoords{L-iii}(j,1) - topCoords{L-iii}(corrBotsEnd(j
            ),1);
        foundOne = false;
        trySome(tryCount,1) = abs(finalDiff1 - initialDiff1);
        trySome(tryCount,2) = abs(finalDiff2 - initialDiff2);
        trySomeIndex(tryCount,1) = j;
        trySomeIndex(tryCount,2) = corrBotsEnd(j);
        if tryCount == length(trySome)
            foundOne = true;
            nFound = nFound+1;
            useVal = min(trySome,[],"all");
            use = find(trySome == useVal,1);
            found(nFound) = trySomeIndex(use);
            trySome = trySome*0;
            tryCount = 0;
        end
    end
    if nFound >= pointsDesired
```

```
        searching = false
    elseif j >pointsAcross-1
        searching = false
    end
end
clear j

% Using x strain from Method DC, we may calculate y below:

ygap0 = topCoords{datum}(found,2) - bottomCoords{datum}(found,2)
xstrainE = xstrainDC
ystrainE_ALL = zeros(L,length(found));
ygap = zeros(L,length(found));
ygap(datum,:) = ygap0;
ystrain_eng = zeros(L,1)

for i = datum+1:L-iii+3
    if isempty(topCoords{i})
        break
    end
    ygap(i,:) = topCoords{i}(found,2) - bottomCoords{i}(found,2);
    dy = ygap(i,:) - ygap(i-1,:);
    ystrainE_ALL(i,:) = ystrainE_ALL(i-1,:) + log(ygap(i,:)./ygap(i-1,:));
  % ystrain_eng(i) = dy/ygap0 + ystrain_eng(i-1);
end

stressE = stressDC % bc the stress is independant of y strain!
ystrainE = mean(ystrainE_ALL,2)

poissonE_full = -xstrainE./ystrainE_ALL
poissonE = mean(poissonE_full,2)
% Write to file
fileOut = sampleName + "_Method_E_ÏČ-Îţ_and_t.csv";
writematrix([ystrainE_ALL, stressE, time],directoryOut+fileOut);
close all
```

```
plot(ystrainE_ALL,stressE)
hold on
title(sampleName + " ÏČ–Îţ")
ylabel("ÏČ (MPa)")
xlabel("Îţ (mm/mm)")
grid on
xlim([0 max(ystrain)*1.05])
ylim([0 max(stress)*1.05])
hold off


Method F: All Average
This is a hybrid of Methods B and C. Method F used the average strains output
    by GOM in x andy and y across the sample as well as calcuating and
    averaging the strain in z at the top, middle and bottom. Sime Method F
    used the same x and y strains as Method C, those; along with center Z
    strain from Method B are brout directly in:
ystrainF = ystrainC; xstrainF = xstrainC; zstrainCenter = zstrainB

dztop = zeros(L,1);
zinstTop = zeros(L,1);
zstrainTop = zeros(L,1);
zrefTop = mean(topCoords{datum}(:,3))
dzbottom = zeros(L,1);
zinstBottom = zeros(L,1);
zstrainBottom = zeros(L,1);
zrefBottom = mean(bottomCoords{datum}(:,3))
for i = 1:L-iii+3
    if isempty(topCoords{i})
        break
    end
    dztop(i) = 2*(mean(topCoords{i}(:,3)) - zrefTop);
    dzbottom(i) = 2*(mean(bottomCoords{i}(:,3)) - zrefBottom);
end

zinstTop(datum) = tmil
```

```
zinstBottom(datum) = tmil
for i = 1+datum:L-iii+3
    if isempty(topCoords{i})
        break
    end
    zinstTop(i) = zinstTop(i-1) - (dztop(i)-dztop(i-1));
    if zinstTop(i)>zinstTop(datum)
        zinstTop(i) = zinstTop(datum)
    end
    zstrainTop(i) = zstrainTop(i-1) + log(zinstTop(i)/zinstTop(i-1));
    zinstBottom(i) = zinstBottom(i-1) - (dzbottom(i)-dzbottom(i-1));
    if zinstBottom(i)>zinstBottom(datum)
        zinstBottom(i) = zinstBottom(datum)
    end
    zstrainBottom(i) = zstrainBottom(i-1) + log(zinstBottom(i)/zinstBottom(i
        -1));
end

% Shift strain to (0,0)
if shift == true
d1plus = false
if zstrainTop(datum) == 0
    if zstrainTop(datum+1) == 0
        dplus1 = true
        while dplus1 == true
            warning('z-strain at datum + 1 = 0')
            zstrainOldT = zstrainTop
            zstrainOldB = zstrainBottom
            for i = 1:length(zstrain)-1
                zstrainTop(i) = zstrainOldT(i+1)
                zstrainBottom(i) = zstrainOldB(i+1)
            end
            zstrainTop(length(zstrainTop)) = 0
            zstrainBottom(length(zstrainBottom)) = 0
            clear zstrainOld
            if zstrainTop(datum+1) ~= 0
```

```
                dplus1 = false
            end
        end
    else
    end
else
    zstrainTop = zstrainTop - zstrainTop(datum)
    zstrainBottom = zstrainBottom - zstrainBottom(datum)
end
end
zstrain = (1/3)*(zstrainTop+zstrainBottom+zstrainCenter)
figure(4)
plot(time,zstrainCenter)
hold on
plot(time,zstrainTop)
plot(time,zstrainBottom)
legend("center","top","bottom")
ylim([-.8,.2])
hold off
```

```
%% Instant Area
% using x strains calculated above and assuming z strains to be the same
Area = zeros(L,1);
stressF = zeros(L,1);
A0 = t*w % mm^2
for i = 1:L-iii+3
    Area(i) = A0*exp(xstrain(i)+zstrain(i));
    stressF(i) = abs(load(i)/Area(i));
end
hold off
plot(ystrainF,stressF)
poissonF = -xstrainF./ystrainF
zstrainF = zstrain
```

```
All Methods ÏČ–Îţ Curves in One File
% Regular Curves
fileOut = "!" + sampleName + "_A-B-C-DC-DTB-E-F_All-Together.csv";

writematrix([time, ystrainA, stressA, ystrainB, stressB, ystrainC, stressC ...
             ystrainD, stressDC, ystrainD, stressDTB, ystrainE, stressE, ...
             ystrainF, stressF],directoryOut+fileOut);
close all

plot(ystrainA,stressA)
hold on
plot(ystrainB,stressB)
plot(ystrainC,stressC)
plot(ystrainD,stressDC)
plot(ystrainD,stressDTB)
plot(ystrainE,stressE)
plot(ystrainF,stressF)
title(sampleName + " ÏČ–Îţ")
ylabel("ÏČ (MPa)")
xlabel("Îţ (mm/mm)")
legend("Method A", "Method B", "Method C", "Method D-C", ...
    "Method D-TB", "Method E",Location="best")
grid on
xlim([0 max(ystrainA)*1.05])
ylim([0 max(stressA)*1.05])
hold off

% Normalized Curves
ns = normalizeTo(1)
ne = normalizeTo(2)
fileOut = "!" + sampleName + "_All_Curves_Normalized.csv";

writematrix([time, ystrainA/ne, stressA/ns, ystrainB/ne, stressB/ns, ystrainC/
    ne, stressC/ns ...
```

```
            ystrainD/ne, stressDC/ns, ystrainD/ne, stressDTB/ns, ystrainE/ne,
                stressE/ns, ...
            ystrainF/ne, stressF/ns],directoryOut+fileOut);
close all

Material Properties
This uses calculations from the above section in finding poisson's ratio,
    young's modulus, failure strain, yield stress, and actual strain rate.
%% INPUTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Part 2
useMethod = "ALL";
Rsqcrit = 0.98;
offset = 0;
offset0 = offset;
guess = datum+offset+1;
critStart = 5;
% Modulus from origin using R^2
methodCount = 1;
allMethods = ["A";"B";"C";"DC";"DTB";"E";"F"]
if useMethod == "ALL"
    finalCount = length(allMethods)
    usingAll = true;
else
    finalCount = 1
    usingAll = false;
end
for ii = methodCount:finalCount
if usingAll == true
    useMethod = allMethods(ii)
end
if useMethod == "A"
    stress = stressA; ystrain = ystrainA;
    xstrain = xstrainA; zstrain = xstrainA;
    methodCall = " Method_A"
elseif useMethod == "B"
    stress = stressB; ystrain = ystrainB;
    xstrain = xstrainB; zstrain = zstrainB;
```

```
    methodCall = " Method_B"
elseif useMethod == "C"
    stress = stressC; ystrain = ystrainC;
    xstrain = xstrainC; zstrain = xstrainC;
    methodCall = " Method_C"
elseif useMethod == "DTB"
    stress = stressDTB; ystrain = ystrainD;
    xstrain = xstrainDTB; zstrain = xstrainDTB;
    methodCall = " Method_DTB"
elseif useMethod == "DC"
    stress = stressDC; ystrain = ystrainD;
    xstrain = xstrainDC; zstrain = xstrainDC;
    methodCall = " Method_DC"
elseif useMethod == "E"
    stress = stressE; ystrain = ystrainE;
    xstrain = xstrainE; zstrain = xstrainE;
    methodCall = " Method_E"
elseif useMethod == "F"
    stress = stressF; ystrain = ystrainF;
    xstrain = xstrainF; zstrain = zstrainF;
    methodCall = " Method_F"
end
stresssq = stress.^2;
ystrainsq = ystrain.^2;
stressystrain = stress.*ystrain;
upperLimit = guess; % initialize
start = datum + offset;
attempt = 0
Rsq = 1; % initialize
while Rsq > Rsqcrit
    attempt = attempt + 1;
    upperLimit = upperLimit + 1;
    bestfit = fitlm(ystrain(start:upperLimit),stress(start:upperLimit),"linear
        ")
    Rsq = bestfit.Rsquared.ordinary
    if or(attempt > L/2,upperLimit > L-2)
```

```
        break
    end
    if or(isnan(Rsq),Rsq < 0.85)
        Rsq = 1
        start = start+1
        offset = offset+1
        if start > critStart
            offset = offset0; start = datum + offset;
            break
        end
    end
end

E = bestfit.Coefficients.Estimate(2)
E2 = (stress(upperLimit) - stress(start))/(ystrain(upperLimit)-ystrain(start))

% Yield Stress @ 0.2%
if overrideE == true
    Ecalc = Ein
elseif usealtE == true
    Ecalc = E2
else
    Ecalc = E
end
cond = false
y2p_strain = ystrain + 0.002;
y2p_stress = ystrain.*Ecalc;
yieldError = abs(y2p_stress-stress);
try
    [stressAtYield,yield]=polyxpoly(ystrain,stress,y2p_strain,y2p_stress);
catch
    warning('Problem using function.  Assigning a value of 0.')
    yield = 0;
end

p02 = false;
```

```
if isempty(yield)
    p02 = true;
    y02p_strain = ystrain + 0.0002
    y02p_stress = ystrain.*Ecalc
    yieldError = abs(y02p_stress-stress)
    [stressAtYield,yield]=polyxpoly(ystrain,stress,y02p_strain,y02p_stress)
end

p00002 = false;
if isempty(yield)
    p02 = false;
    p00002 = true;
    y00002p_strain = ystrain + 0.00002
    y00002p_stress = ystrain.*Ecalc
    yieldError = abs(y00002p_stress-stress)
    [stressAtYield,yield]=polyxpoly(ystrain,stress,y00002p_strain,
        y00002p_stress)
end

if isempty(yield)
    yield = stress(L-1);
end

while cond == false
    cond = true;
end

plot(y2p_strain,y2p_stress)
hold on
plot(ystrain,stress)
title("0.2% Offset Yield")
hold off

poisson = -xstrain./ystrain
propLim = stress(upperLimit)
nu = mean(poisson(datum+offset+1:upperLimit))
```

```
nupl = poisson(upperLimit:length(poisson))
nuplmean = mean(nupl)
plastfailstrain = max(ystrain) - ystrain(upperLimit)
instantStrainRate = ystrain./time
plasticStrainRate = mean(instantStrainRate(upperLimit:L))
names = ["E";"E_Alt";"prop_Limit";"sigma_y.2";"epsilon_f";"nu";"nu_pl";"SR_pl
    ";"t_0";"w_0"]
propsToFile = [E;E2;propLim;yield(1);plastfailstrain;nu;nuplmean;
    plasticStrainRate;t;w]

if p02 == true
    names(4) = "sigma_y_0.02_note-the-extra-0"
elseif p00002 == true
    names(4) = "sigma_y_0.00002_note-the-extra-0s"
end
% Write to file
if overrideE == true
    names(length(names)+1) = "E_Override"
    propsToFile(length(propsToFile)+1) = Ein
end
fileOut = sampleName + methodCall + "_Properties.csv";
writematrix([names,propsToFile],directoryOut+fileOut);
close all

if useMethod == "A"
    Method_A_props = propsToFile
elseif useMethod == "B"
    Method_B_props = propsToFile
elseif useMethod == "C"
    Method_C_props = propsToFile
elseif useMethod == "DTB"
    Method_DTB_props = propsToFile
elseif useMethod == "DC"
    Method_DC_props = propsToFile
elseif useMethod == "E"
    Method_E_props = propsToFile
```

```
elseif useMethod == "F"
    Method_F_props = propsToFile
end



%%%% MAT_187 Plastic Curves %%%%%%%%%%%%%%%%%%%%%%

% If young's modulud from the above cannot be trusted or to use E2 (see
% inputs)
if overrideE == true
    E = Ein
end
if usealtE == true
    E = E2
end
% verify what E is
E
plastystrain = ystrain - stress/E;
plaststress = stress;
for i = 1:L
    if plastystrain(L+1-i) <= 0
        plastystrain(L+1-i) = [];
        plaststress(L+1-i) = [];
    end
end
backward = true
while backward == true
    L2 = length(plastystrain);
    for i = 1:L2-1
        plastystrain(L2+1-i) - plastystrain(L2-i); % optional display for a
            visual check
        if plastystrain(L2+1-i) < plastystrain(L2-i)
            plastystrain(L2+1-i) = [];
            plaststress(L2+1-i) = [];
        end
```

```
    end
    if L2 - length(plastystrain) == 0
        backward = false
    end
end

% Write to file
fileOut = sampleName + methodCall + "_PlasticTensionCurve.csv";
writematrix([plastystrain,plaststress],directoryOut+fileOut);
close all
end

All Method Properties in One File
% Regular Curves
if usingAll == true
fileOut = "!" + sampleName + "_Properties_All-Together.csv";

writematrix([names,Method_A_props,Method_B_props,Method_C_props,
    Method_DC_props, ...
            Method_DTB_props, Method_E_props, Method_F_props],directoryOut+
                fileOut);
close all

fileOut = "!" + sampleName + "_Poisson_All-Together.csv";

writematrix([time,poissonA,poissonB,poissonC,poissonDC,poissonDTB,poissonE,
    poissonF],directoryOut+fileOut);
close all
end

if p02 == true
    plot(y02p_strain,y02p_stress)
else
    plot(y2p_strain,y2p_stress)
end
hold on
```

```
plot(ystrain,stress)
title("0.2% Offset Yield")
ylim([0,50])
hold off
```

B.2 CFD to Solid Model Stress Mapping

```
Get Stress

Scope
The scope of this script is to interpolate stresses at element centroids. "
    centroids.txt" should contain four columns. One of element IDs, one of
    costant X values, and two of Y and Z centroid values for working with a
    flat plate. There is no X since X is constant accross the flat plate. Once
     that, along with CFD's data is loaded in, the stresses are interpolated
    at the centroid of each element and are ready to copy-paste into a keyword
     file.
** IMPORTANT This was initally writted assuming the file to be saved in the
    same directory as the input and output files.
** IMPORTANT The surface size must fit on or inside of CFD's surface for the
    best results. For the first data set, this is a 51 x 51 elemnt grid of
    2,601 total elemnents covering 0 to 16 by 0 to 16 mm.
Steps:
Create a model similar to what CFD's results are from in LS-Dyna. Apply B.C.s,
     controls, materials and such, but no loads.
 From the .k file, copy paste the *ELEMENT_SOLID section of the keyword file
     into a text file and name it "elements.txt". Then, cope-paste the *NODE
     section into a file and name it "nodes.txt".
Now, ensure that CFD's results are saved with the same file names as shown
    below and that everything is in the same working folder. Check the "Inputs
    " section below, once verified, run the script.
After the script finishes running, copy-paste the desired results into the
    keyword file.

Inputs
Here are where the user inputs are located. These are things that may change
```

```
    at different times and/or stages.
clear all clc;

%% Input Data
dt = [redacted]
timeSteps = [redacted]; % This is the number of timesteps in the  data.
% It is equal to width(pressure) but that isn't yet loaded and this is used
% to initialize it. It shouldn't change unless we get new stress data.
surfx = [redacted]; % just below the x value of the top surface
%% LS-Dyna and Output Data
nint = 1; % number of integration points per element
t = 1; %time step being used for printing *INITIAL_STRESS_SOLID Card
INITIAL_STRESS_SOLID = false; % print for *INITIAL_STRESS_SOLID Card
USER_LOADING = false; % print for user defined load card
LOAD_SEGMENT = true; % print pressure for load segment
NODAL_FORCES = true; % print shear as concentrated nodal forced
COMBINE_LOAD_CURVES = false; % for multiple options that create load curves,
% this combines the load curves only into a single text file
COMBINE_ALL = true; % this combines all the partial keyword files into one
    file
% which can be copied directly into LS-DYNA. This is only for pressure and
% shear applied as equivelant nodal forces.

%% Set Units
% this depends on the input units and model units. For instance, if the
% coordinate text file is in m while Dyna is in mm, coordFactor = 10^3
coordFactor = 10^3;
stressFactor = 10^-6;

lcid = 100; % initialize load curve id. this will be one less than the minimum
% lcid output by the script. Ensure to use a large enough number as to
% avoid duplications.

Read in Data
Here, we turn CFD's text files into matrices. Coordinates, pressure, and shear
     stresses. Next, the element centroid input is turned into a matrix as
```

```
    well.
%% Coordinates
% These are currently modlled for CFD's flat plate data with [redacted]
    coordinates and [redacted] time steps.
coordinate_data = fopen('coordinate_data_for_MATLAB.txt','r');
formatCoord = '%f';
sizeCoord = [3, inf];
coord = transpose(fscanf(coordinate_data,formatCoord,sizeCoord));
format = ['%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f' ...
          '%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f' ...
          '%f %f %f ... [true length redacted, %f to end] ... %f %f %f'];
%% Pressure
pressure_data = fopen('pressure_data_for_MATLAB.txt','r');
pressure = cell2mat(textscan(pressure_data,format));

%% X-Y Shear
tau_xy_data = fopen('tau_xy_data_for_MATLAB.txt','r');
tau_xy = cell2mat(textscan(tau_xy_data,format));

%% X-Z Shear
tau_xz_data = fopen('tau_xz_data_for_MATLAB.txt','r');
tau_xz = cell2mat(textscan(tau_xz_data,format));

%% Corresponding Nodes
corresponding_nodes = fopen('elements.txt','r');
formatNode = '%f %f %f %f %f %f %f %f %f %f';
elements = cell2mat(textscan(corresponding_nodes,formatNode));
%elements(:,1) = []; % remove the part number

elements(:,10) = []; % remove the NAN

the_nodes = fopen('nodes.txt','r');
formatNode = '%f %f %f %f %f %f';
nodes = cell2mat(textscan(the_nodes,formatNode));
nodes(:,[5,6]) = [];
fclose all;
```

```
% Set units
coord = coord*coordFactor;
pressure = pressure*stressFactor;
tau_xy = tau_xy*stressFactor;
tau_xz = tau_xz*stressFactor;

Calcuations
Now, with everything matricized, we may perform some calulations. First, some
    variables are initialized outside of for loops and units are set. Next,
    the coordiantes closest to centroids are found in Y and Z directions, from
     here, these values are averaged and used to interpolate the stresses and
    pressure at the centroid of each element.
%% Stresses at Element Centroids

node_forcey = zeros(length(nodes),timeSteps+1); % force in y applied to each
    node
node_forcez = zeros(length(nodes),timeSteps+1); % force in y applied to each
    node
elArea = zeros(length(elements),2); % area of each element
nodalx = zeros(length(elements),width(elements));
nodaly = zeros(length(elements),width(elements)); % y coordinates of each node
nodalz = zeros(length(elements),width(elements)); % z coordinates of each node

node_forcey(:,1) = nodes(:,1);
node_forcez(:,1) = nodes(:,1);

%% intialize loop variables
centy = 0; % y coordinate of element centroid
centz = 0; % z " "

% a, b, c, d, A, B, C, and D are used to simplify expressions below
a = 0;
b = 0;
c = 0;
d = 0;
```

```
A = 0;
B = [0,0];
C = [0,0;0,0];
D = [0;0];

% Difference between coordinate and centroid point
diff = zeros(length(coord),1); % Total differnce. Used for interpolation
diffz = zeros(length(coord),1);% Z difference, used to determine alignment in
    z
diffy = zeros(length(coord),1);% Y difference, used to determine alignment in
    y
mindiff = [0;0;0;0]; % coordinate points closest to centroid in a square
    around it
index = [0,0,0,0]; % indexies of mindiff points
% y and z coordinate values used for interpolation
y1 = 0;
y2 = 0;
z1 = 0;
z2 = 0;

% Ensure that the interpolation points form a square
square = false; % criterion for if statement
squarey = 0; % criterion for y
squarez = 0; % criterion for z
samez = false; % true if the centroid z is the same as a coordinate z
samey = false; % true if centroid and coordinate y are the same

nodaly(1,:) = elements(1,:);
nodaly(1,:) = elements(1,:);
nodalz(1,:) = elements(1,:);
%topSurfNodes = nodes;

% Find which elements are on the top surface as they will take the loads
topSurfElements = elements;
for j = 1:length(elements)
    for i = 2:width(elements)
```

```
        % find coordinates of each node
        nodalx(j,i) = nodes(elements(j,i),2);
        nodaly(j,i) = nodes(elements(j,i),3);
        nodalz(j,i) = nodes(elements(j,i),4);
    end
end
J = length(elements);
for j = 1:length(elements)
    I = width(elements);
    for i = 2:width(elements)
        if nodes(elements(J,I),2) < surfx % only works for flat model
            % Set non-top curface elements to zero
            topSurfElements(J,I) = 0;
        end
        I = I-1;
    end
    if max(nodalx(J,:)) < surfx % only works for flat model
        % Eliminate elemnts below the surface
        topSurfElements(J,:) = [];
    end
    J = J-1;
    % calculate area assuming rectangular or square elements
    elArea(j,1) = elements(j,1);
    elArea(j,2) = abs((max(nodaly(j,(2:width(elements))))-min(nodaly(j,(2:
        width(elements))))))...
            *abs((max(nodalz(j,(2:width(elements))))-min(nodalz(j,(2:width(
                elements))))));
end

L = length(topSurfElements(:,1)); %readability and key strokes
centroids = zeros(L,4);
eid = topSurfElements(:,1);
epressure = zeros(L,timeSteps+1); % initialize pressure matrix at centroids
epressure(:,1) = eid; % firstcolumn is IDs
etau_xy = zeros(L,timeSteps+1); % initialize elemental tau xy
etau_xy(:,1) = eid;
```

```
etau_xz = zeros(L,timeSteps+1); % initialize elemental tau xz
etau_xz(:,1) = eid;
longTopSurfElements = topSurfElements;

% Reduce the top surface elements from 8 nodes to 4 nodes. The nodes below
% the top surface are eliminated. This is important for using the
% LOAD_SECTION keyword
[m,n]=size(topSurfElements);
for i=1:m
  tmp=nonzeros(topSurfElements(i,:));
  topSurfElements(i,:)=[tmp.', zeros(1,n-numel(tmp))];
end

topSurfElements(:,(6:9)) = []; % only works for 8 node bricks with element
    index
%topSurfElements(:,(5:8)) = []; % only works for 8 node bricks with element
    index
for j = 1:L % calculate element centroids
    centroids(j,1) = topSurfElements(j,1);
    centroids(j,2) = mean(nodalx(j,(2:width(elements))));
    centroids(j,3) = mean(nodaly(j,(2:width(elements))));
    centroids(j,4) = mean(nodalz(j,(2:width(elements))));
end
%allCentroids = zeros(length(elements),4);
for j = 1:length(elements) % calculate ALL element centroids
    centroids(j,1) = elements(j,1);
    centroids(j,2) = mean(nodalx(j,(2:width(elements))));
    centroids(j,3) = mean(nodaly(j,(2:width(elements))));
    centroids(j,4) = mean(nodalz(j,(2:width(elements))));
end
allCentroids = centroids;
for j = 1:L
    centy = centroids(j,3); % y coordinate of centroid of element j
    centz = centroids(j,4); % z coordinate of centroid of element j
    for k = 1:length(coord)
        % this loop compares the y and z coordinates of element j's
```

```
        % centroid again each of CFD's coordiantes to find the closest
        % these close points are then used for interpolation
        diffy(k) = (centy-coord(k,2));
        diffz(k) = (centz-coord(k,3));
        diff(k) = sqrt(diffy(k)^2+diffz(k)^2);

    end
    square = false; % initialize before while loop
    squarey = 0;
    squarez = 0;
    samez = false;
    samey = false;
    while square == false
        % catch for T ot Y shaped interpolations
        % this while loop ensures that there are two points in y greater
        % than the centroid and two points less than the centroid. It does
        % the same for z. When this criteria is met, the loop ends. The
        % loop will also end if samey or samez is triggered
        [mindiff, index] = mink(diff,4);
        if sign(diffz(index(1)))*sign(diffz(index(2)))*...
            sign(diffz(index(3)))*sign(diffz(index(4))) > 0
            squarez = 1;
        elseif sign(diffz(index(1)))*sign(diffz(index(2)))*...
                sign(diffz(index(3)))*sign(diffz(index(4))) == 0
            squarez = 1;
            samez = true;
        else
            diff(index(4)) = diff(index(4))+10000;
        end
        if sign(diffy(index(1)))*sign(diffy(index(2)))*...
                sign(diffy(index(3)))*sign(diffy(index(4))) > 0
            squarey = 1;
        elseif sign(diffy(index(1)))*sign(diffy(index(2)))*...
                sign(diffy(index(3)))*sign(diffy(index(4))) == 0
            squarey = 1;
            samey = true;
```

```
    else
        diff(index(4)) = diff(index(4))+10000;
    end
    if squarey*squarez == 1
        square = true;
    end
end
index = sort(index); % orders the indicies from smallest to largest
% instead of by difference

a = index(1); % readability/less key strokes below
b = index(2);
c = index(3);
d = index(4);
y1 = coord(a,2);
y2 = coord(c,2);
z1 = coord(a,3);
z2 = coord(b,3);
% First, we check if the FEM element centroid falls exactly in line
% with the Y and/or Z coordinate of the corrdinate data. In the case
% that it aligns with one direction, the interpolation is 1D. In the
% case that it aligns in both, there it no iterpolation as we use the
% exact point. These values are displayed so that the user may check
% them if desired.
if and(samez == true,samey == true)
    for i = 2:timeSteps+1
        epressure(j,i) = pressure(j,i-1);
        etau_xy(j,i) = tau_xy(j,i-1);
        etau_xz(j,i) = tau_xz(j,i-1);
    end
    disp("right on")
    disp(j)
    disp(coord(index,[2,3]))
    disp(centroids(j,[3,4]))
elseif and(samey == true,samez == false)
    for i = 2:timeSteps+1 % 1D interpolation over z
```

```
            epressure(j,i) = ((pressure(b)-pressure(a))/(z2-z1))*(centz-z1)+
                pressure(a);
            etau_xy(j,i) = ((tau_xy(b)-tau_xy(a))/(z2-z1))*(centz-z1)+tau_xy(a
                );
            etau_xz(j,i) = ((tau_xz(b)-tau_xz(a))/(z2-z1))*(centz-z1)+tau_xz(a
                );
        end
        disp("y_1 = y_2")
        disp(j)
    elseif and(samez == true,samey == false)
        for i = 2:timeSteps+1 % 1D interpolation over y
            epressure(j,i) = ((pressure(c)-pressure(a))/(y2-y1))*(centy-y1)+
                pressure(a);
            etau_xy(j,i) = ((tau_xy(c)-tau_xy(a))/(y2-y1))*(centy-y1)+tau_xy(a
                );
            etau_xz(j,i) = ((tau_xz(c)-tau_xz(a))/(y2-y1))*(centy-y1)+tau_xz(a
                );
        end
        disp("z_1 = z_2")
        disp(j)
    else % catches for weird results
        if y1 ~= coord(b,2)
            disp("probelm with y1")
            break
        end

        if y2 ~= coord(d,2)
            disp("probelm with y2")
            break
        end

        if z1 ~= coord(c,3)
            disp("probelm with z1")
            break
        end
```

```
        if z2 ~= coord(d,3)
            disp("probelm with z2")
            break
        end

        % Parameters for 2D interpolation
        A = ((y2-y1)*(z2-z1))^-1;
        B = [y2-centy,centy-y1];
        D = [z2-centz;centz-z1];

        for i = 2:timeSteps+1

            %% interpolate stresses using both y and z
            % Pressure
            C = [pressure(a,i-1), pressure(c,i-1)
                 pressure(b,i-1), pressure(d,i-1)];
            epressure(j,i) = A*B*C*D;

            % Shear xy
            C = [tau_xy(a,i-1), tau_xy(c,i-1)
                 tau_xy(b,i-1), tau_xy(d,i-1)];
            etau_xy(j,i) = A*B*C*D;

            % Shear xz
            C = [tau_xz(a,i-1), tau_xz(c,i-1)
                 tau_xz(b,i-1), tau_xz(d,i-1)];
            etau_xz(j,i) = A*B*C*D;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %% Shear as Forces
            yfactor = 1;
            zfactor = 1;
            for r = 2:width(topSurfElements(1,:))
                yfactor = 1;
                zfactor = 1;
%                 if nodes(topSurfElements(j,r),3) < ...
```

```
%                       allCentroids(allCentroids(:,1) ==(topSurfElements(j,1))
    ,3)
%                       yfactor = -1;
%                   end
%                   if nodes(topSurfElements(j,r),4) < ...
%                       allCentroids(allCentroids(:,1) ==(topSurfElements(j,1))
    ,4)
%                       zfactor = -1;
%                   end
% The above stuff would apply to y-z shear stresses, not x-y and x-z.
% If y-z stress were to be used instead,
% then uncomment the appropriate part of that that section.
                node_forcey(topSurfElements(j,r),i) =...
                    node_forcey(topSurfElements(j,r),i) +...
                    etau_xy(j,i)*elArea(elArea(:,1) ==(topSurfElements(j,1))
                        ,2)/4;
                node_forcez(topSurfElements(j,r),i) =...
                    node_forcez(topSurfElements(j,r),i) +...
                    etau_xz(j,i)*elArea(topSurfElements(j,1),2)/4;
            end
        end
    end
    if j == L % show that the loop was completed without problems causing
        breaks
        disp("loop unbroken")
    end
end

% Now, we want to remove zeros for the shear
yLoadedNodes = node_forcey;
zLoadedNodes = node_forcez;
J = length(node_forcey);
for j = 1:length(node_forcey)
    if nodes(J,2) < surfx
        yLoadedNodes(J,:) = [];
        zLoadedNodes(J,:) = [];
```

```
    end
    J = J-1; % go down from top index since we are eliminating rows
end

Output
%% Print Stresses as *INITIAL_STRESS_SOLID
if INITIAL_STRESS_SOLID == true
    initial_stress = zeros(2*L,8); % Assuming no extra History Variables

    for i = 1:L
    initial_stress(i*2-1,1) = eid(i); % element ID
    initial_stress(i*2-1,2) = nint; % number of integration points
    initial_stress(i*2,1) = epressure(i,t+1); % sigxx
    initial_stress(i*2,4) = etau_xy(i,t+1); %sigxy
    initial_stress(i*2,6) = etau_xz(i,t+1); %sigzx
    end
    firstLine = "$#     eid      nint     nhisv     large     iveflg     ialegp
          nthint    nthhsv\n";
    secondLine = initial_stress(1,:);
    thirdLine = "$#   sigxx     sigyy     sigzz     sigxy     sigyz     sigzx
             eps\n";
    theRest = initial_stress;
    theRest(1,:) = [];
    output = fopen("initialStress.txt",'w');
    printFormat1 = "%.4E%.4E%.4E%.4E%.4E%.4E%.4E\n";
    if max(ceil(log10(abs(secondLine(1))+1)),1) == 1 % formatting for various
        element numbers
    printFormat2 =  "         %.0f         %.0f         %.0f         %.0f
                %.0f         %.0f         %.0f         %.0f\n";
    elseif max(ceil(log10(abs(secondLine(1))+1)),1) == 2
    printFormat2 =  "         %.0f         %.0f         %.0f         %.0f
                %.0f         %.0f         %.0f         %.0f\n";
    elseif max(ceil(log10(abs(secondLine(1))+1)),1) == 3
    printFormat2 =  "        %.0f         %.0f         %.0f         %.0f
                %.0f         %.0f         %.0f         %.0f\n";
    elseif max(ceil(log10(abs(secondLine(1))+1)),1) == 4
```

```
printFormat2 = "      %.0f          %.0f          %.0f          %.0f
            %.0f          %.0f          %.0f          %.0f\n";
elseif max(ceil(log10(abs(secondLine(1))+1)),1) == 5
printFormat2 = "      %.0f          %.0f          %.0f          %.0f
    %.0f          %.0f          %.0f          %.0f\n";
elseif max(ceil(log10(abs(secondLine(1))+1)),1) == 6
printFormat2 = "    %.0f          %.0f          %.0f          %.0f
    %.0f          %.0f          %.0f          %.0f\n";
elseif max(ceil(log10(abs(secondLine(1))+1)),1) == 7
printFormat2 = "   %.0f          %.0f          %.0f          %.0f
    %.0f          %.0f          %.0f          %.0f\n";
elseif max(ceil(log10(abs(secondLine(1))+1)),1) == 8
printFormat2 = " %.0f          %.0f          %.0f          %.0f          %.0
    f          %.0f          %.0f          %.0f\n";
elseif max(ceil(log10(abs(secondLine(1))+1)),1) == 9
printFormat2 = " %.0f          %.0f          %.0f          %.0f          %.0f
            %.0f          %.0f          %.0f\n";
elseif max(ceil(log10(abs(secondLine(1))+1)),1) == 10
printFormat2 = "%.0f          %.0f          %.0f          %.0f          %.0f
            %.0f          %.0f          %.0f\n";
else
disp("Wow, that is a LOT of elements") % Unrealistic to get here
end

fprintf(output,firstLine);
fprintf(output,printFormat2,secondLine);
fprintf(output,thirdLine);

nextLine = [0,0,0,0,0,0,0,0];
form = ["%.4E","%.4E","%.4E","%.4E","%.4E","%.4E","%.4E","\n"];
for i = 1:length(theRest)
if i == length(theRest)
    form = ["%.4E","%.4E","%.4E","%.4E","%.4E","%.4E","%.4E"];
else
    form = ["%.4E","%.4E","%.4E","%.4E","%.4E","%.4E","%.4E","\n"];
end
```

```
for j = 1:width(theRest)
    if theRest(i,j) < 0  % account for negative values in formatting
        form(j) = "%.3E";
    end
end
nextLine = theRest(i,:); % Next line to be printed to the file
if mod(i,2) == 0
    if max(ceil(log10(abs(theRest(i,1))+1)),1) == 1 % same formatting as
        mentioned above
        printFormat2 =  "          %.0f          %.0f          %.0f
            %.0f          %.0f          %.0f          %.0f          %.0f\n";
    elseif max(ceil(log10(abs(theRest(i,1))+1)),1) == 2
        printFormat2 =  "          %.0f          %.0f          %.0f
            %.0f          %.0f          %.0f          %.0f          %.0f\n";
    elseif max(ceil(log10(abs(theRest(i,1))+1)),1) == 3
        printFormat2 =  "          %.0f          %.0f          %.0f          %.0
            f          %.0f          %.0f          %.0f          %.0f\n";
    elseif max(ceil(log10(abs(theRest(i,1))+1)),1) == 4
        printFormat2 =  "          %.0f          %.0f          %.0f          %.0f
                    %.0f          %.0f          %.0f          %.0f\n";
    elseif max(ceil(log10(abs(theRest(i,1))+1)),1) == 5
        printFormat2 =  "          %.0f          %.0f          %.0f          %.0f
                    %.0f          %.0f          %.0f          %.0f\n";
    elseif max(ceil(log10(abs(theRest(i,1))+1)),1) == 6
        printFormat2 =  "          %.0f          %.0f          %.0f          %.0f
                    %.0f          %.0f          %.0f          %.0f\n";
    elseif max(ceil(log10(abs(theRest(i,1))+1)),1) == 7
        printFormat2 =  "          %.0f          %.0f          %.0f          %.0f
                    %.0f          %.0f          %.0f          %.0f\n";
    elseif max(ceil(log10(abs(theRest(i,1))+1)),1) == 8
        printFormat2 =  "          %.0f          %.0f          %.0f          %.0f
                    %.0f          %.0f          %.0f          %.0f\n";
    elseif max(ceil(log10(abs(theRest(i,1))+1)),1) == 9
        printFormat2 =  "          %.0f          %.0f          %.0f          %.0f
                    %.0f          %.0f          %.0f          %.0f\n";
```

```
        elseif max(ceil(log10(abs(theRest(i,1))+1)),1) == 10
            printFormat2 =  "%.0f          %.0f          %.0f          %.0f
                        %.0f          %.0f          %.0f          %.0f\n";
        else
            disp("Wow, that is a LOT of elements")
        end
        printFormat = printFormat2;
    else
        printFormat = form(1);
        for k = 2:length(form)
            printFormat = printFormat + form(k);
        end
        nextLine(8) = [];
    end
    fprintf(output,printFormat,nextLine);
    end
    close all
end
%% Print to User Defined Load
if USER_LOADING == true
    disp("User Load")
    % gonna need to define curves for all the bois. This will be a fat
    % output file with some sexy loops in the script
end

%% Print to LOAD_SEGMENT
if LOAD_SEGMENT == true
    % first, make load curves for each element
    % second, define square segments
    output = fopen("load_segment.txt",'w');
    segmentFile = fopen("segment_id.txt",'w');
    numb = 0; % number of charachters in lcid
    Numb = 0; % number of charachters in segment id
    form = ["%.14E","%.14E"]; %initialize form for this part
    segmentid = 0;
    NUMB = [0,0,0,0]; % number of charachters in each node's id
```

```
segmentFormat = ["%.0f","%.0f","%.0f","%.0f"];
lcidFormat = "%.0f";
segmentidFormat = ".0f";
fifthLineFormat = "%.0f";
for j = 1:L
    lcid = lcid+1;
    numb = max(ceil(log10(abs(lcid)+1)),1);
    firstLine = "*DEFINE_CURVE_TITLE\n";
    secondLine = "Element_" + int2str(epressure(j,1)) + "\n";
    thirdLine = "$#     lcid      sidr       sfa        sfo       offa
        offo     dattyp     lcint\n";
    fourthLine = [lcid, 0, 1, 1, 0, 0, 0, 0];
    fifthLine = "$#                  a1                  o1\n\n";
    if numb == 1 % same formatting as mentioned above
        lcidFormat = "         %.0f";
        printFormat2 = "          %.0f          %.0f        %.1f        %.1f
                  %.1f        %.1f          %.0f          %.0f\n";
    elseif numb == 2
        lcidFormat = "         %.0f";
        printFormat2 = "          %.0f          %.0f        %.1f        %.1f
                  %.1f        %.1f          %.0f          %.0f\n";
    elseif numb == 3
        lcidFormat = "         %.0f";
        printFormat2 = "          %.0f          %.0f        %.1f        %.1f
                  %.1f        %.1f          %.0f          %.0f\n";
    elseif numb == 4
        lcidFormat = "       %.0f";
        printFormat2 = "        %.0f          %.0f        %.1f        %.1f
                  %.1f        %.1f          %.0f          %.0f\n";
    elseif numb == 5
        lcidFormat = "      %.0f";
        printFormat2 = "       %.0f          %.0f        %.1f        %.1f
                  %.1f        %.1f          %.0f          %.0f\n";
    elseif numb == 6
        lcidFormat = "     %.0f";
        printFormat2 = "      %.0f          %.0f        %.1f        %.1f
```

```
                %.1f        %.1f          %.0f          %.0f\n";
elseif numb == 7
    lcidFormat = "   %.0f";
    printFormat2 =  "   %.0f          %.0f        %.1f        %.1f
        %.1f        %.1f          %.0f          %.0f\n";
elseif numb == 8
    lcidFormat = "  %.0f";
    printFormat2 =  "  %.0f          %.0f        %.1f        %.1f
        %.1f        %.1f          %.0f          %.0f\n";
elseif numb == 9
    lcidFormat = " %.0f";
    printFormat2 =  " %.0f          %.0f        %.1f        %.1f
        %.1f        %.1f          %.0f          %.0f\n";
elseif numb == 10
    lcidFormat = "%.0f";
    printFormat2 =  "%.0f          %.0f        %.1f        %.1f
        %.1f        %.1f          %.0f          %.0f\n";
else
    disp("Wow, that is a LOT of elements")
end
fprintf(output,firstLine);
fprintf(output,secondLine);
fprintf(output,thirdLine);
fprintf(output,printFormat2,fourthLine);
fprintf(output,fifthLine);
for i = 2:timeSteps+1
    form = ["%.14E","%.14E"];
    if epressure(j,i) < 0
        form(2) = "%.13E";
    end
    printFormat = form(1) + form(2) + "\n";
    nextLine = [(i-1)*dt, epressure(j,i)];
    fprintf(output,printFormat,nextLine);
end
% now, define and print the segemnt ids
% node segments need to move ccw
```

```
% Because of the way dyna defines elements, we will define the
% seemnts in one of two ways to ensure ccw rotation
segment = [0;0;0;0];
if longTopSurfElements(j,2) == 0
    segment = [topSurfElements(j,2);topSurfElements(j,3);
               topSurfElements(j,5);topSurfElements(j,4)];
elseif longTopSurfElements(j,3) == 0
    segment = [topSurfElements(j,2);topSurfElements(j,4);
               topSurfElements(j,5);topSurfElements(j,3)];
else
    disp("segment problem")
    break
end
% Next, we print the segments
segmentid = segmentid +1;
Numb = max(ceil(log10(abs(segmentid)+1)),1);
if Numb == numb
   segmentidFormat = lcidFormat;
elseif Numb == 1
    segmentidFormat = "         %.0f";
elseif Numb == 2
    segmentidFormat = "        %.0f";
elseif Numb == 3
    segmentidFormat = "       %.0f";
elseif Numb == 4
    segmentidFormat = "      %.0f";
elseif Numb == 5
    segmentidFormat = "     %.0f";
elseif Numb == 6
    segmentidFormat = "    %.0f";
elseif Numb == 7
    segmentidFormat = "   %.0f";
elseif Numb == 8
    segmentidFormat = "  %.0f";
elseif Numb == 9
    segmentidFormat = " %.0f";
```

```
elseif Numb == 10
    segmentidFormat = "%.0f";
else
    disp("That is a great amny segments!")
end
for i = 1:length(NUMB)
    NUMB(i) = max(ceil(log10(abs(segment(i))+1)),1);
    if NUMB(i) == 0
        segmentFormat(i) = "          %.0f";
    elseif NUMB(i) == 1
        segmentFormat(i) = "         %.0f";
    elseif NUMB(i) == 2
        segmentFormat(i) = "        %.0f";
    elseif NUMB(i) == 3
        segmentFormat(i) = "       %.0f";
    elseif NUMB(i) == 4
        segmentFormat(i) = "      %.0f";
    elseif NUMB(i) == 5
        segmentFormat(i) = "     %.0f";
    elseif NUMB(i) == 6
        segmentFormat(i) = "    %.0f";
    elseif NUMB(i) == 7
        segmentFormat(i) = "   %.0f";
    elseif NUMB(i) == 8
        segmentFormat(i) = "  %.0f";
    elseif NUMB(i) == 9
        segmentFormat(i) = " %.0f";
    elseif NUMB(i) == 10
        segmentFormat(i) = "%.0f";
    else
        disp("That is a whole damn bunch of nodes!")
    end
end

firstLine1 = "*LOAD_SEGMENT_ID\n";
secondLine1 = "$#       id\n";
```

```
        thirdLine1 = segmentid;
        thirdLinepart2 = " Element_" + int2str(epressure(j,1)) + "\n";
        fourthLine1 = "$#     lcid          sf          at          n1          n2
                  n3          n4          n5\n";
        fifthLine1 = [lcid, 1, 0, segment(1), segment(2), segment(3), segment
            (4), 0];
        fifthLineFormat = lcidFormat + "        %.1f        %.1f" + ...
            segmentFormat(1) + segmentFormat(2) + segmentFormat(3) +
                segmentFormat(4) + "          %.0f\n";

        fprintf(segmentFile,firstLine1);
        fprintf(segmentFile,secondLine1);
        fprintf(segmentFile,segmentidFormat,thirdLine1);
        fprintf(segmentFile,thirdLinepart2);
        fprintf(segmentFile,fourthLine1);
        fprintf(segmentFile,fifthLineFormat,fifthLine1);

    end
    close all
end

%% Print to concentrated nodal forces for shear
if NODAL_FORCES == true
    nodeChar = 0; % characters in node id
    nodeForm = "%.0f";
    % first, we define load curves and then we load the nodes
    lcidFormat = "%.0f"; % reinitialize incase the other loop is turned off
    output = fopen("shear_load_curves.txt",'w');
    nodeLoad = fopen("node_loads.txt",'w');
    nodeLoadFormat = "%.0f";
    %here, we apply load node formatting
    firstLineNL = "*LOAD_NODE_POINT\n";
    secondLineNL = "$#      nid        dof       lcid         sf        cid
        m1         m2         m3\n";
    fprintf(nodeLoad,firstLineNL);
    fprintf(nodeLoad,secondLineNL);
```

```
for j = 1:length(yLoadedNodes) % length should be the same as z loaded
    nodes
    nodeChar = max(ceil(log10(abs(yLoadedNodes(j,1))+1)),1);
    if nodeChar == 0
        nodeForm = "          %.0f";
    elseif nodeChar == 1
        nodeForm = "         %.0f";
    elseif nodeChar == 2
        nodeForm = "        %.0f";
    elseif nodeChar == 3
        nodeForm = "       %.0f";
    elseif nodeChar == 4
        nodeForm = "      %.0f";
    elseif nodeChar == 5
        nodeForm = "     %.0f";
    elseif nodeChar == 6
        nodeForm = "    %.0f";
    elseif nodeChar == 7
        nodeForm = "   %.0f";
    elseif nodeChar == 8
        nodeForm = "  %.0f";
    elseif nodeChar == 9
        nodeForm = " %.0f";
    elseif nodeChar == 10
        nodeForm = "%.0f";
    else
        disp("That is a whole damn bunch of nodes!")
    end

    lcid = lcid+1;
    numb = max(ceil(log10(abs(lcid)+1)),1);
    firstLine = "*DEFINE_CURVE_TITLE\n";
    secondLine = "y Component of Node_" + int2str(yLoadedNodes(j,1)) + "\n
        ";
    thirdLine = "$#     lcid      sidr       sfa        sfo       offa
        offo     dattyp     lcint\n";
```

```
fourthLine = [lcid, 0, 1, 1, 0, 0, 0, 0];
fifthLine = "$#                    a1                    o1\n\n";
if numb == 1 % same formatting as mentioned above
    lcidFormat = "          %.0f";
    printFormat2 =  "          %.0f          %.0f          %.1f          %.1f
              %.1f          %.1f          %.0f          %.0f\n";
elseif numb == 2
    lcidFormat = "          %.0f";
    printFormat2 =  "          %.0f          %.0f          %.1f          %.1f
              %.1f          %.1f          %.0f          %.0f\n";
elseif numb == 3
    lcidFormat = "          %.0f";
    printFormat2 =  "          %.0f          %.0f          %.1f          %.1f
              %.1f          %.1f          %.0f          %.0f\n";
elseif numb == 4
    lcidFormat = "          %.0f";
    printFormat2 =  "          %.0f          %.0f          %.1f          %.1f
              %.1f          %.1f          %.0f          %.0f\n";
elseif numb == 5
    lcidFormat = "          %.0f";
    printFormat2 =  "          %.0f          %.0f          %.1f          %.1f
              %.1f          %.1f          %.0f          %.0f\n";
elseif numb == 6
    lcidFormat = "          %.0f";
    printFormat2 =  "          %.0f          %.0f          %.1f          %.1f
              %.1f          %.1f          %.0f          %.0f\n";
elseif numb == 7
    lcidFormat = "          %.0f";
    printFormat2 =  "          %.0f          %.0f          %.1f          %.1f
          %.1f          %.1f          %.0f          %.0f\n";
elseif numb == 8
    lcidFormat = "          %.0f";
    printFormat2 =  "          %.0f          %.0f          %.1f          %.1f
          %.1f          %.1f          %.0f          %.0f\n";
elseif numb == 9
    lcidFormat = "          %.0f";
```

```
        printFormat2 =  " %.0f          %.0f        %.1f        %.1f
            %.1f        %.1f          %.0f          %.0f\n";
    elseif numb == 10
        lcidFormat = "%.0f";
        printFormat2 =  "%.0f          %.0f        %.1f        %.1f
            %.1f        %.1f          %.0f          %.0f\n";
    else
        disp("Wow, that is a LOT of load curves")
    end
    fprintf(output,firstLine);
    fprintf(output,secondLine);
    fprintf(output,thirdLine);
    fprintf(output,printFormat2,fourthLine);
    fprintf(output,fifthLine);
    for i = 2:timeSteps+1
        form = ["%.14E","%.14E"];
        if yLoadedNodes(j,i) < 0
            form(2) = "%.13E";
        end
        printFormat = form(1) + form(2) + "\n";
        nextLine = [(i-1)*dt, yLoadedNodes(j,i)];
        fprintf(output,printFormat,nextLine);
    end
    % now we apply y load curves to nodes. The format is already there
    nextLineNode = [yLoadedNodes(j,1), 2, lcid, 1, 0, 0, 0, 0];
    nodeLoadFormat = nodeForm + "          %.0f" + lcidFormat + ...
        "        %.1f          %.0f          %.0f          %.0f          %.0f\n
            ";
    fprintf(nodeLoad,nodeLoadFormat,nextLineNode);

end
for j = 1:length(zLoadedNodes) % length should be the same as y loaded
    nodes
    nodeChar = max(ceil(log10(abs(yLoadedNodes(j,1))+1)),1);
    if nodeChar == 0
        nodeForm = "          %.0f";
```

```
elseif nodeChar == 1
    nodeForm = "          %.0f";
elseif nodeChar == 2
    nodeForm = "         %.0f";
elseif nodeChar == 3
    nodeForm = "        %.0f";
elseif nodeChar == 4
    nodeForm = "       %.0f";
elseif nodeChar == 5
    nodeForm = "      %.0f";
elseif nodeChar == 6
    nodeForm = "     %.0f";
elseif nodeChar == 7
    nodeForm = "    %.0f";
elseif nodeChar == 8
    nodeForm = "  %.0f";
elseif nodeChar == 9
    nodeForm = " %.0f";
elseif nodeChar == 10
    nodeForm = "%.0f";
else
    disp("That is a whole damn bunch of nodes!")
end

lcid = lcid+1;
numb = max(ceil(log10(abs(lcid)+1)),1);
firstLine = "*DEFINE_CURVE_TITLE\n";
secondLine = "z Component of Node_" + int2str(yLoadedNodes(j,1)) + "\n
    ";
thirdLine = "$#     lcid      sidr       sfa        sfo       offa
    offo     dattyp     lcint\n";
fourthLine = [lcid, 0, 1, 1, 0, 0, 0, 0];
fifthLine = "$#                  a1                  o1\n\n";
if numb == 1 % same formatting as mentioned above
    lcidFormat = "         %.0f";
    printFormat2 =  "         %.0f         %.0f       %.1f       %.1f
```

```
                %.1f        %.1f          %.0f          %.0f\n";
elseif numb == 2
    lcidFormat = "        %.0f";
    printFormat2 =  "        %.0f          %.0f        %.1f        %.1f
                %.1f        %.1f          %.0f          %.0f\n";
elseif numb == 3
    lcidFormat = "        %.0f";
    printFormat2 =  "        %.0f          %.0f        %.1f        %.1f
                %.1f        %.1f          %.0f          %.0f\n";
elseif numb == 4
    lcidFormat = "       %.0f";
    printFormat2 =  "       %.0f          %.0f        %.1f        %.1f
                %.1f        %.1f          %.0f          %.0f\n";
elseif numb == 5
    lcidFormat = "      %.0f";
    printFormat2 =  "      %.0f          %.0f        %.1f        %.1f
                %.1f        %.1f          %.0f          %.0f\n";
elseif numb == 6
    lcidFormat = "     %.0f";
    printFormat2 =  "     %.0f          %.0f        %.1f        %.1f
                %.1f        %.1f          %.0f          %.0f\n";
elseif numb == 7
    lcidFormat = "    %.0f";
    printFormat2 =  "    %.0f          %.0f        %.1f        %.1f
        %.1f        %.1f          %.0f          %.0f\n";
elseif numb == 8
    lcidFormat = "  %.0f";
    printFormat2 =  "  %.0f          %.0f        %.1f        %.1f
        %.1f        %.1f          %.0f          %.0f\n";
elseif numb == 9
    lcidFormat = " %.0f";
    printFormat2 =  " %.0f          %.0f        %.1f        %.1f
        %.1f        %.1f          %.0f          %.0f\n";
elseif numb == 10
    lcidFormat = "%.0f";
    printFormat2 =  "%.0f          %.0f        %.1f        %.1f
```

```
                %.1f        %.1f          %.0f          %.0f\n";
        else
            disp("Wow, that is a LOT of load curves")
        end
        fprintf(output,firstLine);
        fprintf(output,secondLine);
        fprintf(output,thirdLine);
        fprintf(output,printFormat2,fourthLine);
        fprintf(output,fifthLine);
        for i = 2:timeSteps+1
            form = ["%.14E","%.14E"];
            if zLoadedNodes(j,i) < 0
                form(2) = "%.13E";
            end
            printFormat = form(1) + form(2) + "\n";
            nextLine = [(i-1)*dt, zLoadedNodes(j,i)];
            fprintf(output,printFormat,nextLine);
        end
        % Now we apply z load curves to nodes
        nextLineNode = [zLoadedNodes(j,1), 3, lcid, 1, 0, 0, 0, 0];
        nodeLoadFormat = nodeForm + "          %.0f" + lcidFormat + ...
            "       %.1f          %.0f          %.0f          %.0f          %.0f\n
                ";
        fprintf(nodeLoad,nodeLoadFormat,nextLineNode);
    end
    close all;
end

%% Combines only the load curves
if COMBINE_LOAD_CURVES == true
    loadCurves = fopen("Load_Curves_for_Pressure_and_Shear.txt",'w');
    file1 = fileread("load_segment.txt");
    file2 = fileread("shear_load_curves.txt");
    fprintf(loadCurves,'%s%s',file1,file2);
    fclose all;
end
```

```
%% Combines the files to copy-paste into ls dyna keyword
if COMBINE_ALL == true
    copyToKey = fopen("copy_to_keyword.txt",'w');
    file1 = fileread("load_segment.txt");
    file2 = fileread("shear_load_curves.txt");
    file3 = fileread("segment_id.txt");
    file4 = fileread("node_loads.txt");
    fprintf(copyToKey,'%s%s%s%s',file1,file2,file3,file4);
    fclose all;
end
close all;
fclose all;
```