MICROMECHANICAL ANALYSIS OF ENERGY EXPENDITURE IN SNOW FRACTURE

by

Anthony Michael LeBaron

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

of

Doctor of Philosophy

in

Engineering

MONTANA STATE UNIVERSITY
Bozeman, Montana

July 2017

©COPYRIGHT

by

Anthony Michael LeBaron

2017

All Rights Reserved

ACKNOWLEDGEMENTS

The Benjamin Fellowship funded my first year at MSU. Johnson and Rita Liao, Kenneth Tait, and the parents of Blake Morstad have all aided me over the years through their generous contributions to scholarships at MSU.

TABLE OF CONTENTS

1. BACKGROUND AND OBJECTIVES 1

Fracture Mechanics 1
Fracture Modes 3
Irwin's Modification 3
Diffuse Failure 5
Dry Slab Avalanches 6
Snow Metamorphism 7
Temperature Gradient Metamorphism 8
Radiation Recrystallization 8
Surface Hoar 9
Equilibrium Growth 9
Fracture in Snow 10
The Role of Strain Softening and Damage Accumulation 11
Current Macroscopic Fracture Models 12
Microstructural Fracture Mechanics 16
Fracture in Snow: Laboratory Experimental Approaches 18
Graph Theory 19
Research Objectives 22

2. EXPERIMENTS 24

Procedure 24
Laboratory Snow Breeder 24
Procedure for Homogeneous Samples 26
Samples With Weak Layers 26
Collection of CT Samples 28
Analytical Method 31
Relevant Sample Geometry 31
Defining Failure 32
Work Done on Sample 34
Energy Release Rate 34
Experimental Results 36

3. IMAGE PROCESSING 38

Overview 38
Snow Microstructure Capture 38
Snow Grain Segmentation 41
Distance Transform 42

TABLE OF CONTENTS – CONTINUED

Watershed Transform 43
H-Minima Transform 45
Bond Size and Orientation 46
Approximation of Grains Using Multisphere Method 47
Fine-tuning Volume of Approximation 48
Description of Code 49

4. MINIMUM FRACTURE ENERGY 53

Theory 53
Application of Graph Theory to Snow Microstructure 54
Algorithm 57
Graph Creation 57
Minimum Energy Calculation 59

5. DISCRETE ELEMENT MODEL OVERVIEW 61

Purpose and Objectives 61
General DEM Background 62
Geometric Variables 63
Contact Constitutive Laws 66
Timestep 68
Failure of Contacts 68
Frictional Contacts 69
DEM Formulation for Bonded Particles 69
Normal Stiffness 69
Shear Stiffness 73
Bond Strength Criteria 74
Normal Strength 74
Shear Strength 77
Modified Criteria for Snow 78
Bending and Twisting Stiffness 78
Approximating Grains as Clumps 79
Inertial Properties of Clumps 79
Forces and Torques acting on clumps 80
Contact Placement 80
Frictional Contact 81
Damping 82
Boundary Conditions 83
Execution in YADE 83

TABLE OF CONTENTS – CONTINUED

6. DEM PARAMETRIC ANALYSIS 86

Sensitivity to H-Min Parameter 86
Effect on Stress-Strain Relationship 88
Effect on Energy Dissipation 88
Computational Considerations 90
Sensitivity to Number of spheres per clump 90
Effect of Ice Network Constitutive Properties 92
Elastic Modulus 93
Density 95
Surface Formation Energy 98
Summary of Parameters Chosen 98
DEM Validation Using Simple Geometry 99
Prediction of Energy Release 100

7. DEM RESULTS 103

Spatial and Temporal Analysis of Bond Breakage Events 103
Direction of Propagation 104
Size of FPZ 106
Representative Volume Element 110
Energy Dissipation 112
Comparison to Experiments 112
Homogeneous Samples 115
Samples with Buried SH 116
Estimates of Energy Release Rate 121
Tensile Tests 121
Homogeneous Sample 123
Sample With Buried SH 123
Discussion 123

8. ANALYSIS 125

Anticrack-Based Modeling from Field Experiments 125
Shear Model from Field Measurements 126
Size Effect Law 128
Layered Samples 128
Size of FPZ and Uncertainty 131
Cohesive Crack Faces vs Large FPZ 132
Consideration of Minimum Energy Path 132

TABLE OF CONTENTS – CONTINUED

Stress Concentrations: Effect on FPZ 134

9. CONCLUSIONS 140

Direction of Energy/Loading 141
Considerations for future models 143

REFERENCES CITED 145

APPENDICES 153

APPENDIX A: Image Processing MATLAB Code 154

APPENDIX B: Minimum Energy Code 185

APPENDIX C: Sample YADE Code 192

APPENDIX D: All Results 207

LIST OF TABLES

Table Page

1.1 Summary of critical energy release rate G_c calculated from field studies 16

2.1 Summary of all experimental Results 37

6.1 Values of E^*_{ice} from six different samples 95

6.2 Model parameters used for density sensitivity analysis 96

6.3 Model parameters used for density sensitivity analysis 98

LIST OF FIGURES

Figure Page

1.1 A plate of material with a crack of half-length a. The crack effectively weakens the material 2

1.2 Three different fracture modes. Image: Twisp/Wikimedia Commons 4

1.3 Slab and weak layer stratigraphy of avalanches 7

1.4 Striated, cup-shaped crystals are typical of depth hoar. The angular edges form poor bonds with neighboring crystals. Photo: K. Birkeland/USFS. 9

1.5 Condensation occurs on objects that are colder than the ambient air. Vapor may precipitate in liquid form on the outside of the glass (left) or solid form (right). Photo: K Birkeland/USFS 10

1.6 Water vapor particles concentrate around sharp, convex corners and will flow to concave areas. 11

1.7 Euler's seven bridges problem that spurred the creation of graph theory 20

1.8 The seven bridges of Konigsberg represented as a graph, showing how bridges connect the four landmasses. 20

2.1 The laboratory snow breeder is a convenient way to create nature-similar snow crystals year round. 25

2.2 A load cell measures force as the load actuator advances at a constant velocity 27

2.3 The laboratory surface hoar maker blows moisture-laden air gently across the snow surface. The snow surface emits longwave radiation to the cold ceiling. 29

2.4 A reservoir of liquid water is kept at a constant temperature. Fans (indide ducts, not seen in photo) move warm, humid air out the ducts. The lid is closed during operation. 30

2.5 Illustration of generalized progression of experimental force-displacement curves. 33

LIST OF FIGURES – CONTINUED

Figure		Page
2.6	In experiment 20160809, there are three large drops in force. It is not clear which represents global failure.	34
2.7	Work done on a sample of snow is simply the area under the force-displacement curve.	35
2.8	The two extreme values of energy release.	36
3.1	Overview of how CT scan data is discretized into an assembly of bonds and grains	39
3.2	Cylindrical snow sample placed in the CT scanner, ready to be scanned. X-ray source is left, sensor on right.	40
3.3	Reconstructed images lie in the horizontal plane, outlined by the white square.	41
3.4	A reconstructed grayscale "slice" from the CT scan (left) is binarized (right)	42
3.5	Two bonded grains (left) have the distance transform applied (right) to measure the distance in pixels from a pixel to the edge of the white area	43
3.6	The negative of the distance transform (left) suggests two "basins" separated by a "ridge" in gray. In the original image, this "ridge" represents a bond separating two grains	44
3.7	The initial ice network microstructure (top left) is segmented using various H-minimum thresholds. Different colors represent separate grains	50
3.8	Illustration of multisphere method	51
3.9	Key variables and processes used to process binary CT images into executable DEM simulation *yadeout.py*.	52
4.1	A cross-section of a hypothetical granular microstructure (left) is converted into a node/line graph representing potential fracture paths (right).	55

LIST OF FIGURES – CONTINUED

Figure Page

4.2 Max-flow/min-cut theorem applied to the sample geometry from figure 4.1. Lines with infinite capacity were added to the top and bottom of the sample in order to create top and bottom master nodes that must be separated by the cut. 56

4.3 Example *bonds* and *labelgrn* arrays for simple geometry. While the example is in 2D, the concepts apply to 3D arrays as well. 57

4.4 The end result of the minimum energy cut. The top of the sample (gold) is separated from the bottom (blue) via the path requiring the least surface formation energy. 60

5.1 Six possible degrees of freedom for deformations between spheres. From [87] 64

5.2 Two undeformed spherical DEM elements, labeled with relevant geometric variables 64

5.3 Two spherical DEM elements deform in the normal direction, introducing deformed radii d_1 and d_2 65

5.4 Two spheres in the previous timestep (left) and current timestep (right). After [87] 66

5.5 Local coordinate system and radii of two spheres. 70

5.6 Close-up at the contact between two deformed spheres. Solid lines are sphere surfaces. Dashed lines indicate undeformed surface that would occur if there were no resistance to penetration. Black line represents contact. 71

5.7 Cross section of a circular bond during failure under normal loading. As the crack propagates, bond radius changes by *da* 75

5.8 Cross section of a circular bond during failure initiation under shear loading. 77

5.9 Grain-grain contact is placed between the two closest spheres in neighboring grains 81

LIST OF FIGURES – CONTINUED

Figure Page

6.1 Multisphere assemblies for three different values of H_{min}. A lower H_{min} corresponds to more grains and bonds. 87

6.2 Number of bonds and grains generated with different H_{min} values in the watershed segmentation. Note the logarithmic scale on the y-axis. 87

6.3 Comparison of stress-strain curves for the same sample after applying four different watershed depth criteria 89

6.4 Comparison of dissipation curves for the same sample after applying four different H_{min} values 89

6.5 Comparison of computational speed of a sample after applying seven different watershed depths. 91

6.6 Comparison of bulk sample properties with variation of the number of spheres per clump 91

6.7 Number of spheres per clump has minimal effect on energy dissipation 92

6.8 Ice density has minimal effect on work done, strain energy and kinetic energy. 96

6.9 Close-up view of one bond breakage event 97

6.10 39 binary images comprise the image stack for 2 spheres. The image with the 1-pixel "bond" is outlined in red. 99

6.11 The binary images from Figure 6.10 before segmentation (left) and after (right). 100

6.12 The simple two-sphere geometry is subjected to shear in the YADE environment. Initial configuration (left) and post-failure configuration (right) 101

6.13 Energy released by failure of a simple sphere-sphere pair in tension (left) and shear (right). 102

7.1 No correlation between bond breakage x coordinate and simulation time, indicating no process simulating crack extension. Homogeneous sample 20161120-7mm1 104

LIST OF FIGURES – CONTINUED

Figure Page

7.2 Little linear correlation between bond breakage x coordinate and simulation time, indicating no process simulating crack extension. Sample 20170127-7mm with buried SH. 105

7.3 Plot of all R^2 values derived from plots like Fig. 7.1 in all homogeneous samples..... 106

7.4 Plot showing all R^2 values derived from plots like Fig. 7.2 in all samples with SH layer..... 107

7.5 z coordinates of bond breakage events in homogeneous sample 20161120-7mm1..... 108

7.6 z coordinates of bond breakage events in sample 20170127-7mm with buried SH. 109

7.7 Normalized FPZ height for all homogeneous samples. 110

7.8 Normalized FPZ height for all samples with buried SH. 111

7.9 Shear stress-shear strain plot for numerical subsample 20160803-7mm1..... 113

7.10 Plot of work done, stored elastic energy, and dissipation for numerical subsample 20160803-7mm1. 114

7.11 Dividing energy released by cross-sectional area (left) consistently yields low estimates from the model. Dividing energy by volume (right) yields consistently better results 115

7.12 In homogeneous samples, normalizing by sample volume yielded lower NRMSE than normalizing by sample cross-sectional area 116

7.13 In this subsample from test 20170127, there is very little strain softening and and abrupt drop in force, behavior that closely approximates brittle failure 117

LIST OF FIGURES – CONTINUED

Figure		Page
7.14	In this subsample from test 20170303, there is much strain softening and a gradual drop in force, suggesting fracture behavior that does not closely approximate brittle behavior	118
7.15	In samples with buried weak layer, normalizing by sample volume may yield a lower NRMSE than normalizing by sample cross-sectional area	119
7.16	Of samples with buried SH where NRMSE was evaluated for an area normalization, mostly low NRMSE is observed with three very high results. Two of these results were considered outliers.	120
7.17	Experimental values of G_c are compared to results estimated from DEM for homogeneous samples.	122
7.18	Experimental values of G_c are compared to results estimated from DEM for samples with buried SH.	122
8.1	Fracture size effect in homogeneous laboratory samples	129
8.2	Fracture size effect in samples with buried surface hoar	130
8.3	Minimum Cut G_c compared to actual experimental G_c in homogeneous samples	133
8.4	Minimum Cut G_c compared to actual experimental G_c in samples with buried SH	134
8.5	Ratio of experimental $G_{c(exp)}$ to minimum cut $G_{c(min)}$.	135
8.6	Interface crack in two elastic materials with different elastic moduli	136
8.7	Ratio of actual G_c to $G_{c(min)}$ as a function of ψ.	139
9.1	Summary of G_c for all tests and all models	143

ABSTRACT

A microstructure-based evaluation of snow combining experimental and analytical approaches was performed. Shear tests were performed on both homogeneous and layered samples of un-notched snow. Force and displacement during loading were recorded.

Immediately after testing, small subsamples of snow were subjected to micro-CT scanning to capture 3D microstructure details. Microstructure was then modeled as a grain-bond network. The grain-bond network was subject to minimum energy fracture path calculations as well as discrete element modeling. The discrete element model showed good agreement with experiments.

Taken together, results from models and experiments show a widespread damage accumulation process in snow. A large fracture process zone (FPZ) is observed, even in samples with weak layers. Evidence indicates that even in snow avalanches, there is likely significant energy dissipation within the slab.

1

BACKGROUND AND OBJECTIVES

Fracture Mechanics

Fracture mechanics is an alternative to the stress/strength approach to strength of materials. Fracture mechanics posits that failure originates at flaws or cracks in a material (Fig. 1.1), and failure is complete when a body is separated into two parts. The seminal paper on the subject was published in 1920 [28], and the body of work on the topic has expanded since. Griffith noted that bulk glass is weaker than the sum of the strength of its atomic bonds. Therefore, he hypothesized that the material must contain small flaws or cracks that effectively weaken it by concentrating stress around the crack tips. The original Griffith criterion for fracture states that while a material is undergoing failure [28],

$$\sigma\sqrt{a} = \sqrt{\frac{2E\gamma}{\pi}} \tag{1.1}$$

where σ is far-field tensile stress in the specimen, E is the elastic modulus of the material, and γ is the energy per area required to form a new surface of the material (J/m^2). The geometric parameter a is the *half-length* of the crack. In the edge crack shown in Figure 1.1, the crack has a dimension of crack half-length a. If the plate had a center crack, the flaw would have dimension $2a$.

The Griffith criteria assumes that materials behave in a brittle manner until they fail, i.e. plasticity is assumed negligible. The stress field in the plate of fig. 1.3 is variable, with infinite stress at the tip of the crack and σ is far-field stress. The local stress σ_y at any point (r, ϕ) in a cracked plate is given by [22]

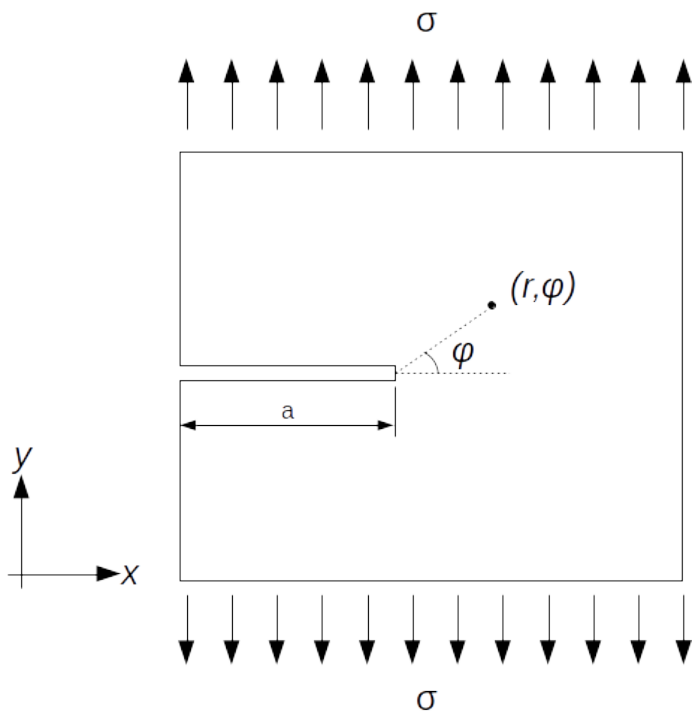


Figure 1.1: A plate of material with a crack of half-length a. The crack effectively weakens the material

$$\sigma_y = \frac{Y\sigma a^{0.5}}{\sqrt{2\pi r}} f(\phi) \tag{1.2}$$

where a is crack half-length and r is distance from the crack tip. ϕ is measured counterclockwise from the $+x$ direction at the tip of the crack. Y is a factor that depends on sample geometry and $f(\phi)$ is a function that depends on sample and crack geometry. These functions have been solved for numerous sample and crack geometries and are can be looked up in a reference such as [82].

Due to Griffith's conclusion of infinite stress at the crack tip, strength of a material could no longer be defined by a maximum stress. Instead, the new concept of *stress intensity factor* was invented. Griffith posited that a material would fail when the stress intensity factor was greater than the material's *fracture toughness.*

To make Equation 1.2 mathematically workable around the crack tip, the limit was taken as $r \to 0$. Stress intensity factor is defined as

$$K = \lim_{r \to 0} \sqrt{2\pi r}\sigma \tag{1.3}$$

Stress intensity factors depend on sample geometry and loading mode and are generally found in a reference such as [82]. Fracture toughness K_c, the stress intensity at the onset of failure, is a material property that can be found in a reference or determined experimentally.

Fracture Modes

Although Griffin's original work dealt with tensile cracks that propagated perpendicular to the direction of loading (Fig. 1.1), shear fracture modes are also part of the fracture mechanics framework. Mode is defined by the direction of loading relative to the direction of crack propagation. Fracture modes I, II and III refer to opening, in-plane shear, and out-of-plane shear, respectively (Fig. 1.2). In Mode II, fracture propagates in the direction of loading, while in modes I and III fracture propagates perpendicular to the direction of loading. Mixed mode fracture is also possible.

Stress intensity factors and fracture toughness vary with fracture mode, so that fracture toughness in Mode I is K_{IC}, while mode II stress intensity is K_{II} and so on.

Irwin's Modification

Of course, no material is perfectly brittle/elastic, and Eq. 1.1 yields the physically impossible result of infinite strain at the crack tip. Therefore, the Griffith criterion has been refined to account for plasticity at the crack tip. Irwin made the assumption that around the crack tip there exists a radius r_p^* wherein the stress is

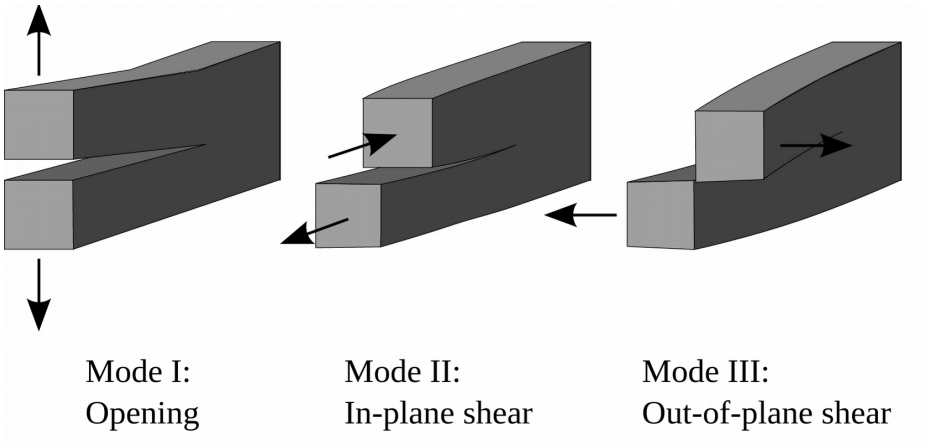


Figure 1.2: Three different fracture modes. Image: Twisp/Wikimedia Commons

$\sigma_y = \sigma_{ys}$ [13]. σ_{ys} is the yield strength of the material. In this zone the material behaves plastically, and it is therefore called the plastic zone. The plastic zone effectively lengthens the crack by length r_p^*. A correction factor must then be applied to compute K and K_c.

As work is done on a sample, some of that work is stored in the material as elastic strain energy. G is the energy available to drive crack growth [78]:

$$G = \frac{dU_w}{dA} - \frac{dU}{dA} \tag{1.4}$$

where U_w is work done on the sample and U is strain energy in the sample. A is surface area of the crack. If G reaches its critical value G_c, then the crack will propagate.

G_c is Irwin's [41] concept of "critical energy release rate". Confusingly, energy release rate has nothing to do with time, with SI units J/m^2. G_c may be thought of as the amount of energy required to increase the size of a crack by one unit area. For a perfectly brittle elastic material, $G_c = 2\gamma$ where γ is the surface free energy of a material, and the coefficient 2 denotes the creation of two new surfaces under

fracture. The $G_c = 2\gamma$ relation should be though of as a lower bound for the true G_c of a material, since any plasticity or a finite fracture process zone will increase G_c.

For brittle fracture, this condition (Fracture propagates when $G \geq G_c$) is equivalent to the condition where fracture propagates when $K \geq K_c$. In plane stress, the terms are related by [13]

$$K_{IC} = \sqrt{EG_{Ic}} \tag{1.5}$$

and the relation for plane strain is

$$K_{IC} = \sqrt{EG_{Ic}(1-\nu^2)} \tag{1.6}$$

Similar relations may be used for mode II and mode III fracture, where K_{IIC}, K_{IIIC}, G_{IIC} and G_{IIIC} are used instead.

When K and G are below their critical values,

$$K_I = \sqrt{EG} \tag{1.7}$$

and the relation for plane strain is

$$K_I = \sqrt{EG(1-\nu^2)} \tag{1.8}$$

Diffuse Failure

Not all materials create a neat, planar fracture surface as described above. Certain geomaterials have been shown to undergo a diffuse failure process wherein damage accumulates throughout a volume of material. Hill [36] describes a criterion where material *instability* is defined. Instability does not necessarily indicate total

failure of a material, but rather a momentary strain energy release that may or may not culminate in total failure. Hill's criterion states that for a volume V of material subject to stress $\vec{\sigma}$ and strain $\vec{\epsilon}$, the material is unstable when

$$\int_V d\vec{\sigma} : d\vec{\epsilon} dV < 0 \tag{1.9}$$

Physically, the criterion means that if strain increases while stress decreases, or vice versa, the material is unstable. Experimentally, material instability may be seen as a negative slop on the stress-strain curve.

Dry Slab Avalanches

Snow avalanches can be classified into two distinct groups: slab avalanches and point releases. Point releases initiate as loose snow on the surface runs downhill [84]. They are not considered in this research. Slab avalanches are occur when a cohesive slab of snow at the surface slides down a slope. Slab avalanches are further subdivided into wet slabs and dry slabs. Dry slab avalanches are caused by overloading a weakness in the snow pack, whereas wet slab avalanches are caused by a weakening of the snowpack due to melting. These two types of avalanches occur in different meteorological settings and have very different mechanical characteristics, although both can carry and bury victims (e.g. skiers or snowmobilers), cause structural damage and close roads. This research will only consider fracture in dry slabs. Dry slab avalanches result when a slope parallel failure (that may or may not involve collapse of a weak layer) occurs at an interface between two layers or within a poorly bonded weak layer in the snowpack (Fig. 1.3). Various meteorological conditions can lead to the formation of these weak layers.

Understanding the weak layer formation and development in the context of underlying ice microstructure is of critical importance. Microstructural changes resulting from metamorphism significantly influence the strength and stability of snowpacks. Snow continually changes as it sits on the ground, and can weaken or strengthen over time, influencing avalanche risk.

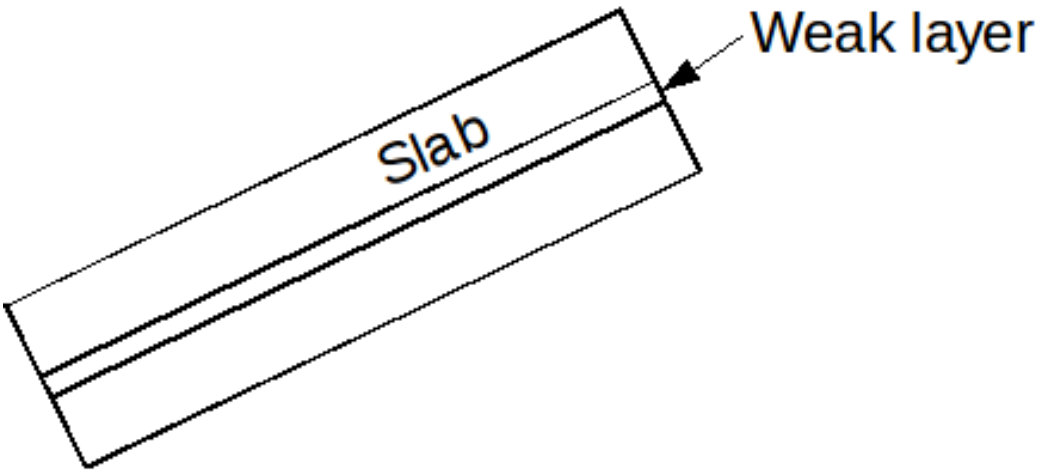


Figure 1.3: Slab and weak layer stratigraphy of avalanches

Snow Metamorphism

Typically, dry snow will metamorphose toward either a weaker faceted granular structure or to stronger rounded sintered ice grains. Both processes may occur simultaneously at different levels within the same snowpack. Which of these processes dominates is largely dependent on the magnitude of the temperature gradient. Temperature gradients greater than $10°Cm^{-}1$ [15] drive water vapor pressure gradients that lead to a fragile, faceted snow layer. The angular, poorly bonded crystal morphology that results is termed the kinetic growth form [15]. Three subtypes of kinetic forms are discussed below. Lower temperature gradients yield rounded sintered grains, commonly referred to as the equilibrium growth form [16].

Temperature Gradient Metamorphism

Water vapor pressure gradients in the snowpack are generally driven by spatial temperature gradients. Assuming saturated conditions, warmer areas have higher vapor pressures, and colder areas have lower vapor pressure. Temperature gradient is an easy measurement to take in the field, and the process is termed *Temperature Gradient Metamorphism.* Temperature gradients greater than $10^oC/m$ result in *kinetic* metamorphism and the snow therein trends toward unstable, faceted forms [70].

The most common result of temperature gradient metamorphism is depth hoar. Any ground surface not containing permafrost stays at around 0^oC. Meanwhile, winter air temperatures are often much lower. If the gradient through the depth of the snowpack is greater than about $10^oC/m$, depth hoar forms. During the process, water vapor coming off warm snow grains near the ground moves upward to fill in the void created by the low-temperature, cold air above. The process weakens the snowpack by forming angular, poorly bonded crystals (Fig. 1.4). Depth hoar is often described as "sugary" for its granular texture and brittle behavior.

Radiation Recrystallization Radiation recrystallization is a specific type of temperature gradient metamorphism that occurs near the snow surface on clear, sunny days. Solar shortwave radiation heats the shallow subsurface of the snowpack. Meanwhile, the surface of the snow emits longwave radiation to the cold clear sky [9]. This combination of radiative heating/cooling processes has been shown to create temperature gradients of over $100^oC/m$ in the near subsurface [10]. If the facets are preserved long enough to be buried by new storm snow, they create an unstable weak layer.

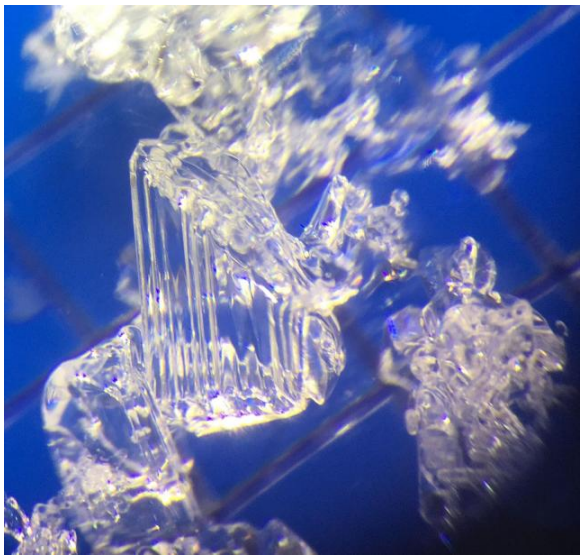


Figure 1.4: Striated, cup-shaped crystals are typical of depth hoar. The angular edges form poor bonds with neighboring crystals. Photo: K. Birkeland/USFS.

Surface Hoar Surface hoar forms when ambient water vapor condenses on the snow surface. Deposition occurs when the snow surface is colder than the air around it, just as water vapor commonly deposits in liquid form on other surfaces that are colder than their surroundings (Fig. 1.5) due to temperature-induced pressure gradients. Surface hoar layers are thin, weak and bond poorly to the snow around them.

To form surface hoar, the surface of the snow cools via long-wave radiation emission to the sky [70]. Meanwhile, moisture-laden air flows gently over the surface and acts as a source of water vapor for deposition.

Equilibrium Growth

Temperature gradients less than $10^oC/m$ create smaller pressure gradients. As a result, small pressure gradients around each ice grain dominate metamorphism [16]. Water vapor pressure near an ice surface is dictated by that surface's radius of curvature and surface free energy. Convex surfaces with smaller radii of curvature (i.e. sharp corners) have the highest vapor pressure. Conversely, air near concave



Figure 1.5: Condensation occurs on objects that are colder than the ambient air. Vapor may precipitate in liquid form on the outside of the glass (left) or solid form (right). Photo: K Birkeland/USFS

surfaces has the lowest vapor pressure (Fig. 1.6). As a result, water vapor flows from sharp convex corners toward flat or concave portions of the ice grain surface. As this process continues, sharp corners round and the crystal shape tends toward spherical, well-bonded grains.

Fracture in Snow

Fracture in snow avalanches is characterized by two macroscopic processes: slope-parallel fracture propagation and, in general, slope-normal collapse of the weak layer. Fracture initiates in the weak layer or at the weak layer-slab interface. It is assumed that fracture initiates at so-called super-weak zones [3] within the weak layer, analogous to the flaws studied in fracture mechanics. When flaws reach critical size, or when load is sufficient, fracture propagates. In practice, additional loading from storms, skiers or explosives can be sufficient to create propagating avalanche

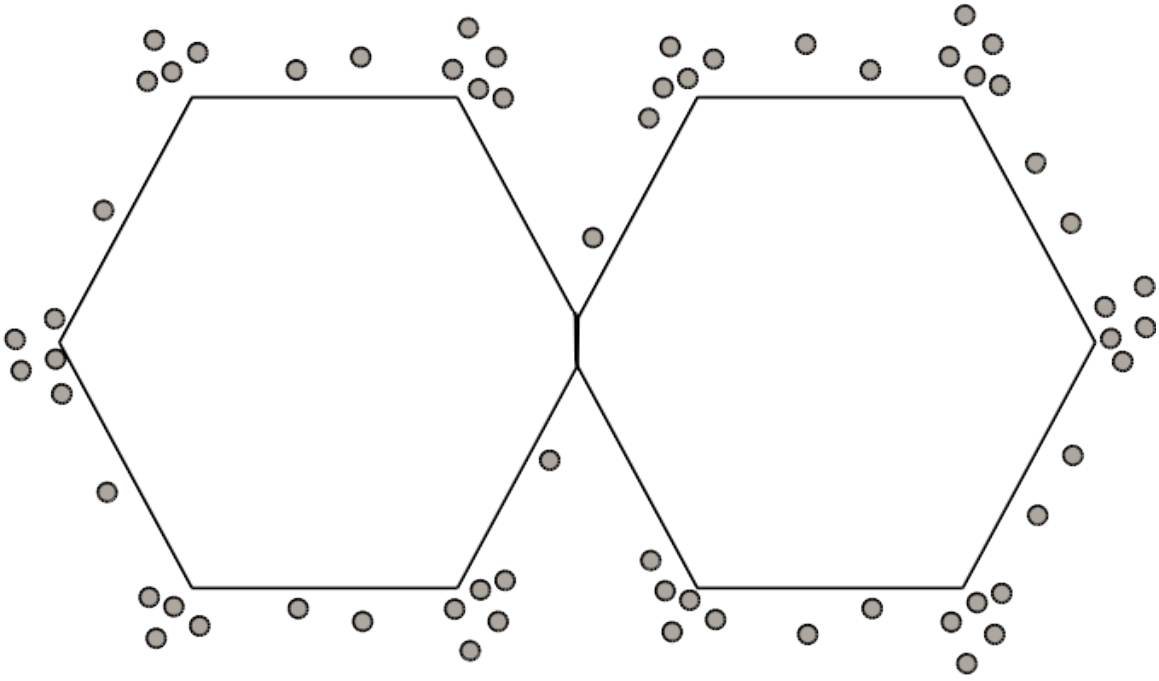


Figure 1.6: Water vapor particles concentrate around sharp, convex corners and will flow to concave areas.

fractures. The role and timing of slope-normal collapse is not universally agreed upon.

Fracture propagation models require material properties, namely critical energy release rate (G_c) or fracture toughness (K_{IC} or K_{IIC} for mode I or II), to be known. In addition, strain energy may be released through nonelastic mechanisms such as strain softening or microcracking. The mechanism depends on the material, although all materials display some plasticity.

The Role of Strain Softening and Damage Accumulation

If plasticity around the crack tip is insignificant, a material may be studied under the framework of linear elastic fracture mechanics (LEFM). At high strain rates, snow is typically approximated as an elastic-brittle material [23] [65] [49]. Therefore, fracture in snow is often studied under the simplifying assumption that a sufficiently

small fracture process zone (FPZ) allows use of LEFM. Although there will always be some energy dissipation near the crack tip, if the FPZ is small relative to the crack length, the LEFM solution may closely approximate fracture energy. However, the linear elastic simplification is not universally accepted.

There is some evidence for strain softening and damage in snow even at high strain rates. Borstad and McClung [12] find evidence of strain softening in beam bending tests involving both notched and un-notched samples of homogeneous snow. They note that un-notched samples show considerably more strain softening than notched samples. Un-notched samples were also stronger. McClung [58] assumes a finite damage zone, but notes that it "may be a significant fraction of the slab depth."

Current Macroscopic Fracture Models

In stratified snow, fracture has been modeled using energy principles applied to the overlying slab resulting in weak layer failure. Prevalent macroscopic snow fracture theories have their roots in geomechanics [56] [34]. Historically, fracture propagation has been treated as pure mode II shear fracture (e.g. [3, 5, 56, 57, 61]).

David McClung's pioneering work of snow fracture modeling [56] was based on a shear band soil model developed by Palmer and Rice [66]. The McClung model is based upon a *progressive failure* concept: Loading causes strain softening (localized damage) in the weak layer until a critical state is reached. Then, fracture propagates rapidly. The quasi-brittle assumption used implies that the fracture process zone is finite. The slab is assumed linear elastic, and deformation in the weak layer is one-dimensional in the slope-parallel direction. Collapse is not accounted for in fracture propagation as it is assumed to be a dynamic after-effect.

McClung's 2009 update to the model [58] accounts for slope-normal deformation in the weak layer, but still treats the fracture as pure mode II. By McClung's

approximation, slope-normal collapse accounts for a change in graviational potential energy of $0.6 - 1.2\,J/m^2$, far higher than recorded G_c values. He uses this fact to argue that while slope-normal deformation may aid in fracture initiation, collapse must still be a dynamic after-effect of failure.

Modifications to account for a large FPZ in the weak layer and plasticity in high-stress regions of the slab [58] have been considered.

McClung [59] also proposed a slightly different approach based on an assumed FPZ size for quasi-brittle fracture behavior, without considering behavior of the slab. The method does not consider energy expenditure in the slab. Instead, it is based upon assumed deformation and FPZ size in the weak layer.

An alternative to the shear concept, first proposed by Heierli et al [34], hypothesizes that fracture behaves as an interface crack between the weak layer and the overlying slab. Energy for fracture still comes from the overlying slab, but snow fracture is driven by a simultaneous combination of collapse and shear which may be characterized as a mixed-mode fracture consisting of mode II and a mode I anticrack process. A mode I anticrack is a fracture that is in the opposite direction of a classical mode I fracture, i.e. the fracture faces are initially separated by a porous material and then come together as the material at the crack tips fails in compression. Then, the porous material densifies rather than undergo a traditional mode I fracture. The anticrack model has been applied to fracture in rock faces separated by partially dissolved rock, compaction bands in sandstone [81], and in brittle foams under compressive stress [33].

As applied to snow, the anticrack model posits that energy for fracture can come from both slope-normal and slope-parallel deformation of the slab [34] and that collapse is part of the failure process. To model the slab, a Timoshenko beam is used that considers both bending and shear deformations of the slab. As the beam deforms,

the weak layer collapses and fracture propagates from the crack tip. Furthermore, the model allows for fracture propagation in the weak layer without slab release. This allowance accounts for the well documented phenomenon of whumpfing where a weak layer fails but a slab does not release. Avalanche slab release, then, is a separate phenomenon that requires sufficient slope-parallel force to overcome friction between the now-touching crack faces. Some experimental evidence shows simultaneous shear and collapse of the weak layer [86], but the contribution and timing of collapse is not universally accepted [58].

Anticracking offers an interesting approach to address the collapse on flat slopes, although it is more simplistic in the mechanical properties assigned to snow than the McClung shear model. The Heierli anticrack model [34] assumes the weak layer to be rigid but collapsible, and that fracture occurs at the interface between weak layer and slab. This implies that all deformation occurs in the slab, and the role of deformation in the weak layer prior to failure is neglected. In contrast, McClungs shear approach assumes that the dominant deformation is in the weak layer rather than the slab.

The Heierli model also assumes linear elastic deformation in the slab, where McClung's more recent work has considered non-recoverable work on the slab [58]. Heierli's simplifications allow the model to draw on previous work for mixed-mode fracture between layers in elastic laminates [40].

Field methods have also been used to determine weak layer critical energy release rate. However, field studies have typically consisted of taking observations, applying a specific propagation model, and then back-calculating the relevant material property. This technique has been used with both the anticrack model [72, 85] and the shear model [58, 59, 61, 62].

Several papers have been published that performed notch experiments on weak layers in the field, then apply a model to back-calculate G_c. Results of these tests are

summarized in Table 1.1. Although all methods vary slightly in execution, these are all estimates of G_c derived from analyses of weak layers in the field. Estimates vary from 0.015 J/m^2 to 2.7 J/m^2. This wide variation of G_c in weak layers is problematic. One explanation is proposed here: The anticrack model assumes there is no damage in the slab, and therefore attributes all energy release to the weak layer. In contrast, some iterations of the shear model do permit damage/plasticity in the slab, or do not consider energy in the slab at all. Therefore, they may more precisely isolate the energy release contribution of the weak layer while not counting energy released in deformation of the slab.

Schweizer et al [72] performed one of the first field studies to apply Heierli's anticrack model to notch tests in snow. They estimated slab elastic modulus using a snow micro penetrometer (SMP), and calculated mixed-mode G_c from slab modulus and critical crack length according to Heierli's analysis:

$$G_c = \frac{H}{2E}\left[w_0 + w_1\frac{r}{H} + w_2\frac{r}{H^2}^2 + w_3\frac{r}{H}^3 + w_4\frac{r}{H}^4\right] \tag{1.10}$$

where H is slab height, E is slab elastic modulus, and r is cut length. The terms w_1, w_2, w_3 and w_4 are functions of overburden stress induced by the slab on the weak layer. They found G_c on the order of 1 J/m^2, and noted that these results were significantly higher than other published results for G_c.

van Herwijnen et al [85] performed a similar analysis, but employing particle tracking velocimetry (PTV) to analyze the deformation field of the slab and compute the slab elastic modulus. Their study yielded mixed-mode G_c of $0.08 - 2.7\, J/m^2$.

Field analyses based on various iterations of the quasi-brittle shear model have typically yielded much lower results for weak layer G_c. Estimates from actual avalanches [61] were $G_{IIc} = 0.006 - 0.1\, J/m^2$ based on McClung and Schweizer's [60]

analysis of snow with weak layers. G_{IIc} was higher in homogeneous snow, aroud $0.7 - 1.4\, J/m^2$.

Table 1.1: Summary of critical energy release rate G_c calculated from field studies

Study	$G_c(J/m^2)$	Model
Schweizer et al (2011)	≈ 1	Anticrack
McClung (2007)	0.015	Shear
Schweizer et al (2016)	0.6-2.2	Anticrack
Sigrist & Schweizer (2007)	0.07	FEA
van Herwijnen et al (2016)	0.08-2.7	Anticrack

In addition to the closed-form solutions mentioned above, numerical modeling approaches have been applied to study fracture propagation. Layered snow under mixed-mode loading with cohesive fracture faces has been studied in finite element models to predict propagation rates and critical crack length [54, 55].

Gaume et al [24] offer an interesting numerical approach that purports to emphasize the importance of the slab-weak layer interaction, something that has arguably been glossed over in the closed-form solutions discussed above. They use a discrete element model that has a rigid substrate and a slab modeled as square-packed spheres. The slab elements are not microstructural; rather they constitute a continuum approximation. The weak layer is modeled as collapsible triangular packings of spheres so that it may mimic whatever sort of mixed-mode failure envelope realistically simulates real WL failure. Although this offers an interesting approach, G_c was used as an input to the model to estimate critical crack length.

Microstructural Fracture Mechanics

Ultimately, fracture in snow is a microstructural process arising from the destruction of bonds and rearrangement of grains [58]. Although very little research

has been done on the micromechanical fracture process in snow [29,51], work has been done with other bonded granular materials. Closed-form microstructural formulations for fracture toughness have been proposed for cellular materials with regularly repeating microstructures [25] and applied to snow [46] as a function of snow density.

More recently, higher-powered computers and 3D imaging techniques have allowed researchers to create numerical models to model the failure in real snow microstructure. Walters [88, 89] has used finite elements to model linear elastic deformation in snow microstructure. However, the finite element model does not detect new contacts between elements as the sample fails and particles rearrange. Then, the dynamic aspects of microstructural failure are not adequately simulated, making traditional FEA inadequate to model the dynamic failure process.

Discrete element modeling (DEM) is better suited for microstructural fracture modeling. DEM is a dynamic modeling technique where Newton's laws are applied to moving particles. The use of dynamic elements and contact detection means that phenomena of grain rearrangement and contact failure may be better modeled. The DEM modeling technique used in this work is described in detail in Chapter 5.

Hagenmuller et al have modeled compressive failure in snow using dynamic, cohesive discrete elements created from real snow microstructure [31]. It is believed that microstructure-based discrete element modeling of shear fracture in snow has never been done.

Hagenmuller et al [32] have also modeled microstructural failure of snow under tension using a finite element approach. Results showed nonlocalized failure with damage occurring prior to global failure of the sample. There is some variation in the reported size of the FPZ ranging from $5 - 10$ cm [58] to ≈ 100 times the grain size [60]. The relative size of the FPZ further supports the need for microstructural

consideration. Since the fracture process zone is on the order of the thickness of some weak layers, there is a size effect on fracture toughness measurements [74].

Reiweger et al propose an approach that involves modeling snow using a bundled fiber model [68]. The model creates fibres with statistically assigned strengths between two plates to simulate a weak layer between two slabs of snow. The fibers are then failed under loading. This was one of the first statistical models to show strain softening and damage accumulation in snow that matched the significant strain softening in experimental stress/strain curves. The model also reproduced the brittle-ductile transition at different strain rates.

When considering micromechanics of heterogeneous materials, the Representative Volume Element (RVE) must be considered. RVE is the smallest volume of a heterogeneous material that adequately mimics the material's macroscopic mechanical properties. RVE in rounded snow has been shown to be $2.8^3 mm^3$ in tension [32], and $4^3 - 5^3 mm^3$ in compression [31]. Walters [89] used microstructural models with cross-sections of $5 \times 5\, mm$.

Fracture in Snow: Laboratory Experimental Approaches

To complement models, several efforts have been made to determine fracture toughness and/or energy release rate experimentally. Several laboratory experiments in both tension and shear have used notched samples and used Griffith's criterion (Eq. 1.1) to calculate the fracture property. Sigrist et al [75] found the energy release rate for notched snow samples with weak layers to be $0.04 \pm 0.02 Jm^{-2}$, and that this energy release rate was primarily a function of the weak layer properties and not the surrounding slab.

K_c and G_c are often approximated as a function of density [60] [49], although density can be an unreliable predictor of snow properties. Results suggest that K_{IIC}

and K_{IC} are both roughly equal in recently deposited snow [47], but K_{IIC} effectively doubles when friction between touching crack faces is considered [48]. In weak layers, K_{IIC} is approximately one fourth K_{IC}.

Graph Theory

Graph theory is a broad branch of mathematics that concerns any system that can be represented as a network. Since snow microstructure is a network of bonds and grains, graph theory will be discussed here.

The field was famously invented by Swiss mathematician Leonhard Euler in 1736 [7]. The city of Konigsberg, Prussia was bisected by the Pregel river (Fig. 1.7), and connected by seven bridges. Euler wished to determine if it was possible to create a route across the city that crossed each of the seven bridges exactly once . To carry out the study, Euler created a mathematical abstraction of the land and bridges in Konigsberg. Each distinct landmass (1-4 in Fig. 1.7) was represented as a node, and each bridge was represented as an edge between the two nodes it connected. This representation, consisting of edges and vertices, is shown in Figure 1.8.

Formally, a graph is defined as a finite set of vertices connected by a finite set of edges. Graph theory is a useful way to study all manner of networks, such as shipping routes, social networks, data networks, etc.

Numerically, a graph may be represented as a matrix G where the edges are represented by a number and are placed at the position in the matrix that corresponds to the two vertices the edge connects. For example, for a graph G of the Konigsberg bridges, there are two bridges connecting nodes 1 and 2. So, $G_{12} = 2$. Since bridges may be traversed either direction, $G_{21} = 2$ as well. Since no node connects to itself, all diagonals $G_{ii} = 0$. A complete matrix for the Konigsberg bridge graph is shown in Equation 1.11.

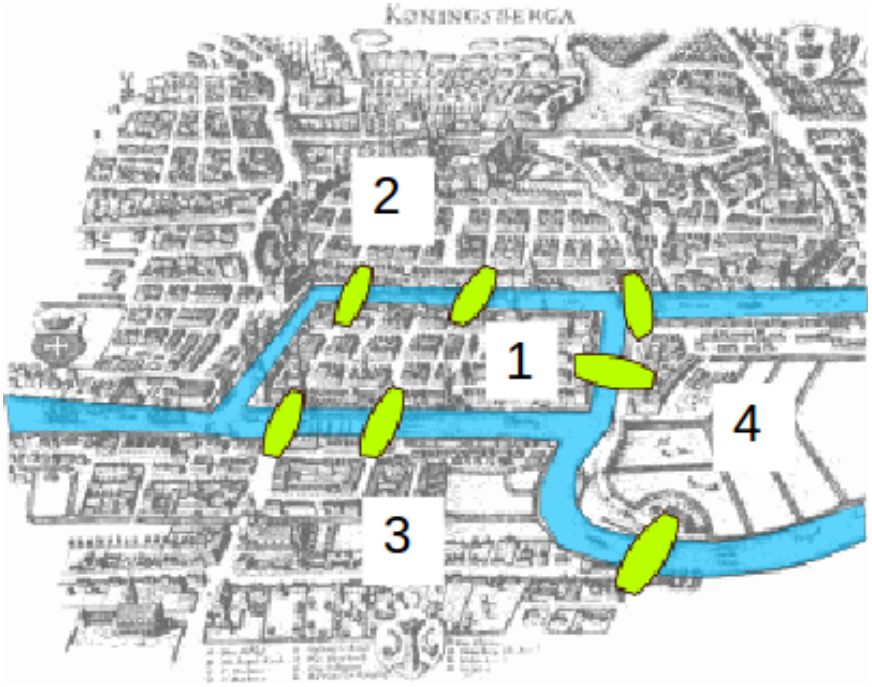


Figure 1.7: Euler's seven bridges problem that spurred the creation of graph theory

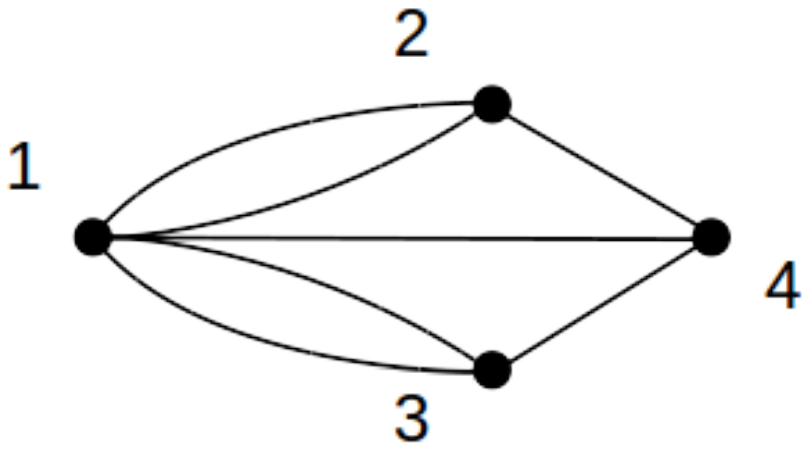


Figure 1.8: The seven bridges of Konigsberg represented as a graph, showing how bridges connect the four landmasses.

$$G = \begin{bmatrix} 0 & 2 & 2 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \tag{1.11}$$

A common problem in graph theory is that of the *minimum cut*. The minimum cut problem seeks to separate the vertices of a graph into two separate subsets by minimizing some parameter. In the case of the bridges, a minimum cut problem could be formed as an attempt to separate nodes 1 and 4 into to separate graphs:

How can an invading army isolate landmass 1 from landmass 4 while minimizing the number of bridges destroyed?

This is a minimum cut problem because it is desired to minimize the number of cuts to separate the two nodes. Certainly, the bridge from 1 to 4 could be destroyed, but nodes 1 and 4 would still be connected by way of nodes 2 and 3. A path would still exist. At a minimum, three bridges must be destroyed: the bridge from 2 to 4, the bridge from 3 to 4, and the bridge from 1 to 4. This is the *minimum cut* if nodes 1 and 4 are to be separated.

The minimum cut may also be computed for large, complex graphs where the solution is not immediately obvious. This is accomplished via the *max-flow min-cut theorem*, which states that the value of maximum flow between two nodes is also the minimum cut between those two nodes [20]. Imagine a flow of people crossing bridges to travel from node 1 to node 4. Certainly the paired bridges from 1 to 2 would facilitate a high flow of travelers, as would the paired bridges from 1 to 3. However, there would be a bottleneck at 2 as a double-bridgeload of travelers had to fit on the one bridge from 2 to 4. The same would happen at node 3. Ultimately, a flow of people traveling from 1 to 4 is limited to three bridges' worth. Not coincidentally,

three is also the minimum number of bridges that can be destroyed to separate 1 and 4.

Several algorithms have been created to solve the max-flow min-cut problem by finding the max flow. The first such algorithm was proposed by Ford and Fulkerson [20]. They state that for a flow from node s to node t, each vertex i must have a capacity c_i. The algorithm finds any path forr flow from s to t. Then, the algorithm finds any *augmenting path* that will increase the total flow. This is repeated until no augmenting path can be found.

Edmunds and Karp [18] tweaked the Ford-Fulkerson algorithm by always choosing the augmenting path with the fewest vertices, leading to increased computational efficiency.

Research Objectives

The objective of this research is to use a micromechanical approach to simulate energy release during fracture in snow and ultimately find G_c. By using a micromechanical approach, assumptions about macroscopic fracture mode (i.e. shear vs anticrack) can be avoided and the nature of the failure process can be observed. It is hoped that this work will provide insight into the macroscopic mechanisms of fracture and shed light on the shear vs. anticrack debate.

The approach consists of six steps:

1. Perform laboratory shear tests of actual snow. Both homogeneous snow and layered snow with weak layers will be considered (Chapter 2)

2. Obtain 3D snow microstructure imagery of that snow using a micro-CT scanner and process the images so the mechanical response of snow may be modeled (Chapter 3)

3. From microstructure data, numerically simulate shear failure using two methods:

   (a) A minimum energy criterion (Chapter 4)

   (b) Discrete element modeling (Chapter 5)

4. Validate the model (Chapters 6 and 7)

5. Record information that can not be obtained through physical experiments. This includes (Chapter 7):

   (a) Location and timing of every bond breakage event

   (b) Exact amount of strain energy stored in system

6. Determine what meaningful conclusions can be drawn about the snow fracture process. What does it reveal about the validity of assumptions used in other snow fracture models, and in turn the validity of those models? (Chapter 8)

   (a) What is G_c for these samples?

   (b) How large is the FPZ?

   (c) Does snow follow a conventional fracture propagation? Or is the Hill instability criterion or damage mechanics approach more appropriate?

EXPERIMENTS

This work centers around a series of laboratory shear experiments on nature-similar snow. Both homogeneous slabs and layered samples were tested. This chapter will describe the experimental methods used to determine G_c. Subsequent chapters will describe how actual snow samples from these experiments were processed and modeled.

In order to replicate fracture in natural snow, un-notched samples were used. It was hypothesized that notched samples provide an artificial stress concentration that may limit the size of the fracture process zone and not realistically simulate microcracking and/or real-world damage accumulation. The idea was to let the naturally existing super-weak zones [3] within the sample initiate failure on their own. In an experiment, a super-weak zone could be a small zone in the snow without any ice grains (essentially an air pocket) that serves as a notch to concentrate stress. The super-weak zone could be an especially weak portion of the weak layer. Without a notch, whatever flaws or weaknesses exist in the sample will serve to initiate fracture.

Each experiment is referred to by the date on which it took place in a YYYYMMDD format. So, for example, sample 20160809 was tested on August 9, 2016.

Procedure

Laboratory Snow Breeder

Most snow for experiments was made using a laboratory snow breeder in the SSERF. Bones and Adams [11] built a small table-top version that has been scaled up and refined by various SSERF researchers. In principle, the laboratory breeder creates snow the same way snow forms in nature: A humid air mass is cooled to

induce supersaturation, and as a result ice crystals (snowflakes) form upon nucleation sites [70]. This process is replicated using the setup shown in Fig. 2.1. The temperature of the room is set to -25^oC, and fans (not in photo) blow air across the room. As air hits the far wall, it is forced to enter the "tunnel" portion of the snow breeder. At the entrance of the tunnel, liquid water is forced through a highly porous material, specifically a reusable furnace filter. As air blows through the furnace filter, it accumulates moisture and becomes humid. The humid air continues to blow through the length of the tunnel, where it cools to supersaturation and ice crystals nucleate on the strings hanging in the vertical chimney. To collect the newly formed snowflakes, an operator's hand applies an opened-palm blow ("smack" or "karate chop") upon the side of the chimney. The applied force vibrates the strings, releasing the loosely attached snowflakes to fall into a box below for collection.

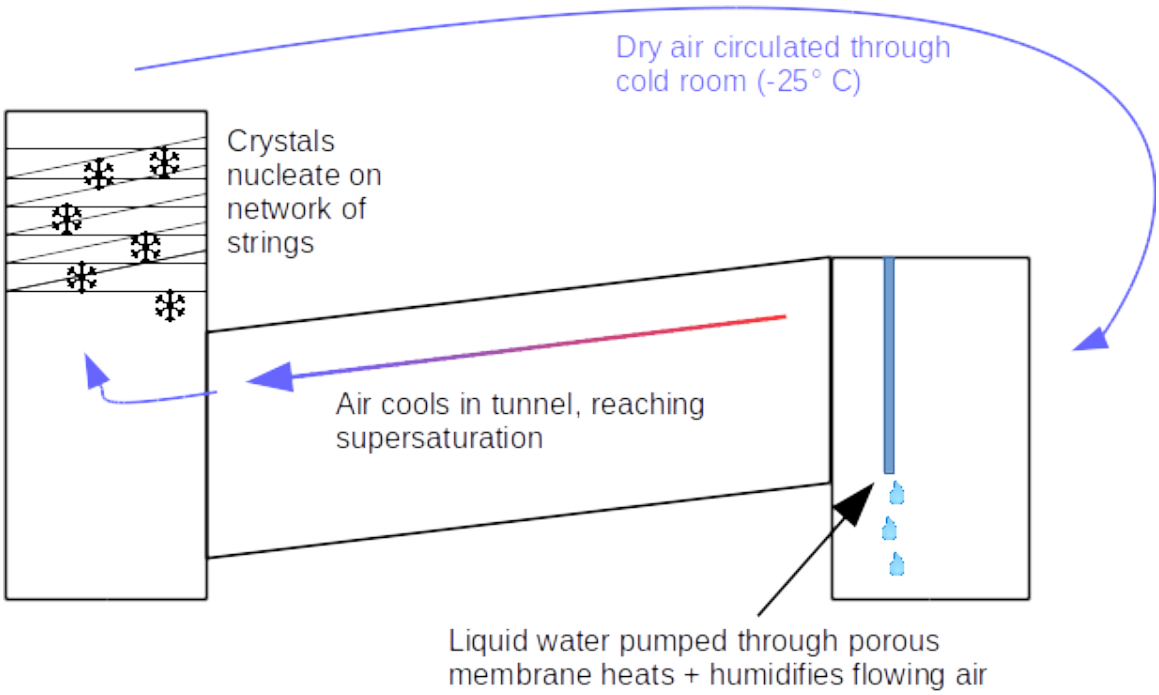


Figure 2.1: The laboratory snow breeder is a convenient way to create nature-similar snow crystals year round.

Procedure for Homogeneous Samples

All shear tests were performed in a large box with an attached load actuator and load cell. Old ($>$ 1 year), well sintered snow was in place in the base of the box prior to experiments described here. This strong, heavy snow served as a stout base (fixed boundary condition) upon which new, weaker snow was tested experimentally. Before adding snow to test, old snow from previous tests was scraped away, leaving a surface upon which to put new snow.

New snow from the laboratory snow breeder was sifted into a box using a # 10 sieve (nominal opening 2mm). In one experiment, natural dry snow collected $<$ 24 hours after it fell was used instead. A small cylindrical container used for CT scans was placed in the box as well. A 15.5×15.5 cm shear frame was placed on top of 3-4 cm of snow, and more snow was sifted into the shear frame until it was full. The snow was allowed to sit undisturbed for 12-24 hours in order to let grains sinter and create a strong, well-bonded snowpack. Snow around the shear frame was then removed using the thin metal blade of a paint scraper, creating a rectangular specimen. The snow was then loaded in shear at a rate of 0.85 mm/s using a Geojac Geo-Tac load actuator. Force F was measured using a Transducer Techniques MLP-25 load cell connected to a Transducer Techniques LCA-9PC load cell amplifier. Room temperature during sintering and testing was -15^oC. In later tests, a 15.5x15.5cm flat plate was thermally welded to the top of the snow specimen in place of the shear frame. This setup is photographed in Figure 2.2.

Samples With Weak Layers

The testing procedure was similar for samples with weak layers, but a weak layer of surface hoar first had to be grown and then buried. The surface hoar growth method and apparatus were primarily refined by Brad Stanton [80] and more recently

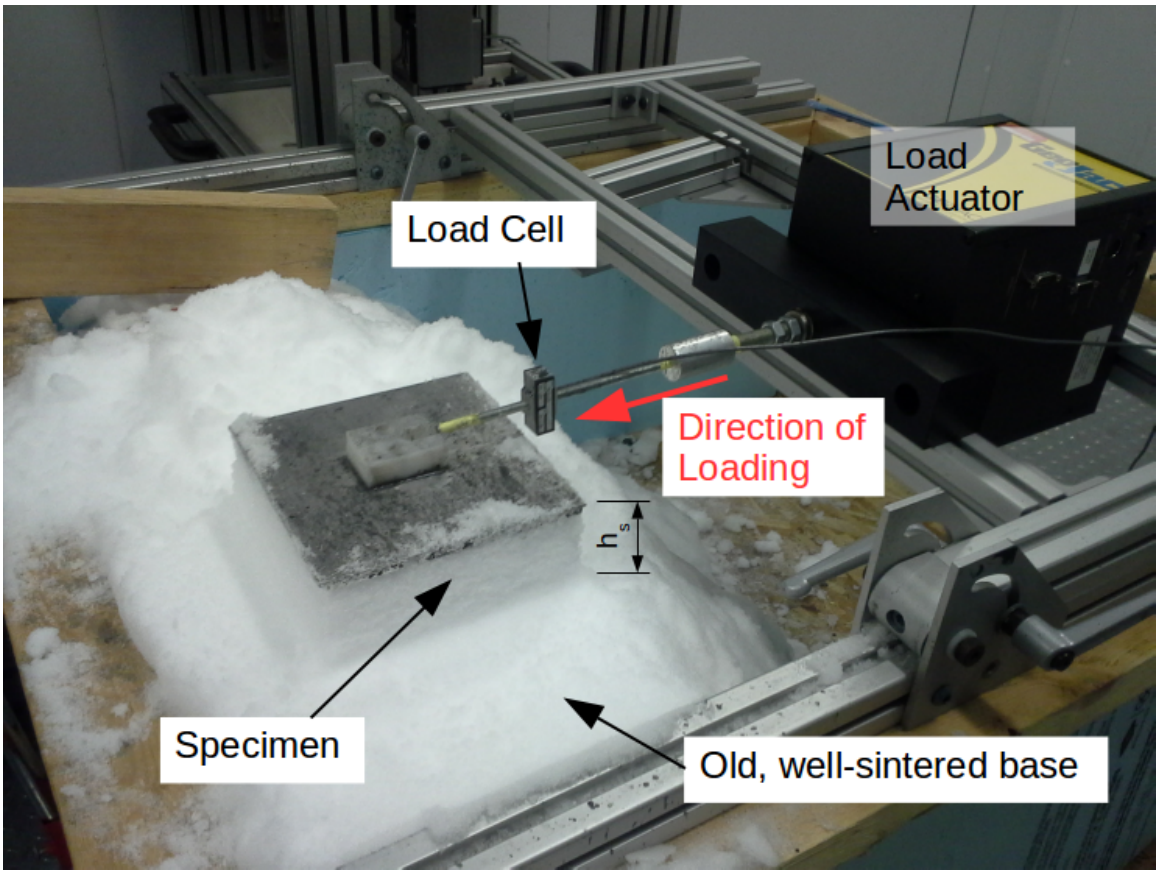


Figure 2.2: A load cell measures force as the load actuator advances at a constant velocity

by various other SSERF researchers. As noted in the literature review, surface hoar forms when moist air moves slowly across snow and crystallizes at the surface (See Ch. 1). For the moisture in the air to crystallize on the snow surface, the surface must be colder than the air around it. In nature, cooling is accomplished through radiative heat loss to a clear sky. In the SSERF, the process is mimicked by cooling the ceiling above the snow sample to -50^oC while the air temperature in the room stays at -15^oC. In this way, snow radiates heat to the ceiling just as natural snow radiates heat to the sky.

The surface hoar making apparatus is shown in Figs. 2.3 and 2.4. A tank of liquid water is kept at a constant 32^oC using a tank heater and a thermostat. The tank is kept inside an insulated box. Fans are mounted to the side of the box and push the warm, humid air found above the tank through ducts. The ducts are wrapped with rope heaters at the box to prevent ice accumulation on the fans. Duct outlets are mounted to blow air horizontally across the snow surface.

The snow surface was scraped flat, and then the surface hoar maker was activated. Surface hoar was grown for 6-8 hours, producing approximately 2mm crystals. The surface hoar was then buried by sieving homogeneous rounded grain snow through a #10 sieve (nominal opening 2mm) and burying the weak layer several centimeters deep. The slab was then allowed to sinter for approximately 24 hours. For exact sintering times, see Appendix D.

After the slab was sintered, a 15.5x15.5cm plate was thermally welded to the top of the slab. The plate was then loaded at 0.85 mm/s using a Geojac Geo-Tac load actuator. Force F was measured using a Transducer Techniques MLP-25 load cell connected to a Transducer Techniques LCA-9PC load cell amplifier.

Collection of CT Samples

During each experiment, some of the snow was removed from the box for CT scanning. This section will explain the snow sample extraction process. For details of the CT scanning procedure, see Ch. 3.

For convenient snow extraction, a 3D-printed snow sample container was used. The container consists of a base that attaches to the CT scanner's specimen mount and a hollow cylinder that snaps into the base.

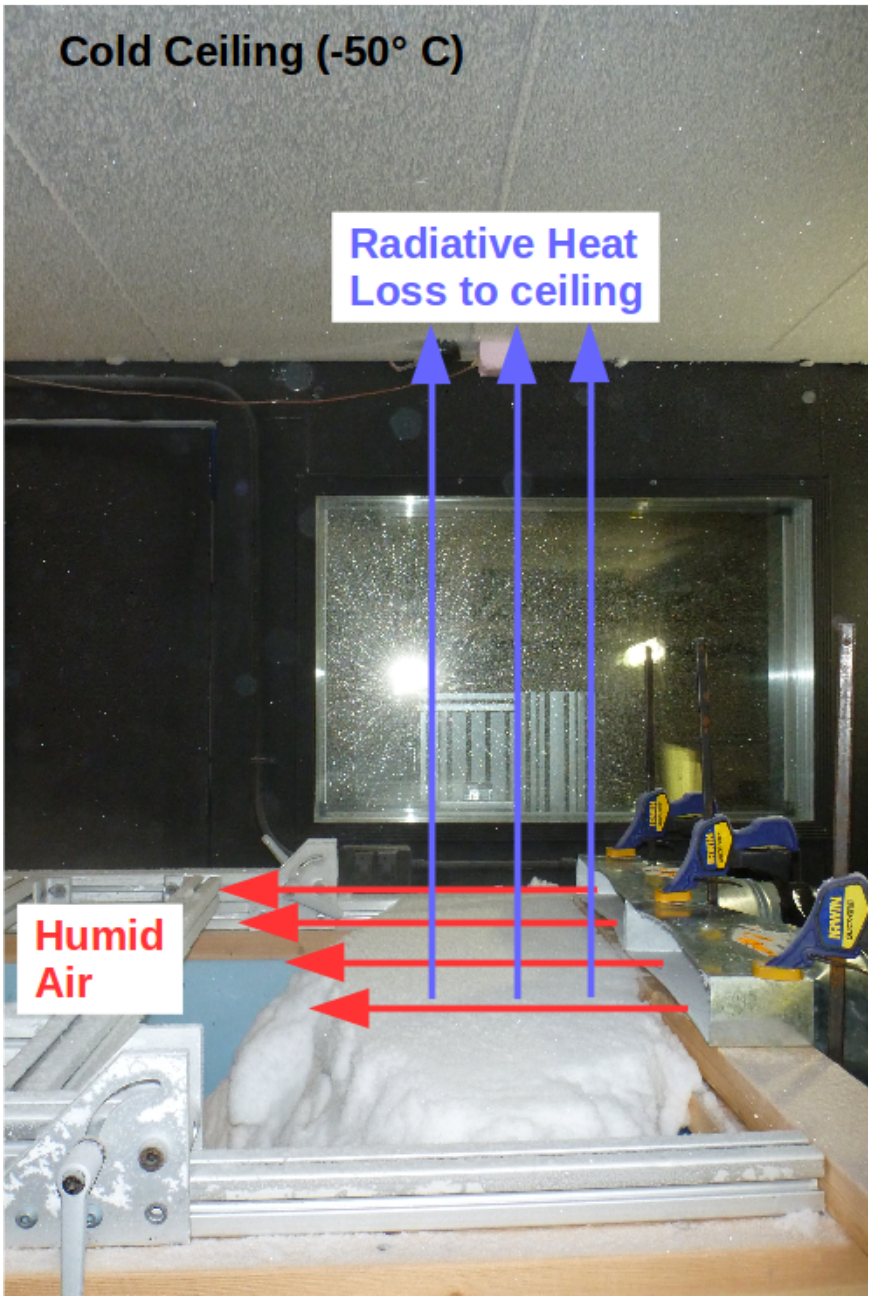


Figure 2.3: The laboratory surface hoar maker blows moisture-laden air gently across the snow surface. The snow surface emits longwave radiation to the cold ceiling.

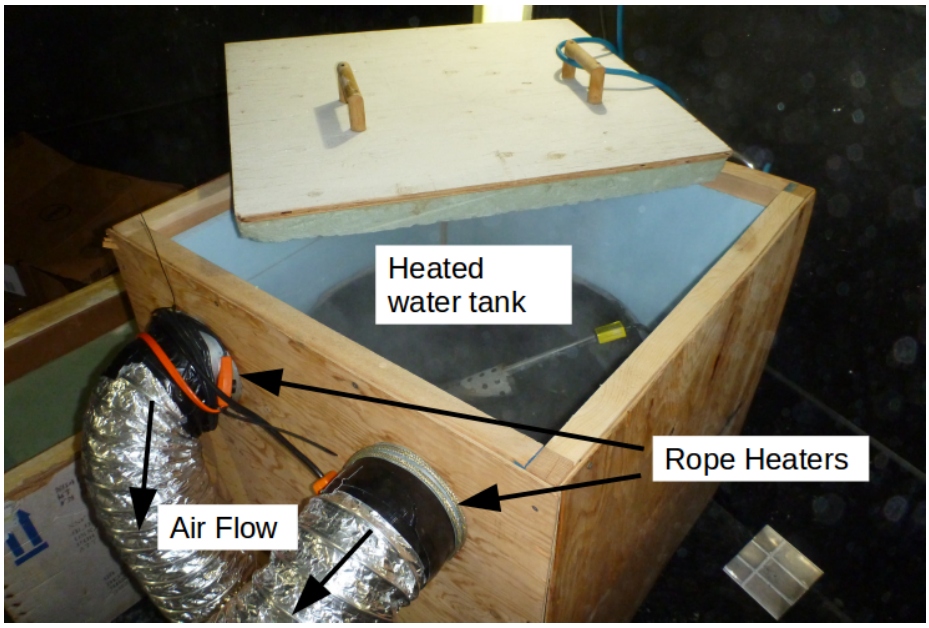


Figure 2.4: A reservoir of liquid water is kept at a constant temperature. Fans (indide ducts, not seen in photo) move warm, humid air out the ducts. The lid is closed during operation.

For homogeneous samples, snow was simply sifted into the base-cylidner assembly at the same time it was deposited for the sample. The assembly was then extraced and placed in the CT scanner.

For surface hoar samples, it was critical to extract the delicate surface hoar layer without damaging it. After surface hoar crystals finished growing, but before burial of the weak layer, the thin-walled cylinder was pressed into the snow. The cylinder penetrated the surface hoar layer and was pushed about $\approx 1cm$ into the hard base layer of snow. Damage to the weak layer was therefore minimized, as it was not necessary to disturb the slab to insert a sample container later in the process. The surface hoar was then buried. The CT sample was carefully extracted from the snow within 15 minutes of the mechanical test.

Analytical Method

From experimental results, shear modulus G, shear strength, and critical energy release rate G_c of samples were calculated for comparison with the model. Shear modulus and strength were calculated from stress-strain curves. Energy release rate was calculated by computing the cumulative amount of work done on the sample at fracture and then accounting for an estimated recovery. This methodology, based loosely on a common framework for quantifying damage in granular materials [50], has the advantage of computing work/energy directly, and is not limited by any assumptions about fracture mode or size of FPZ (i.e. brittleness of the fracture). The method assumes that some work done on the snow sample by the load actuator is stored as recoverable elastic energy in the ice network. Any work done that can not be recovered elastically is considered to be dissipated. Many mechanisms may contribute to dissipation, including frictional grain-grain contact, release of elastic strain energy as bonds break, or possibly other energy sinks.

Relevant Sample Geometry

To calculate shear strain, the height of the sample h_s had to be known. In homogeneous slabs, h_s was taken to be simply the height of the portion of the sample that is being sheared (Figure 2.2). However, in samples with buried weak layers, the weak layer will likely carry the bulk of the deformation. Here it is assumed that all deformation is in the weak layer, so that for strain calculations h_s is the height of the weak layer as measured in the laboratory. van Herwijnen et al [85], based on the work of Scapozza [69] report that elastic modulus E of snow is approximated by

$$E = 5.07 \times 10^9 \left(\frac{\rho_{snow}}{\rho_{ice}} \right)^{5.13} \tag{2.1}$$

In samples considered in this work, slab density was found to be 2-5 times weak layer density (See Chapter 8 for details). At a minimum, then, the slab would have been $2^{5.13} = 35$ times stiffer than the weak layer, and at most $5^{5.13} = 3850$ times stiffer than the weak layer. Thus, it was assumed that a negligible portion of deformation was in the slab. Therefore, strain in the weak layer could be computed from displacement of the load actuator and height of the weak layer. Strain in the slab was assumed zero for purposes of strain calculation.

Defining Failure

To ascertain when the specimen failed, the force-displacement curve of each experiment was analyzed. Experiments tended to show a series of events during loading. A linear elastic portion was always seen initially. Often, minor non-propagating damage events of briefly decreasing load (Fig. 2.5) were seen. Next, a period of decreased shear modulus was generally followed by peak stress. Stress then dropped quickly to a low level and stayed relatively stable for the remainder of the experiment.

Hill's instability criterion (1.9) is applied to these experiment. The specimen is failing when Equation 1.9 is satisfied. For these experiments in uniform simple shear, with shear force F and load actuator displacement x recorded at finite timesteps, the criterion becomes

$$\Delta F \cdot \Delta x < 0 \tag{2.2}$$

In these tests, samples are loaded at a constant 0.85 mm/s and sampled at a constant 1000 Hz. Therefore, Δx is constant for any sampled experimental interval, and the criterion for instability is simply $\Delta F < 0$. This criterion is manifested on the force-displacement curve by a negative slope, i.e. a decreasing load. However,

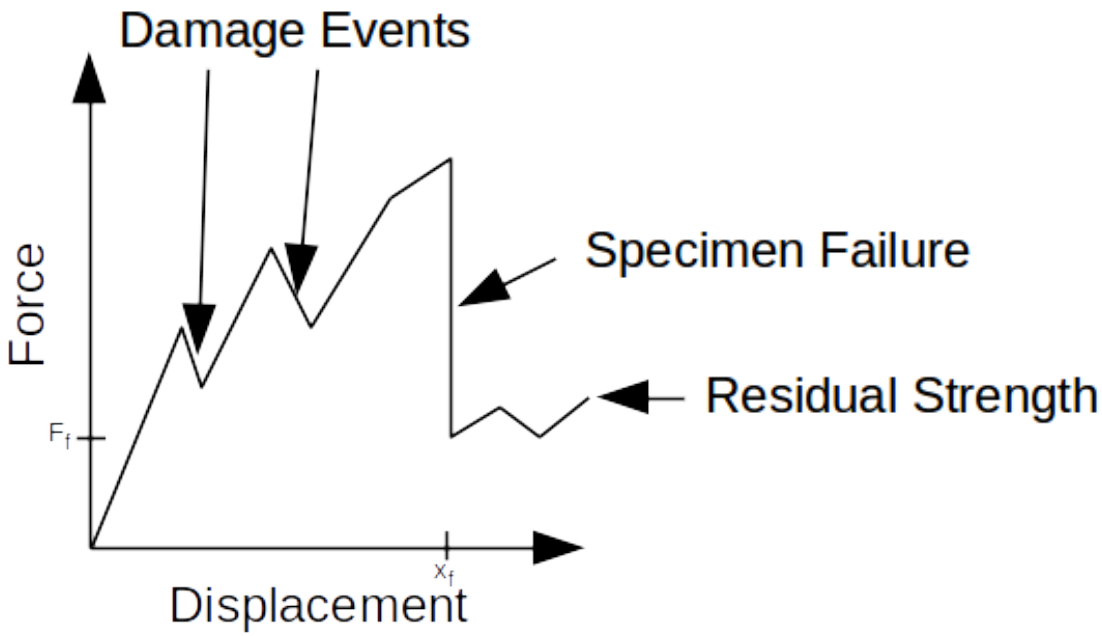


Figure 2.5: Illustration of generalized progression of experimental force-displacement curves.

there may be several regions in a force-displacement curve where $\Delta F < 0$ (Fig. 2.5). Failure then must be complete as soon as $\Delta F \geq 0$, indicating that the sample has finished failing and is back to a stable configuration.

A drawback of this method is that a given force displacement curve will almost always show several failures, and it is not always clear which region of $\Delta F < 0$ indicates global failure of the specimen. Consider the force-displacement curve obtained from experiment 20160809. Results showed three large drops in force during loading (Fig. 2.6). There are at least two plausible scenarios: the first large drop represents global failure, followed by collapse of the slab, followed by a re-interlocking of grains and subsequent force buildup. Alternatively, the first drop in force may simply be a large damage event, not global failure. Using the analysis of force-displacement curves alone, rigorous quantitative identification of global failure proves elusive.

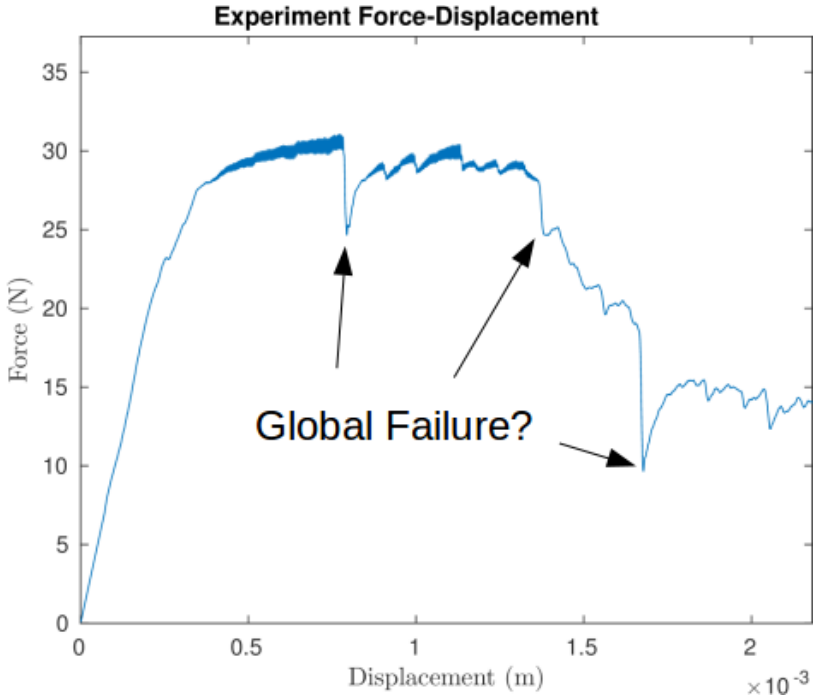


Figure 2.6: In experiment 20160809, there are three large drops in force. It is not clear which represents global failure.

Work Done on Sample

Although the point of global failure is not rigorously defined, cumulative work done on the sample by the load actuator must be computed. For experimental force-displacement curves, such as that shown in Figure 2.6, area was calculated using trapezoidal numerical integration.

Energy Release Rate

As the specimen is loaded, strain energy stored in the ice network is released. It is converted to surface energy, thermal energy, and kinetic energy. However, it is not necessary to explicitly account for all these terms. The amount of energy dissipated by fracture is simply the amount of work done on the sample minus the recoverable elastic energy remaining in the sample [50].

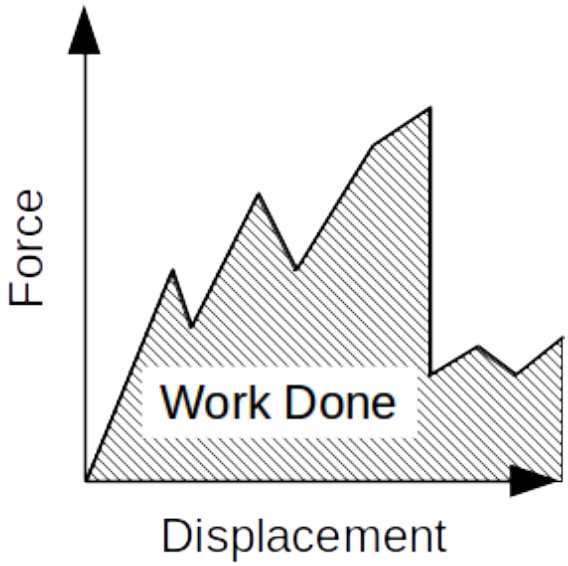


Figure 2.7: Work done on a sample of snow is simply the area under the force-displacement curve.

After failure of the snow specimen, some elastic strain energy (recoverable) likely remains in the ice network, but it is unclear how much. At one extreme, the sample could hypothetically recover fully. (Fig. 2.8a). A full recovery is taken to be an elastic rebound having the same slope as the initial linear portion of the force-displacement curve. The shear modulus G_{snow} of the sample is used to calculate the assumed maximum recovery. The initial slope of the Force-displacement curve is simply $G \cdot h_s$ where h_s is the height of the sample. In samples with surface hoar layers, h_s is the height of the surface hoar layer as measured in the laboratory. Residual force after failure is F_f and displacement at failure is x_f (Fig. 2.8).

Alternatively, there could be no elastic recovery, meaning that all the work done on the sample has been dissipated (Fig. 2.8b). Both these extremes represent improbable scenarios, and the actual amount of recovery must fall somewhere in between.

Critical energy release rate G_c of the sample falls in the range

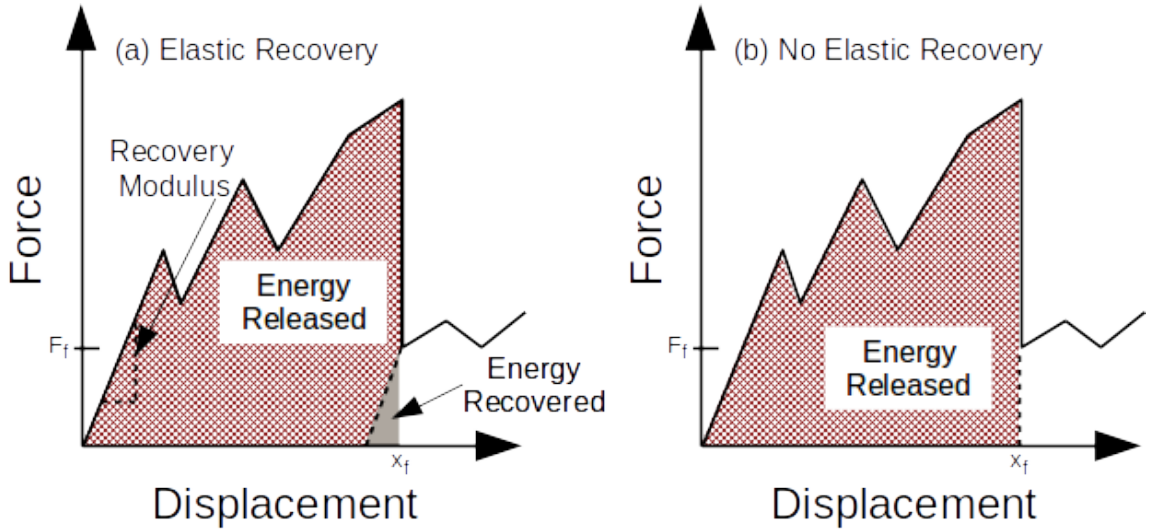


Figure 2.8: The two extreme values of energy release.

$$\frac{\int_0^{x_f} F dx - \frac{F_f^2 A_s}{2Gh_s}}{A_s} \leq G_c \leq \frac{\int_0^{x_f} F dx}{A_s} \tag{2.3}$$

where A_s is the cross-sectional area of the sample. $\frac{F_f^2 A_s}{2Gh_s}$ is the recoverable energy in the sample, represented by the area in gray in Figure 2.8a.

Experimental Results

Experimental results are summarized and explained here, and are compared to model results in Ch. 7. Energy release rate, elastic modulus, and peak strength are shown for all experiments in Table 2.1. In two experiments, energy release rate is not tabulated. This is due to some experiments not manifesting a sudden, obvious failure (e.g. Fig. 2.6). Complete stress-strain plots with comparison to model are shown in Appendix D.

Test Date	**Test Type**	**Strength**(Pa)	**Min** $G_c(J/m^2)$	**Max** $G_c(J/m^2)$
20160802	Homog	1496	1.06	1.22
20160803	Homog	1126	0.89	1.00
20160804	Homog	1263	0.64	0.74
20160809	Homog	1293	–	–
20160916	Homog	760	0.24	0.27
20160919	Homog	780	0.23	0.26
20161120	Homog	826	0.86	0.90
20170125	SH	1116	0.54	0.55
20170127	SH	810	0.89	0.89
20170131	SH	1029	0.31	0.32
20170224	SH	806	0.12	0.13
20170301	SH	1012	0.41	0.43
20170303	SH	1028	–	–
20170306	SH	1050	1.21	1.23

Table 2.1: Summary of all experimental Results

IMAGE PROCESSING

Overview

As quickly as possible after experiments (less than 15 minutes), snow was subjected to CT scans to capture 3D snow microstructure data. Specimens for CT scanning were taken from intact snow adjacent to the snow tested in shear. To process the CT scan data in a way suitable for numerical modeling, several steps were taken.

The purpose of this chapter is to explain in detail all techniques applied to capture and process CT scan data. CT scans provide three dimensional geometry of the ice grain network in a snow sample. This ice network geometry must be processed to glean the size and location of each grain and the orientation, size and location of every bond.

First, 3D geometry data is stored as a 3D binary array of voxels. In this 3D array, a value of 1 represents ice and a value of 0 represents air (pore space). The array is then subject to a watershed segmentation algorithm that determines which voxels are bonds and which voxels are grains (Fig. 3.1). Two arrays are then stored: one containing the voxels of every grain and one containing the voxels of every bond. The bond voxels are then analyzed to determine the area and orientation of every bond. The grain voxels are processed so that the shape and location of each grain is approximated in a way that is computationally efficient for DEM. Finally, grain and bond data are combined into a discrete element code.

Snow Microstructure Capture

To create representations of snow microstructures, details of the ice grain, bond and pore network geometry are needed. Actual snow samples were analyzed using

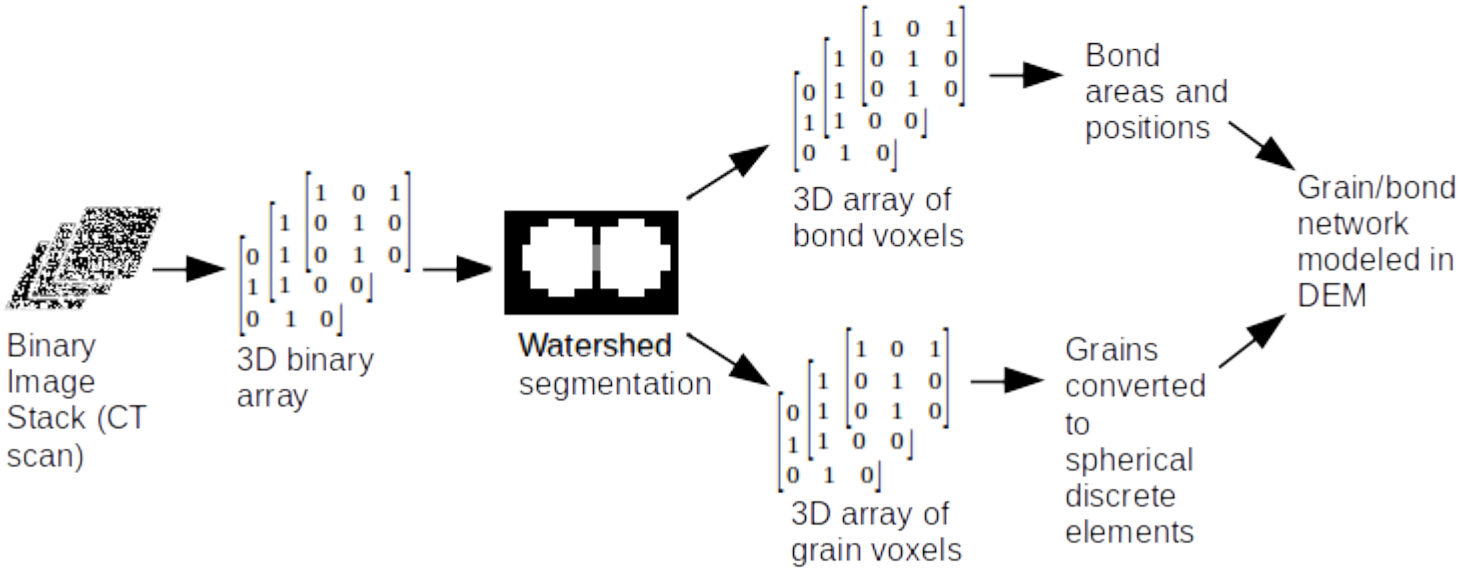


Figure 3.1: Overview of how CT scan data is discretized into an assembly of bonds and grains

a Skyscan 1173 x-ray micro-computed tomography (CT) scanner specially adapted for use in a cold lab. The scanner is capable of non-destructive 3D imaging of snow samples with resolution as fine as 5 μm.

In these tests, resolutions between 13 and 15 μm were used and cylindrical snow samples $25 \times 35\, mm$ were scanned (Figure 3.2). X-ray power was set to approximately 40 mV and 190 μA, although this varied from test to test. Exact power and exposure settings for each scan may be found in Appendix A.

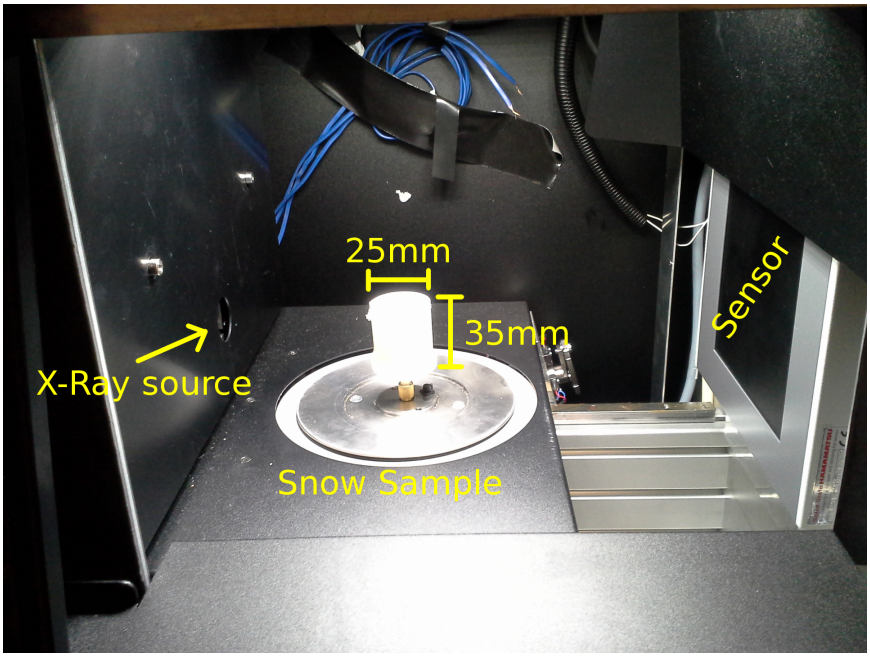


Figure 3.2: Cylindrical snow sample placed in the CT scanner, ready to be scanned. X-ray source is left, sensor on right.

The micro-CT scanner rotates the snow sample while capturing X-ray images every 0.7^o. X-rays are emitted from the source, pass through the sample, and are recorded with a digital sensor. The scan process results in a series images. The 3D sample was then reconstructed as a series of 8-bit grayscale image "slices" in the horizontal plane (Fig. 3.3). Since CT scan data was noisy, each slice was smoothed with a 7×7 pixel Gaussian filter [14].

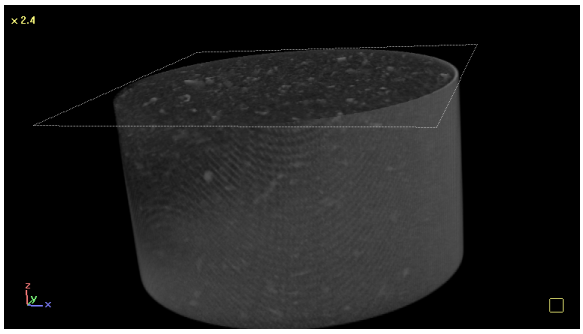


Figure 3.3: Reconstructed images lie in the horizontal plane, outlined by the white square.

These "slices" were then binarized to clearly demarcate which pixels represented ice and which pixels represented air (Fig. 3.4). A thresholding technique was used that assumes that pixels brighter than a certain value are ice and pixels darker than that value are air. This threshold was iteratively adjusted until the volume fraction of ice in the binary reconstruction matches the volume fraction of ice in the physical snow sample. Volume fraction was calculated from density of the sample. These images are imported into the MATLAB workspace and converted into a 3D binary array. In this array, a value of 1 denotes ice and a value of 0 denotes empty space.

Snow Grain Segmentation

To work with the snow as a network of bonds and grains, they must be reliably located and sized. In the original array, all voxels of ice are represented by a value of 1. This information is inadequate; it is necessary to know which voxels represent bonds and which represent grains. It is well accepted that bonds may be identified by looking for narrow "necks" of ice between larger grains. A neck-finding segmentation based on the well-established watershed transformation [21] technique is used. The segmentation algorithm works in two main steps:

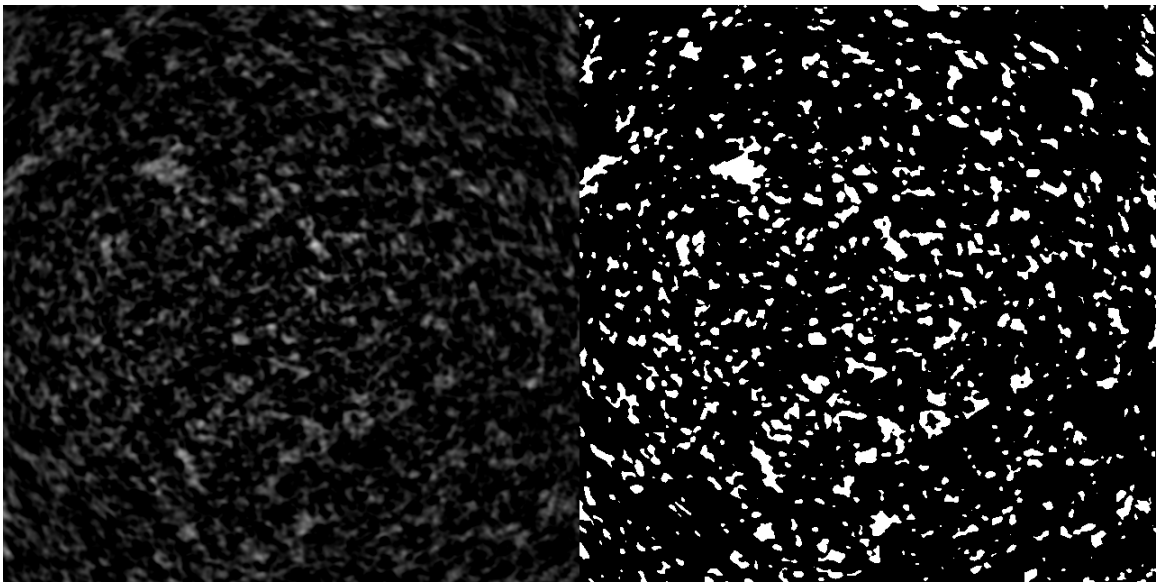


Figure 3.4: A reconstructed grayscale "slice" from the CT scan (left) is binarized (right)

1) Take the distance transform of the binary image.

2) Segment objects by locating ridges between catchment basins in the negative distance transform.

The technique will be explained with the simple example shown in Figure 3.5. Figure 3.5 is a pixelated binary image of two circular grains connected by a bond in the middle. White pixels represent ice and black pixels represent empty space. While this simple example is in 2 dimensions, these concepts are also applicable in 3D. The objective of the procedure is to develop a technique that will segment the image into two circular grains and identify the bond between them.

Distance Transform

First, a distance transform is applied to the image. The distance transform calculates the distance from any pixel in the object (white) to the closest pixel of empty space (black), and represents how close or far a pixel is from the edge of the

object. Specifically, the center-to-center pythagorean distance from each white pixel to the closest black pixel is calculated. A pixel at the edge of the object has a distance transform value of 1 because the distance from its center to the center of the closest black pixel is 1 pixel. Pixels at corners have distance transform values of 1.41 because the distance from the center of that pixel to the center of the closest black pixel is $\sqrt{1^2 + 1^2} \approx 1.41$. This procedure is repeated for each pixel.

Next, the negative of the distance transform is taken. Plotted in three dimensions, the negative distance transform (Fig. 3.6) visually suggests two catchment basins or watersheds separated by a ridge. The center of each circle has the most negative value because it has the greatest value in the distance transform.

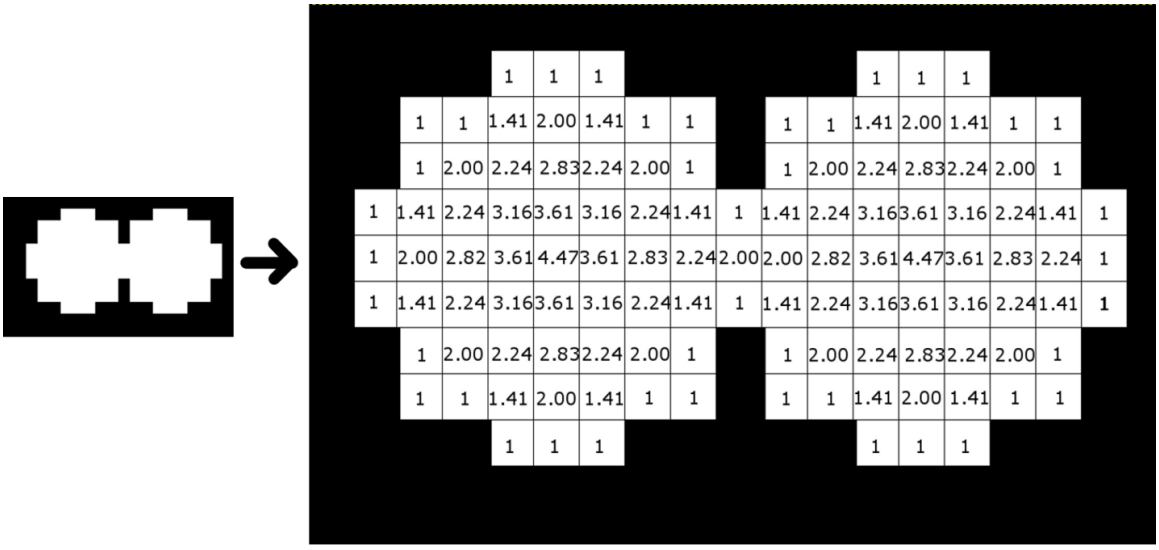


Figure 3.5: Two bonded grains (left) have the distance transform applied (right) to measure the distance in pixels from a pixel to the edge of the white area

Watershed Transform

While a visual inspection suggests the "ridge" and "watersheds" seen, a watershed transform algorithmically delineates the boundary between these two watersheds. MATLAB's watershed segmentation was used in this project and is based

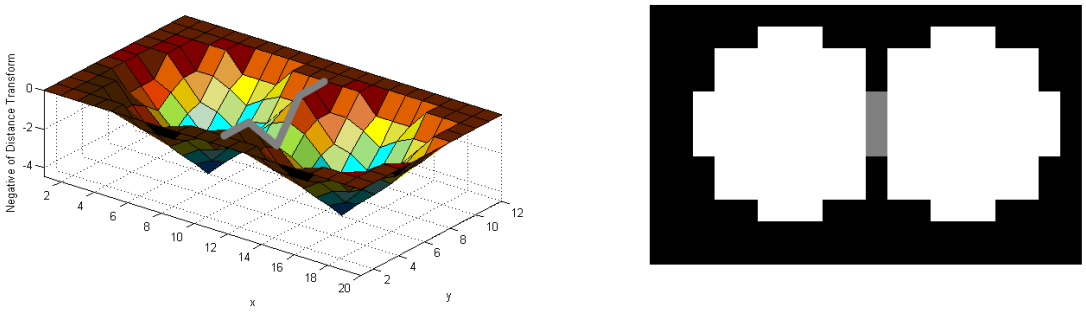


Figure 3.6: The negative of the distance transform (left) suggests two "basins" separated by a "ridge" in gray. In the original image, this "ridge" represents a bond separating two grains

on the Meyer algorithm [63] to find ridges and basins. The output of the watershed transform is a *label matrix*, where each voxel is assigned a number corresponding to its watershed. The label matrix L for the two circles in in Figures 3.5 and 3.6 would have two labeled grains:

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.1}$$

It does not matter which grain is labeled '1' and which is labeled '2,' only that each discrete grain has its own number. Also, note that the "ridge" (bond) between the two basins is not part of the label matrix. A separate array of bond voxels is exported as well.

H-Minima Transform

When using the watershed command, a sensitivity parameter was used to produce reasonable segmentations. If every single local minimum in the ice network's negative distance transformation is considered, the image becomes oversegmented. Oversegmentation creates too many grains and bonds, decreasing the computational efficiency of the model. Therefore, the ice network should be segmented into as few grains and bonds as possible while still creating mechanically realistic models.

Manipulation of the sensitivity of the watershed segmentation is accomplished by performing an H-minima transformation of the distance transform ([77],pp. 242-247). The H-minima transform first finds all local minima in an image. In the example of Figure 3.5, there are two local minima within the negative distance transform: one at the center of each grain. If these minima are shallower than a user-defined threshold h_{min}, they are suppressed.

As an example, Figure 3.5 represents the distance transformation of two overlapping circles. On the "ridge" between the two basins, there is a pixel with a distance transform value of 2 pixels. The center of each circle represents the deepest portion of the watershed, a local minimum, with distance transform values of 4.47. The local depth of each watershed is simply $4.47 - 2 = 2.27$.

If the H-minima transform is applied to the distance transform, minima below a certain threshold are suppressed. If an H-minima threshold of 2 is applied to the distance transform in Fig. 3.5, the watershed transformation is unaffected since the

local depth of each watershed is $2.27 > 2$. However, if an H-minima transform with $h_{min} = 3$ is applied to Fig. 3.5, all local minima less than 3 voxels deep are suppressed. Because local depths are $2.27 < 3$, a watershed segmentation of the h-min transform would show only one watershed (one grain).

The effect of the H-minima transformation on watershed segmentation on an actual ice grain network is seen visually in Fig. 3.7. The top left image is the original 3D binary image. The other three images are constructed from three label matrices using different values of h_{min}. Note that because these images are based on label matrices, the bond voxels between grains are not shown. Each discrete grain is plotted as a different color.

Figure 3.7 shows that when an $h_{min} = 0$, the microstructure is split into many grains (top right). Conversely, setting $h_{min} = 2$ creates no segmentation, also an undesirable result. A subjective visual inspection reveals that a threshold around 1 may be reasonable. However, this is a small, non-representative sample. Furthermore, human judgement is not reliable [2]. Therefore, the modeled mechanical response resulting from each of these segmentations was used to decide on a threshold and is discussed further in Chapter 6.

Bond Size and Orientation

After watershed segmentation, the area of each bond must be calculated. All bond voxels are stored in a 3D binary array the same size as the original 3D array. Voxels that represent grains between bonds are not included in this array. It is assumed that bonds are planar. A MATLAB routine considers the voxels comprising each bond and uses the least squares method to fit a plane to that collection of voxels. Then, a voxel projection technique previously developed for snow by Flin et al [19] is used to calculate the actual area of each bond. First, the normal $\vec{n}_{ls}$ of the least

squares plane fit is calculated. Then, to project each cubic voxel into a nonaxial plane, a ratio is calculated [19]:

$$ratio(\vec{n}_{ls}) = \frac{\|\vec{n}_{ls}\|}{\|\vec{n}_f\|} \tag{3.2}$$

where

$$\|\vec{n}_f\| = max(|n_x|, |n_y|, |n_z|) \tag{3.3}$$

This ratio effectively projects each voxel into either the x, y or z plane. Then, the projection is translated back into the original plane n_{ls}:

$$A_{est} = \sum_{n=1}^{n_v} ratio(\vec{n}_{ls}) \tag{3.4}$$

For code, see function *planefit.m* in Appendix A.

Approximation of Grains Using Multisphere Method

DEM uses spherical elements, but ice grains comprising the snow matrix are not neatly spherical. The multisphere method of Amberger et al [1] was used to take advantage of the computational efficiency of spherical elements while still offering a good representation of microstructure geometry. The multisphere method begins with a binary array of voxels representing a grain and replaces the voxels with a clump of overlapping spheres that approximates the geometry of that grain (Fig. **??**). Any spheres in a single clump may not move relative to one another, but the clump itself may move relative to its initial position. Therefore, forces and moments acting on a clump member are applied to the entire clump.

Let A_{lmn} be a 3D binary image array containing a single grain from a segmented CT scan. Ice voxels have a value of 1; empty space has a value of 0. Let D_{lmn} be

the distance transform of A_{lmn}. To begin the multisphere approximation, a sphere is inscribed centered at the voxel with the largest distance transform value. This sphere's radius is equal to the value of the distance transform. Thus, the newly inscribed sphere will fill the object to the grain boundary. Voxels that fall within the sphere are erased. The erasing procedure for a sphere centered at (i, j, k) with radius r_{ijk} is to examine all voxels $A(l, m, n)$ in the array and determine whether they fall within the sphere. If

$$\sqrt{(l-i)^2 + (m-j)^2 + (n-k)^2} < r_{ijk} * r_p \tag{3.5}$$

then the value of $A(l, m, n)$ is set to 0. The term coefficient r_p is introduced to allow the spheres to overlap by some amount. In this model, $r_p = 0.8$ was subjectively found to be reasonable.

A distance transform is performed on this newly modified $A(l, m, n)$ array and a new sphere is inscribed at the voxel with the largest distance transform value. As this process is repeated, the volume of the grain is filled with spheres. The process is stopped after a prescribed number of spheres. The best number of spheres per grain to use is discussed in Chapter 6.

Fine-tuning Volume of Approximation As spheres are inscribed within the grain volume, the volume of the multisphere approximation $V(A)$ will approach, but never equal, the volume $V(O)$ of the original shape. Amberger et al [1] solve this problem by adjusting sphere radii and sphere centers using a scaling factor:

$$s = \sqrt[3]{V(O)/V(A)} \tag{3.6}$$

Then, s is used to adjust radii and center locations of all spheres within that clump. For sphere i, radius radius r_i and center c_i are adjusted:

$$r_i \rightarrow sr_i \tag{3.7}$$

$$c_i \rightarrow (1-s)m + sc_i \tag{3.8}$$

where m is the center of mass of the original grain. Equations 3.6-3.8 ensure that the multisphere approximation of a grain will have the same volume as the original grain.

Description of Code

Image processing is done through a wrapper script *yadebuild.m* that runs several subroutines to accomplish all the tasks described in this chapter. The output from *yadebuild.m* is an executable yade script *yadeout.py*. Each subsample to be tested through the DEM must have its own *yadeout.py* file. The flow chart in Figure 3.9 summarizes the relationships and outputs of all functions. Full .m files are found in Appendix A.

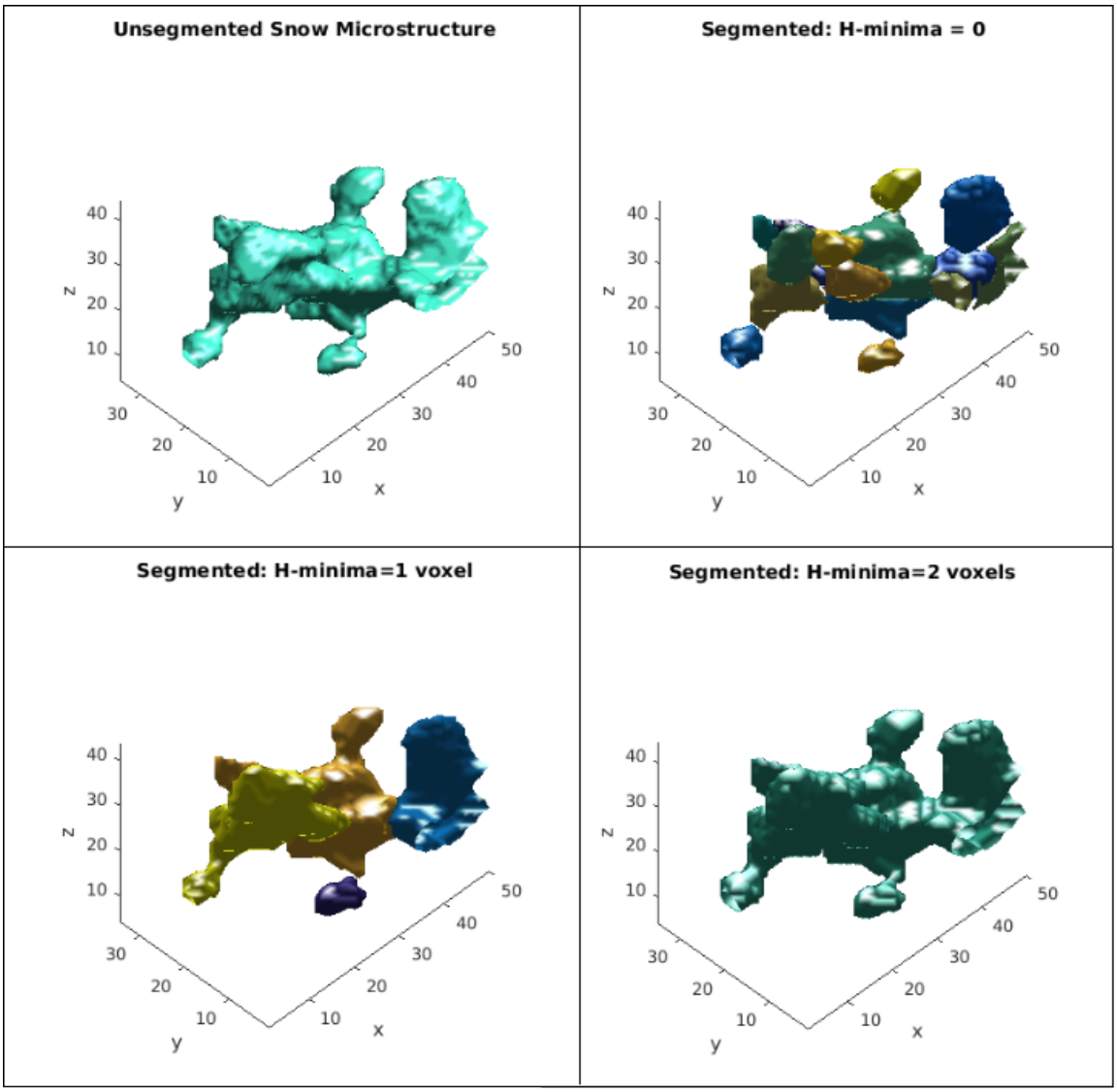
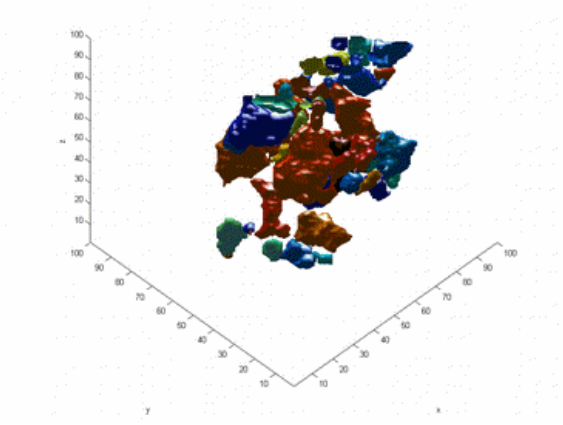


Figure 3.7: The initial ice network microstructure (top left) is segmented using various H-minimum thresholds. Different colors represent separate grains

[Original grains represented as voxels after watershed segmentation]



[Grains approximated as clumps of spheres]

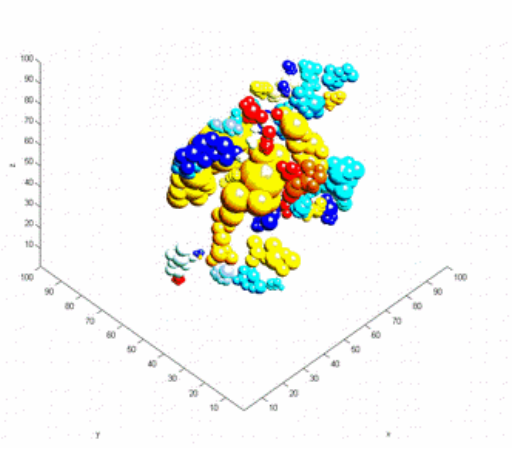


Figure 3.8: Illustration of multisphere method

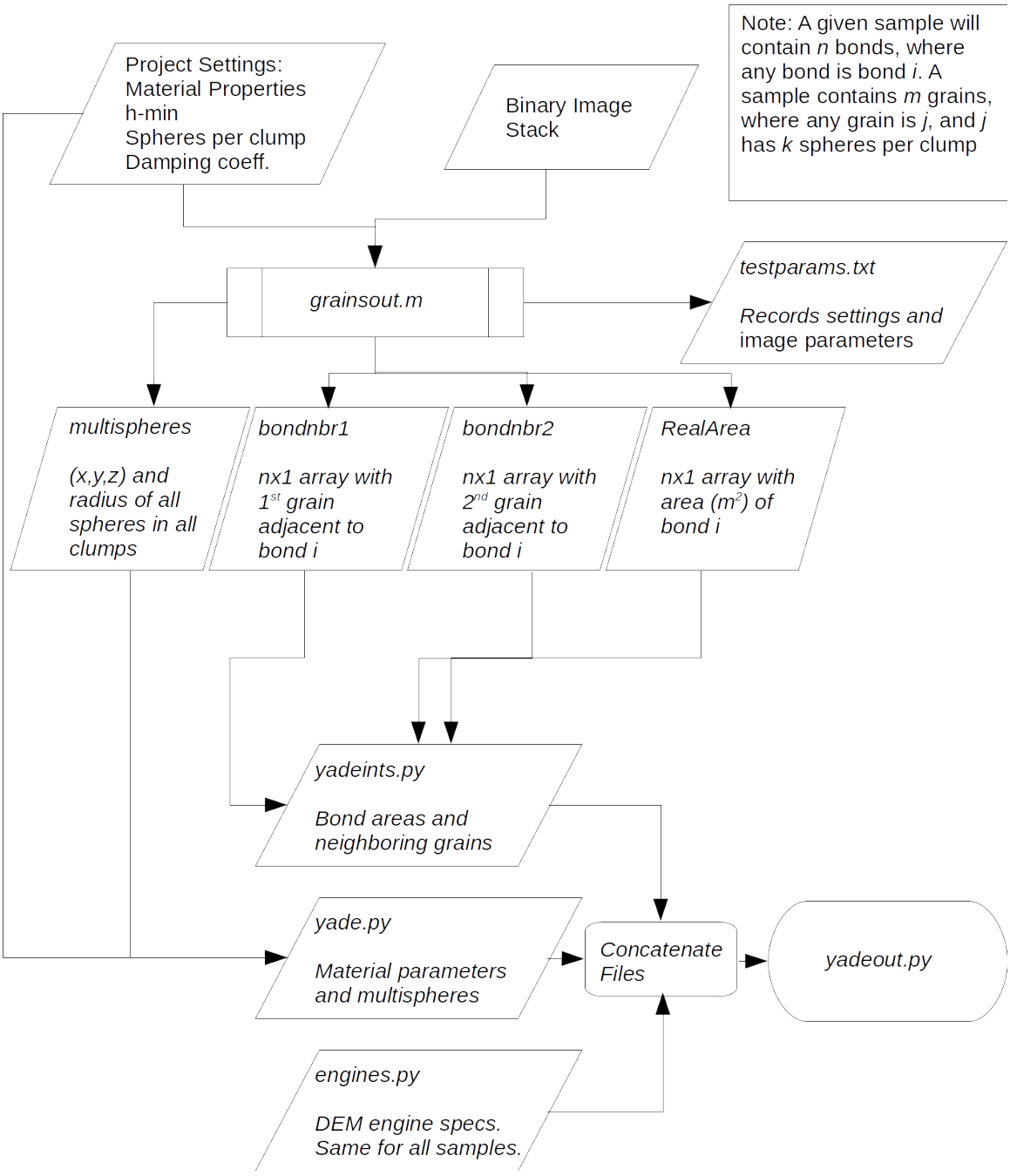


Figure 3.9: Key variables and processes used to process binary CT images into executable DEM simulation *yadeout.py*.

MINIMUM FRACTURE ENERGY

"For every complex problem there is an answer that is clear, simple and wrong"
-H.L. Mencken

Theory

Snow is often analyzed using linear elastic fracture mechanics (LEFM) of continuum materials (e.g. [34], [75]). In LEFM, the energy release rate of a material is minimized. In other words, the energy involved in fracture is only that required to break bonds between particles or between atoms along one fracture path. Other energy sinks, such as plastic deformation, microcracking, friction, movement of particles, and inter-particle damping are not considered. The purpose of this analysis is to determine whether fracture in snow follows the minimum energy path.

Griffiths energy criterion for fracture states that the minimum amount of energy that can be released during fracture of a material is

$$G_c = 2\gamma \tag{4.1}$$

where γ is the energy required to create a new surface of unit area of the material (coefficient 2 denotes two new surfaces). It has been suggested that fracture in granular materials propagates across the grain boundary path that requires the least energy to create (e.g. [6, 38, 79]).

In snow, the minimum energy fracture path is the set of bonds that, if broken, would minimize the energy cost to break the sample into two pieces. Because the energy expended in fracture is a function of G_c and bond area, minimizing the area of broken bonds also computes the minimum energy path.

By finding the *minimum energy path*, minimum fracture energy release can be compared to actual fracture energy release to assess the validity of the LEFM assumption.

A hypothetical structure of circular ice grains (white) connected by bonds (red) is shown in Figure 4.1 (Note that this simple example is in two dimensions, but the concepts are equally applicable in 3D). To find the minimum energy fracture path, the minimum amount of energy needed to break the sample into two separate pieces must be found. Since the amount of energy required to fail a unit area (G_c) is roughly constant, it follows that minimizing the total area of broken bonds will minimize the total dissipation of energy during fracture.

Application of Graph Theory to Snow Microstructure

Since snow microstructure is a network of bonds and grains, it can be represented as a graph. This is accomplished by converting the microstructure of a sample into a graph of nodes and edges for analysis as shown in figure 4.1. Each grain becomes a node, and each bond becomes an edge between nodes. Each line is assigned a weight equal to the minimum energy required to break it:

$$W_f = 2\gamma A_i \tag{4.2}$$

where W_f is the line weight and A_i is the cross sectional area of bond i. Then, finding the minimum energy fracture surface is a matter of finding the minimum sum of W_f values that will break the sample into two discrete pieces, thereby creating a complete fracture surface [39].

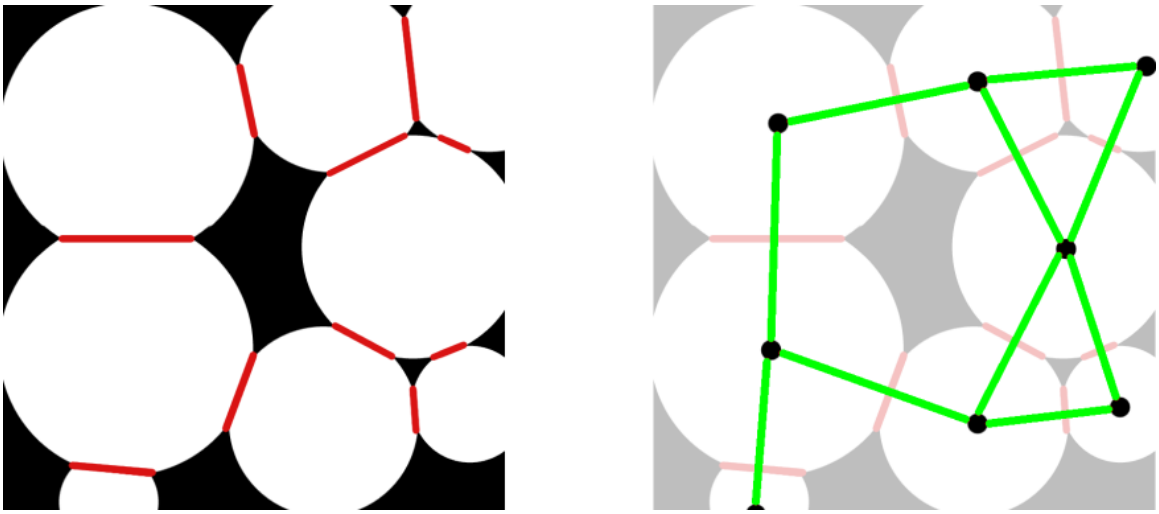


Figure 4.1: A cross-section of a hypothetical granular microstructure (left) is converted into a node/line graph representing potential fracture paths (right).

The minimum energy surface is found by applying max-flow min-cut theorem (See Chapter 1). This concept will be applied to the network in Figure 4.1 as an example.

Imagine that the specimen is loaded so that a fracture will separate the top of the sample from the bottom of the sample. It may be loaded vertically in tension or horizontally in shear or some combination thereof; this model does not address loading in finding the minimum energy path. Each real bond is assigned a weight proportional to the energy required to break it (Figure 4.2). Note that in this hypothetical scenario, weights are for illustrative purposes only, but larger bonds have higher weights (higher energy cost to break). By imagining that each weight is a flow capacity, Figure 4.2 shows a valid maximum flow path with the flow through each line and the capacity of each line. Then, by the max-flow min-cut theorem, the minimum energy fracture path (drawn in magenta) has an energy cost of 17 units.

To ensure that a cut separates the top of the sample from the bottom of the sample, an additional top node and bottom node are required. These nodes are added

to the graph and connected by vertices with infinite capacity. The infinite capacity is analagous to an infinite energy cost to break. Therefore, these newly added edges in the graph can not be part of the minimum cut.

Although this example is in two dimensions for illustrative simplicity, this method is also applicable to a 3D graph network [39].

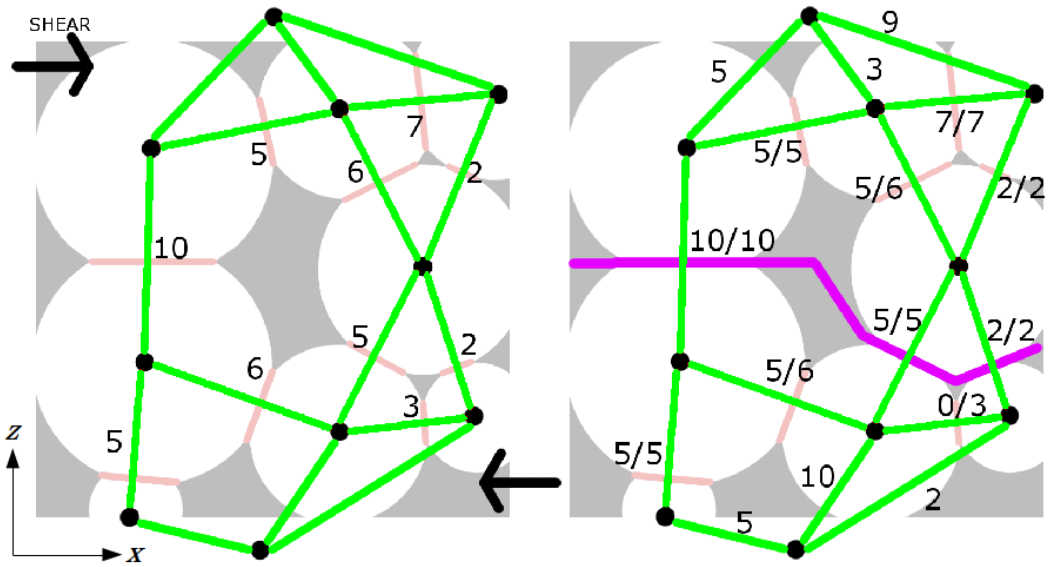


Figure 4.2: Max-flow/min-cut theorem applied to the sample geometry from figure 4.1. Lines with infinite capacity were added to the top and bottom of the sample in order to create top and bottom master nodes that must be separated by the cut.

Ford and Fulkerson (1962) [20] provided the first algorithm to compute maximum flow through a graph. To begin, any allowed flow is forced through a network. The algorithm then finds any available path that will increase total flow, called an augmenting path, and adds the augmenting path to the flow. This process is repeated iteratively until there is no valid increase in flow. Edmonds and Karp (1972) [18] refined the algorithm to make it more computationally efficient by choosing the augmenting path with the fewest edges each time.

Algorithm

Graph Creation

First, grains and bonds of a snow sample are demarcated using the watershed segmentation as specified in Ch. 3. The following outputs from the watershed segmentation are used to assemble the graph:

-Size of each bond

-The two grains neighboring each bond

From the image processing routine, two 3D arrays are output: *bonds* and *labelgrn*. *bonds* is simply a 3D binary array that contains all bond voxels in the sample. All bond voxels have a value of 1. Array *labelgrn* is a 3D array where each discrete grain is labeled. The voxels representing grain 1 have a value of 1, the voxels representing grain 2 have a value of 2, and so on. The matrices in Figure 4.3 represent a simple example of *bond* and labelgrn arrays. Grains 2 and 1 are shown in *labelgrn*, and the bond between them is shown in array *bonds*

$$bonds = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad labelgrn = \begin{bmatrix} 0 & 2 & 2 & 2 & 2 & 0 \\ 0 & 2 & 2 & 2 & 2 & 0 \\ 0 & 2 & 2 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Figure 4.3: Example *bonds* and *labelgrn* arrays for simple geometry. While the example is in 2D, the concepts apply to 3D arrays as well.

An important caveat must be addressed here. To find the minimum cut, the grain-bond network is imagined to have a flow associated with it so the maximum flow can be computed (See Ch. 1). In numerical tests, coordinates are arranged so that both tension and shear are expected to create a fracture that separates grains

at the top of the sample (high z) from grains at the bottom of the sample (low z). Therefore, the simulated flow must go in the z direction throughout the model in order to bifurcate the sample along the z-plane, so that the minimum cut surface will separate grains with high z from grains with low z. The flow directions assigned in the graph could go in either the positive or z, and the cut result would be the same. It was arbitrarily decided that the flow direction is from + to - z. Setting the flow from - to + z would yield the same max flow/min cut result.

The function *GraphBuild2.m* determines which two grains neighbor each bond, and the "flow" direction through each bond: which grain is the "from" grain and which grain is the "to" grain. The grain with a higher z coordinate is the "from" grain, and the grain with a lower z coordinate is the "to" grain.

A pseudocode is presented in Algorithm 4.1, and complete code is in Appendix B. *GraphBuild2* does use the *PlaneFit* function, found in Appendix A.

Algorithm 4.1: Pseudocode for Graphbuild2.m

Input: 3D arrays *bonds* and *labelgrn*

```
from=[] %Create empty arrays
to=[]
area=[]
for all bonds i
  for all voxel coordinates j within bond i
    nbrgrains=unique values of grainlabel in neighborhood of j
  end
  from(i)=ID number of grain with higher z centroid
  to(i)=ID number of grain with lower z centroid
  Area(i)=PlaneFit(Bond i) %Outputs area of bond in voxels
end

for all i
  G(From(i),To(i))=Area(i)
end
```

GraphBuild2 generates graph G representing the bond-grain network, with each edge weighted by the size of the bond plane it represents. For any bond i with area A_i that connects grains j and k, let

$$G(j,k) = A_i \tag{4.3}$$

Using the example of Figure 4.3, the bond has an area of 2 voxels and it connects bonds 1 and 2. Because flow was assigned to go from high z to low z, the graph must allow flow from 2 to 1 and not from 1 to 2. Therefore, the graph G would have only one nonzero value: $G(2,1) = 2$.

Minimum Energy Calculation

Once graph G is assembled, minimum cut energy is calculated using the Edmunds-Karp algorithm [18] in the freely available, open-source Matlab Boost Graph Library (MatlabBGL) [26]. An example of a snow sample analyzed for minimum cut is shown in Figure 4.4. The cut separates grains at the top (shown in gold) from grains at the bottom (blue) by the minimum cut energy.

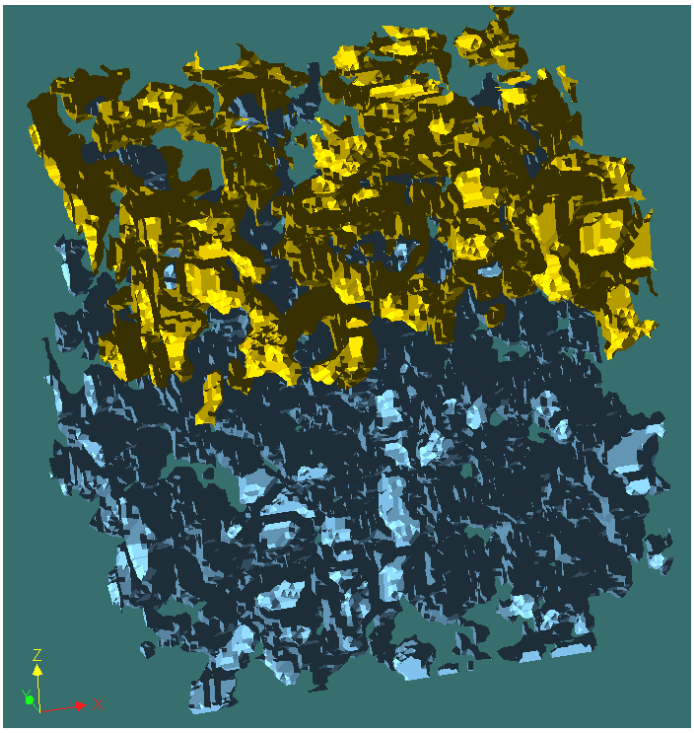


Figure 4.4: The end result of the minimum energy cut. The top of the sample (gold) is separated from the bottom (blue) via the path requiring the least surface formation energy.

DISCRETE ELEMENT MODEL OVERVIEW

Purpose and Objectives

A microstructural model was desired that would realistically simulate the fracture process. In addition to failure of bonds, a model must also track phenomena like grain/bond motion and inter-particle collision. Discrete element modeling (DEM) is uniquely appropriate for simulating fracture in granular materials. Originally conceived to model granular materials [17], DEM is a dynamic modeling technique that tracks particle interaction and movement, as well as cohesive particle-particle bonds. DEM has been widely applied in ceramics (e.g. [43, 44, 52]), geomaterials [67] and even snow [31].

Finite Element Analysis (FEA) has also been used to model fracture in snow. However, FEA does not generally consider dynamic effects or particle interaction. Most FEA modeling in snow has been macroscopic [12, 54, 55] where snow is approximated as a continuum.

Therefore, a discrete element model (DEM) was used to simulate snow fracture experiments and explicitly track energy sources and sinks during the fracture process. DEM analyses were performed on models representing CT scans of real snow. Bond-grain networks representing actual snow were obtained using the image processing techniques described in Chapter 3. The model was built in the free open-source DEM software YADE (Yet Another Dynamic Engine) [76]. The objective was to begin with realistic snow microstructure (bond and grain) information, apply loading to the microstructure, and compute energy terms associated with the loading and failure process. The microstructural model precisely records load, elastic strain energy, energy release, kinetic energy and frictional loss throughout the duration of the test.

Because the DEM models every bond and grain in snow microstructure, an extremely detailed analysis is possible. It is possible to tell which bonds have ruptured, where those bonds are located, and how much energy release is associated with each bond. DEM can also compute exactly how much elastic strain energy is stored in the system at any time and how much kinetic energy is present, giving a detailed representation of energy sinks within the sample.

Forces acting on the elements (grains) are dictated by inter-elemental forces defined by constitutive laws. While undergoing mechanical testing, the energy into the system is simply the work done by a load actuator. During the mechanical loading process, it was assumed that mechanical work done on the sample could be converted into either of the following forms of energy:

1. Elastic strain energy stored in the deformed ice matrix
2. Kinetic energy due to movement of grains

Furthermore, elastic and kinetic energy in the system may be dissipated in any of the following ways as the sample fails:

1. Strain energy conversion to surface energy as new surfaces form
2. Dissipation of thermal energy due to frictional grain-grain contact

A complex interaction of these processes dictates the energy release observed during the fracture process. It is the objective of this model to quantify the amount of energy stored in the system and dissipated at all times during failure.

General DEM Background

DEM is a dynamic modeling technique based on contact forces and associated displacements between discrete particles (elements). Generally, elements are spherical

[17], although some models employ computationally more expensive cylindrical or polyhedral elements. By Newton's second law, the sum of forces $\vec{F}$ acting on any particle i at position $\vec{r}_i$ are given by [53]:

$$\sum \vec{F} = m_i \frac{d^2\vec{r}_i}{dt^2} \tag{5.1}$$

Similarly, for moments about point O acting on a particle,

$$\sum \vec{M}_O = \frac{d\vec{H}_O}{dt} \tag{5.2}$$

where $\vec{H}_O$ is the angular momentum of the particle about point O. Differential equations 5.1 and 5.2 are solved numerically over a discrete timestep $\Delta t \approx dt$ to determine acceleration, velocity and displacement of the particle at any time t in the simulation. $\vec{F}$ and $\vec{M}$ acting on the particle may be dictated by any number of constitutive laws describing contact between two particles. $\vec{F}$ and $\vec{M}$ may be set by the user as boundary conditions. In any model, an adequate number of constitutive equations and boundary conditions must be known to solve equations 5.1 and 5.2.

In general, a pair of spherical elements may have up to six degrees of freedom (DOF) [87]: 1 normal, 2 shearing, 1 twisting, and 2 bending (Figure 5.1.

Geometric Variables

Several terms related to sphere-sphere interactions must be defined. For any two interacting spheres [76], undeformed sphere radii are r_1 and r_2 (not the same as the vector $\vec{r}$ in Equation 5.1 above) and sphere centers are $\vec{C}_1$ and $\vec{C}_2$. Since they define an undeformed configuration, r_1 and r_2 are constant. $\vec{C}_1$ and $\vec{C}_2$ will change as spheres move. A normal axis n connects sphere center to sphere center, and shear axes t_1 and t_2 are orthogonal to n (Figure 5.2).

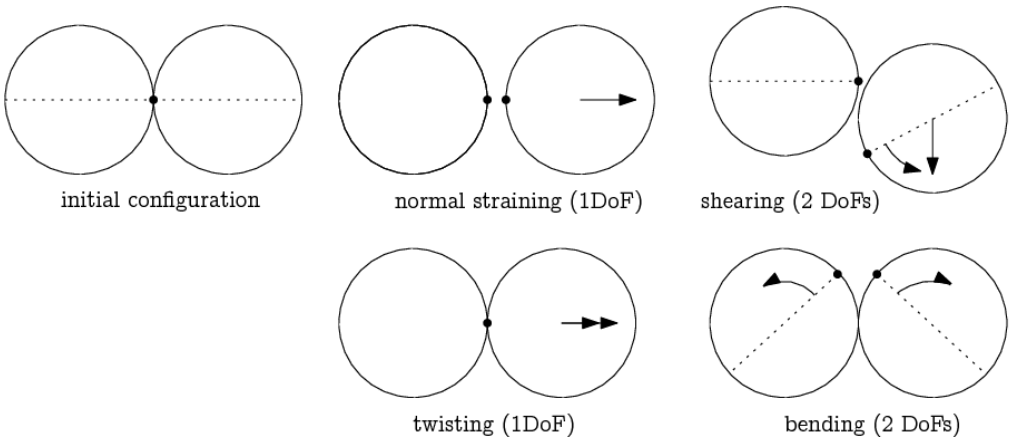


Figure 5.1: Six possible degrees of freedom for deformations between spheres. From [87]

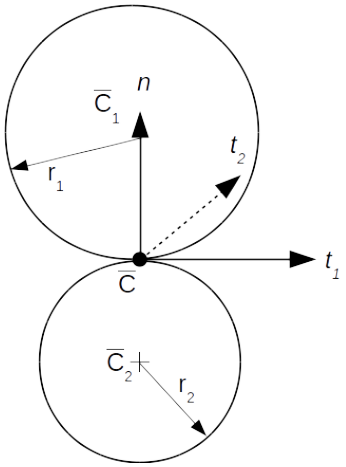


Figure 5.2: Two undeformed spherical DEM elements, labeled with relevant geometric variables

As spheres interact, they may penetrate each others' surfaces and deform elastically in the normal direction. The center of interaction is $\vec{C}$, and d_1 and d_2 are deformed radii that may be equal to, greater than or less than r_1 and r_2 (Figure 5.3. The quantity $r_1 + r_2 = d_1 + d_2$ is the equilibrium configuration [87]. During normal deformation, there is generally some resistance to penetration. Constitutive relationships are discussed in the next section.

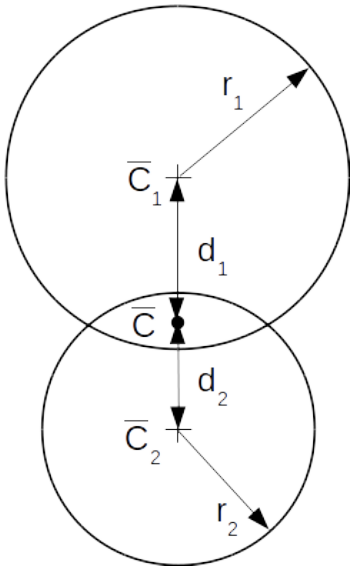


Figure 5.3: Two spherical DEM elements deform in the normal direction, introducing deformed radii d_1 and d_2

$$d_0 = |\vec{C_1} - \vec{C_2}| \tag{5.3}$$

Then, normal relative displacement between sphere centers is

$$\delta_n = d_0 - (r_1 + r_2) \tag{5.4}$$

Shear displacement δ_t may also occur. Shear deformation considers a combination of sphere center displacements and rotational motion. Consider two spheres in

an initial configuration, denoted by superscript $^-$ and a current configuration denoted by superscript ^{o}

$$\vec{\delta}_t^{\circ} = \vec{\delta}_t^{-} - \vec{\delta}_t^{-} \times (\vec{n}^{-} \times \vec{n}^{\circ}) - \vec{\delta}_t^{-} \times \left\{ \frac{\Delta t}{2} \vec{n^{\circ}} \cdot (\omega_1^{\ominus} + \omega_2^{\ominus}) \right\} n^{\circ} - \Delta t \vec{v_{12}^{\perp}} \tag{5.5}$$

where $\vec{v}_{12}^{\perp}$ is the component of the velocity of sphere center 1 relative to sphere center 2 perpendicular to n° (shear velocity), $\omega_1^{\ominus}$ and $\omega_2^{\ominus}$ are angular velocities of spheres 1 and 2, with $\omega^{\ominus}$ the mean angular velocity of previous and current timesteps. For more details on the derivation of Eq. 5.5, see [76].

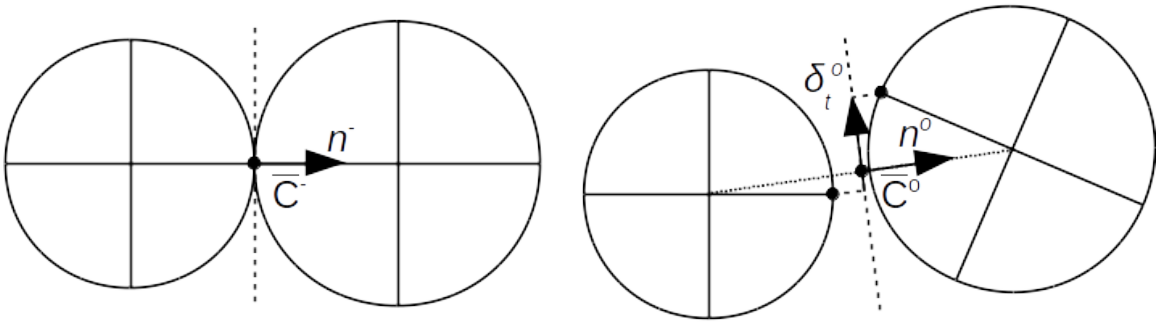


Figure 5.4: Two spheres in the previous timestep (left) and current timestep (right). After [87]

Contact Constitutive Laws

The laws of motion and computations of displacements were outlined above. Here, constitutive relations that dictate those motions are given. Cohesive contacts (analogous to bonds between ice grains) are modeled as a linear spring acting between two spheres. The force that a contact applies to a sphere depends on its stiffness. A contact may have normal stiffness k_n (N/m) in the direction of n, shear stiffness

k_t (N/m) along axes t_1 and t_2, bending stiffness k_r (Nm/rad) about t_1 and t_2, and twisting stiffness k_{tw} (Nm/rad) about n. All the constitutive laws here are from [76].

For the two-sphere pair in Figure 5.3, the normal force in a linear elastic model is given by:

$$F_n = k_n \delta_n \tag{5.6}$$

For the special case of spheres 1 and 2 in frictional, non-cohesive contact,

$$k_n = \frac{E_1 r_1 E_2 r_2}{E_1 r_1 + E_2 r_2} \tag{5.7}$$

where E_1 and E_2 are elastic moduli of spheres 1 and 2, respectively. For bonded grains, a constitutive relationship is derived later.

Similarly, shear force is:

$$\vec{F_t} = k_t \vec{\delta_t} \tag{5.8}$$

Bending moment is:

$$\vec{M_r} = k_r \vec{\Theta_r} \tag{5.9}$$

Where Θ_r is relative angular displacement between the two spheres. It is determined by the non-twisting component of relative angular velocity ω_{12}:

$$\vec{\Theta}_r^{\circ} = \vec{\Theta}_r^{-} + \Delta t(\vec{\omega}_{12} - (\vec{n}^{\circ} \cdot \vec{\omega}_{12})\vec{n}^{\circ}) \tag{5.10}$$

For twisting moment,

$$\vec{M_{tw}} = k_{tw} \vec{\Theta_{tw}} \tag{5.11}$$

with twist angle in the current timestep $\vec{\Theta}^{\circ}_{tw}$ given by

$$\vec{\Theta}^{\circ}_{tw} = \vec{\Theta}^{-}_{tw} + \Delta t(\vec{n}^{\circ} \cdot \vec{\omega}_{12})\vec{n}^{\circ}) \tag{5.12}$$

Formulations of k_n, k_s, k_r and k_{tw} specific to this bonded-particle model are derived later in this chapter.

Timestep

Since discrete element is a dynamic method, a discrete timestep $\Delta t \approx dt$ must be defined for use in Eqs. 5.1 and 5.2. A large Δt is computationally cheaper but may render inaccurate results. Conversely, a small Δt is more computationally expensive (requiring more timesteps to run the model). A critical minimum timestep is established [76]:

$$\Delta t_{cr} = \frac{2}{\omega_{max}} \tag{5.13}$$

where ω_{max} is the natural angular frequency of the spring-mass system. For a large system, ω_{max} is the highest frequency in the system. For a spring-sphere pair [76],

$$\Delta t_{cr} = \sqrt{2}\sqrt{\frac{m_i}{k_i}} \tag{5.14}$$

where k_i is the highest value of k for a contact on the sphere (i.e. k_n, k_s,k_r,k_{tw})

Failure of Contacts

For a bond between two spheres, failure may be reached according to a constitutive law or failure envelope. Available failure criteria vary widely between

different DEM codes. In this model, a failure criterion is specifically defined in Section 5.

Frictional Contacts

Although cohesive contacts are assigned in the undeformed sample, deformation of the sample may rearrange grains in such a way that they contact one another during failure. Spheres that come into contact in this way interact using a frictional elastic contact law, usually a Mohr-Coulomb envelope defined with a coefficient of friction.

DEM Formulation for Bonded Particles

This section explains the exact constitutive laws, stiffness and strength formulations, and geometry used in the model for snow. A fracture mechanics energy-based constitutive approach is presented. Linear models for normal force (Eq. 5.6), shear force (Eq. 5.8), twisting moment (Eq. 5.11), and bending moment (Eq. 5.9) in bonds are used. Values of k_n, k_s, k_r and k_{tw} are derived in detail here. Shear and normal strength based on application of Griffith's criteria are also derived in this section.

Normal Stiffness Normal stiffness is dictated by the simple linear constitutive relation given in Eq. 5.6. This section will derive a value k_n for a bonded sphere model. This derivation follows the general procedure of Johnson [45] for solids of revolution based on Hertzian [35] mechanics, but is carried out for the specific case where both particles are spheres of the same material. Two undeformed spheres that are just touching are assigned a right-handed local coordinate system. The x and y axes are in a plane tangent to the contact. Any displacement of sphere centers occurs

in the z direction in this section to calculate normal stiffness. Sphere 1 has radius r_1, and sphere 2 has radius r_2 (Fig. 5.5).

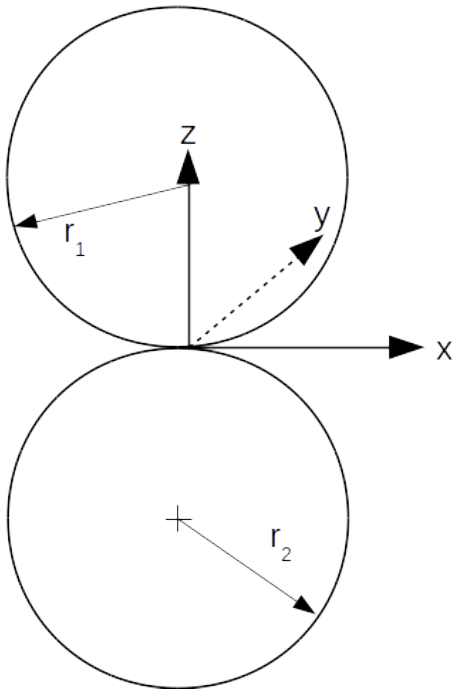


Figure 5.5: Local coordinate system and radii of two spheres.

Near the contact, Johnson [45] approximates the surface of sphere 1 as

$$z_1 = \frac{1}{2r_1}x^2 + \frac{1}{2r_1}y^2 \tag{5.15}$$

Similarly, for sphere 2,

$$z_2 = -\frac{1}{2r_2}x^2 - \frac{1}{2r_2}y^2 \tag{5.16}$$

Note that these equations are only valid approximations for values $(x, y) \ll (r_1, r_2)$. Defining the separation between surfaces $h(x, y) = z_1 - z_2$ and combining Eqs. 5.15 and 5.16,

$$h(x,y) = \frac{1}{2}\left(\frac{1}{r_1} + \frac{1}{r_2}\right)(x^2 + y^2) \tag{5.17}$$

For notational convenience, define $R^2 = x^2 + y^2$ and $\frac{1}{r} = \frac{1}{r_1} + \frac{1}{r_2}$ so that Eq. 5.17 becomes

$$h = \frac{1}{2r}R^2 \tag{5.18}$$

Now, the spheres are pressed together (Figure 5.6). The center of sphere 1 moves a distance δ_1 towards the contact and the center of sphere 2 moves a distance δ_2 toward the contact. As this happens, the surfaces of the spheres near the contact deform. The contact is assumed to be a circle in the x-y plane with radius a. The z direction deformation at any R of sphere i is defined as $u_{zi}(R)$.

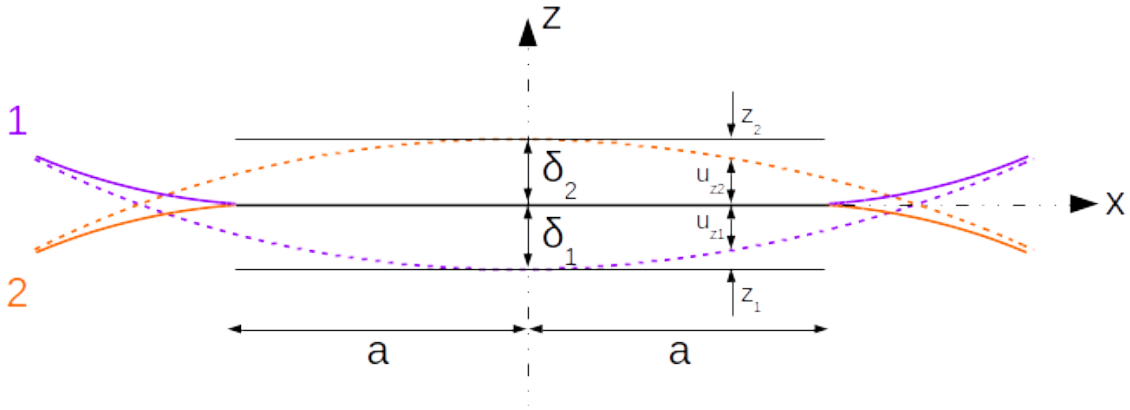


Figure 5.6: Close-up at the contact between two deformed spheres. Solid lines are sphere surfaces. Dashed lines indicate undeformed surface that would occur if there were no resistance to penetration. Black line represents contact.

From Figure 5.6, it can be seen that for any $R \leq a$, the spheres are in contact. Surface deformation of the spheres is given by

$$u_{z1} + u_{z2} + z_1 - z_2 = \delta_1 + \delta_2 \tag{5.19}$$

Defining $\delta = \delta_1 + \delta_2$ and substituting in Eq. 5.18,

$$u_{z1} + u_{z2} = \delta - \frac{1}{2r}R^2 \tag{5.20}$$

Then, Hertz's findings [35] for normal stress distribution on a sphere are applied:

$$\sigma = \sigma_0 \left[1 - \left(\frac{R}{a}\right)^2\right]^{\frac{1}{2}} \tag{5.21}$$

where σ_0 is the maximum stress, found at $R = 0$. Eq. 5.21 also shows that $\sigma = 0$ at $R = a$, the edge of the contact. Hertz [35] also showed that for one sphere subjected to the pressure distribution in 5.21,

$$u_z = \frac{1-\nu^2}{E}\frac{\pi\sigma_0}{4a}(2a^2 - R^2),\ R \leq a \tag{5.22}$$

where E and ν are elastic modulus and poisson's ratio of the material, in this case ice. Then, Eq. 5.22 is applied to Eq. 5.20. For notational convenience, let

$$\frac{1}{E^*} = 2\frac{1-\nu^2}{E} \tag{5.23}$$

and get

$$\frac{\pi\sigma_0}{4E^*a}(2a^2 - R^2) = \delta - \frac{1}{2r}R^2 \tag{5.24}$$

Eq. 5.24 is valid at all R. By setting $R = 0$, a simple expression for δ as a function of peak stress appears:

$$\delta = \frac{\pi a\sigma_0}{2E^*} \tag{5.25}$$

For DEM, it is useful to have these quantities as functions of applied force P, not peak stress σ_0. The stress distribution given in Eq. 5.21 is integrated over the area of the bond and combined with Eq. 5.25, yielding the surprisingly simple relation

$$P = (E^* a)\delta \tag{5.26}$$

Then,

$$k_n = E^* a \tag{5.27}$$

Shear Stiffness The derivation of shear stiffness is based on the spheres in Figs. 5.6 and 5.5 and again, summarized from Johnson's book [45].

Tangential traction (in the x-y plane in Fig. 5.5) across the surface of two contacting spheres is distributed as [35]

$$q_x(R) = q_0 \left(1 - \frac{R^2}{a^2}\right)^{-0.5} \tag{5.28}$$

where, for a shear force Q_x distributed over the contact area, average surface traction is

$$q_0 = \frac{Q_x}{2\pi a^2} \tag{5.29}$$

Shear displacement of the center of an elastic sphere is [45]

$$u_x = \frac{\pi(2-\nu)}{4G} q_0 a \tag{5.30}$$

where G is shear modulus. For two spheres with contact radius a, relative displacement between sphere centers is

$$\delta_x = \frac{2-\nu_1}{4G_1}\frac{Q_x}{2a} + \frac{2-\nu_2}{4G_2}\frac{Q_x}{2a} \tag{5.31}$$

which, for two particles of the same material, simplifies to

$$\delta_x = \frac{(2-\nu)Q_x}{4Ga} \tag{5.32}$$

Since E for polycrystalline ice is already defined in the model, it is convenient to substitute the well-known isotropic relationship (See, for example, [73]):

$$G = \frac{E}{2(1+\nu)} \tag{5.33}$$

yielding

$$\delta_x = \frac{(2-\nu)(1+\nu)Q_x}{2aE} \tag{5.34}$$

Rearranging Eq. 5.34 and isolating the stiffness term as defined in Eq. 5.8:

$$k_t = \frac{2aE}{(2-\nu)(1+\nu)} \tag{5.35}$$

For notational convenience, let $E^{**} = \frac{2E}{(2-\nu)(1+\nu)}$ so that Eq. 5.35 becomes

$$k_t = aE^{**} \tag{5.36}$$

Bond Strength Criteria

Normal Strength To quantify the normal strength P_c of a bond, the energy associated with crack extension is considered. P_c is the normal (tensile) force required to initiate critical crack propagation in the bond. The edge of the bond is considered

the crack tip, and the infinitesimal extension of the crack is da (Fig. 5.7) with initial bond radius a.

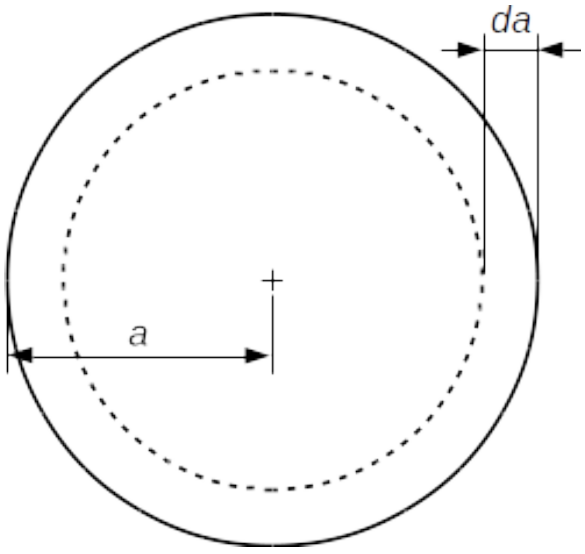


Figure 5.7: Cross section of a circular bond during failure under normal loading. As the crack propagates, bond radius changes by da.

During crack propagation, strain energy must be converted into surface energy at the critical energy release rate G_c [78]:

$$G_c = \frac{dU_w}{dA} - \frac{dU}{dA} \tag{5.37}$$

where U_w is work done by the force P and U is stored elastic energy. A is the cross-sectional area of the bond. To compute the maximum force that the bond can withstand at the onset of fracture, $\frac{dU_w}{dA}$ and $\frac{dU}{dA}$ are considered separately. First, consider the work done on the sphere-sphere pair by a constant force P over displacement δ:

$$U_w = \int P d\delta \tag{5.38}$$

Then, applying Eq. 5.26,

$$\frac{dU_w}{dA} = \frac{dU_w}{d\delta}\frac{d\delta}{dA} = P\frac{d\left(\frac{P}{E^*a}\right)}{dA} \tag{5.39}$$

$$\frac{dU_w}{dA} = \frac{dU_w}{du}\frac{du}{dA} = P\frac{d\left(\frac{P}{E^*a}\right)}{dA} \tag{5.40}$$

Taking P and E^* to be constant, and setting $dA = -2\pi a(da)$:

$$\frac{dU_w}{dA} = \frac{P^2}{2\pi}\frac{d(1/a)}{da} = \frac{P^2}{2\pi E^* a^3} \tag{5.41}$$

Now, working on the second term of Eq. 5.37, the stored elastic energy in the bond prior to breakage is:

$$U = \frac{1}{2}P\delta = \frac{P^2}{2E^*a} \tag{5.42}$$

During crack extension, again holding N and E^* constant and setting $dA = -2\pi a(da)$,

$$\frac{dU}{dA} = \frac{P^2}{2E^*}\frac{d(1/a)}{da} = \frac{P^2}{4\pi E^* a^3} \tag{5.43}$$

Combining Eqs. 5.37,5.41, and 5.43 and defining P_c as the force at failure:

$$G_c = \frac{P_c^2}{4\pi E^* a^3} \tag{5.44}$$

which is the solution presented in [43] for the small contact approximation. Eq. 5.44 is then rearranged to solve for the normal strength of the bond:

$$P_c = 2\sqrt{G_c \pi E^* a^3} \tag{5.45}$$

Shear Strength A similar exercise is performed for shear strength. But for shear fracture, $dA = -2a \cdot 2da$ as the crack is presumed to initiate at the center and propagate outward [43].

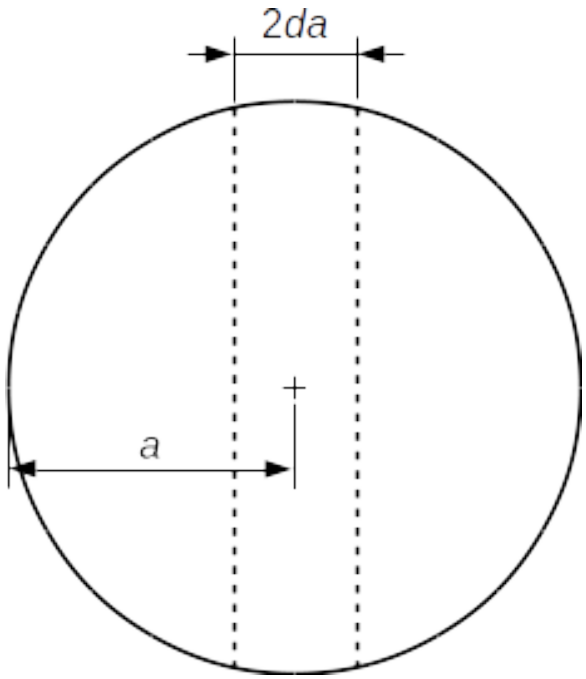


Figure 5.8: Cross section of a circular bond during failure initiation under shear loading.

Then, the process for calculating strength is again carried out, but with shear terms rather than normal terms. Q replaces P, E^{**} replaces E^*, and $dA = -4ada$. The work term from Eq. 5.37 is

$$\frac{dU_w}{dA} = \frac{dU_w}{d\delta}\frac{d\delta}{dA} = Q\frac{d\left(\frac{Q}{E^{**}a}\right)}{-4ada} = \frac{Q^2}{4E^{**}a^3} \tag{5.46}$$

And the stored elastic energy term:

$$\frac{dU}{dA} = \frac{d(Q^2/(2E^{**}a))}{-4ada} = \frac{Q^2}{8E^{**}a^3} \tag{5.47}$$

Applying Eqs. 5.46 and 5.47 to Eq. 5.37 yields

$$Q_c = \sqrt{8 G_c E^{**} a^3} \tag{5.48}$$

The relations above were developed for sintered spheres with circular bonds, specifically ceramics (e.g. [43], [**?**]).

Modified Criteria for Snow

In the snow model, the watershed analysis computes a bond area, not a bond radius. Bonds are not necessarily circular, although it is common in snow to assume such (e.g. [64]). To fit the constitutive model, bond i with area A_i is approximated as a circular bond of the same area with radius a_i:

$$\pi a_i^2 = A_i \tag{5.49}$$

When combined, Eqs. 5.45 and 5.49 yield

$$P_c = 2\sqrt{G_c \pi E^*} \left(\frac{A_i}{\pi} \right)^{0.75} \tag{5.50}$$

Similarly, combining Eqs. 5.48 and 5.49 yields

$$Q_c = \sqrt{8 G_c E^{**}} \left(\frac{A_i}{\pi} \right)^{0.75} \tag{5.51}$$

Eqs. 5.50 and 5.51 are used directly in the model as bond strength criteria.

Bending and Twisting Stiffness

Bending and twisting stiffness add rigidity to the model. Without them, a pair of bonded spheres may not transmit moments. Then, bonded grains roll and twist freely relative to one another. So, even though bending and twisting moments do

not influence failure within the model, they are important to create a realistic elastic response in the ice matrix. A popular formulation [42, 43, 67], based on the response of a circular beam with radius a,

$$k_r = \frac{1}{4}E^* a^3 \tag{5.52}$$

$$k_{tw} = \frac{1}{2}E^{**} a^3 \tag{5.53}$$

Again, Eq. 5.49 is used to approximate the bond as a circle. Eqs. 5.54 and 5.55 become, respectively,

$$k_r = \frac{1}{4}E^* \left(\frac{A}{\pi}\right)^{1.5} \tag{5.54}$$

$$k_{tw} = \frac{1}{2}E^{**} \left(\frac{A}{\pi}\right)^{1.5} \tag{5.55}$$

Bonds may not fail due to moments. Bonds may only break when shear force exceeds shear strength or when tensile force exceeds normal strength.

Approximating Grains as Clumps

Inertial Properties of Clumps Snow grain geometry is approximated by using clumps of overlapping spheres. Because the spheres overlap, computing mass and mass moments of inertia of the clump is not a simple matter of adding mass or mass moments of constituent spheres. Rather, a fine grid is passed over the clump, and the mass of each grid cell is used to calculate the inertial properties. [76]

Forces and Torques acting on clumps

Clump members are treated as stand-alone particles for the purposes of contact detection. In other words, contacts may only exist between spheres, not between clumps. By definition, spheres within one clump are not assigned contacts with one another. Any external force or moment acting on a clump must act on exactly one member (sphere) of the clump. Then, during motion integration, all external forces F_i and couples M_i acting on all spheres i of a clump c are applied to the clump [76]:

$$\vec{F_c} = \sum \vec{F_i} \quad and \quad \vec{M_c} = \sum \vec{r_i} \times \vec{F_i} + \vec{M_i} \tag{5.56}$$

Contact Placement As discussed above, contact is permitted between spheres, not clumps. Any contact force or moment acting on a sphere in the model is then transferred to the clump through the contact. However, in real snow, contacts exist between grains. The question arises, then, of which two spheres should carry the contact between two clumps. For two clumps l and m, the two closest spheres in l and m were assigned a contact with properties of the real bond gleaned from the CT scan. Two arbitrary multisphere approximations are illustrated in Fig. 5.9. The contact is shown in red, placed where the distance between any sphere in l and any sphere in m is minimized.

This bond placement is accomplished using the function *findclosphere.m* (Appendix A). Using nested for loops, *findclosphere.m* loops through all possible $l - m$ sphere combinations until it finds the pair of spheres with the least distance between them.

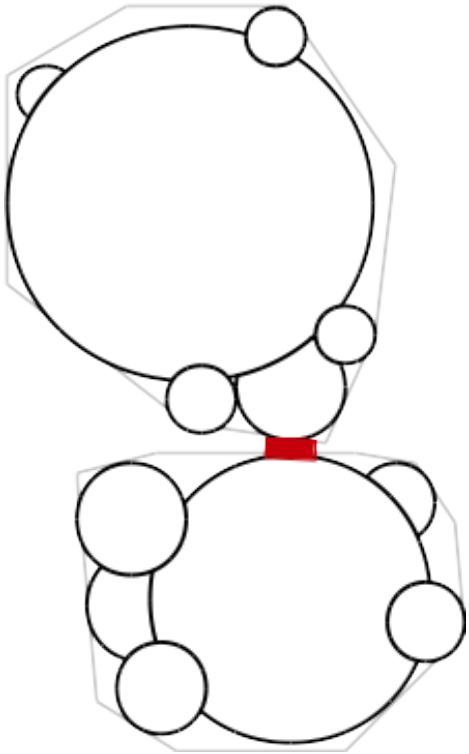


Figure 5.9: Grain-grain contact is placed between the two closest spheres in neighboring grains.

Frictional Contact

As grains are rearranged, new contacts may be created. These contacts are frictional rather than cohesive. They follow a Mohr-Coulomb friction criterion in the tangential (shear) direction and a simple spring in the radial (normal) direction [76].

$$P = k_n \vec{\delta}_n \tag{5.57}$$

$$Q^T = k_t \vec{\delta}_t \tag{5.58}$$

where $\vec{\delta}_n$ and $\vec{\delta}_t$ are normal and shear displacements, respectively. Force P must be compressive; frictional contacts do not support tension. Q^T is a test force to be compared against a maximum friction criterion.

For two spheres of radius r_1 and r_2 and elastic moduli E_1 and E_2, Normal stiffness is [87]:

$$k_n = 2\frac{E_1 r_1 E_2 r_2}{E_1 r_1 + E_2 r_2} \tag{5.59}$$

For a model with only one material, $E_1 = E_2 = E_{ice}$ and Eq. 5.59 simplifies to

$$k_n = 2\frac{E_{ice} r_1 r_2}{r_1 + r_2} \tag{5.60}$$

Frictional (shear) forces are defined using a friction angle ϕ and contact stiffness K_s.

$$k_s = K_n \frac{G_{ice}}{E_{ice}} \tag{5.61}$$

Then, the friction force criterion is

$$F_s = \begin{cases} |\vec{F}_n| \tan\phi, & \text{if } |\vec{F}_s^T| > |\vec{F}_n| \tan\phi \\ \vec{F}_s^T & \text{otherwise} \end{cases} \tag{5.62}$$

Frictional contacts can not transmit bending or twisting moments.

Damping

Bonds are represented in this model as undamped springs. This can lead to high force oscillations in a sample during fracture. Because fracture tests are meant to be quasi-static, a non-physical numerical damping scheme was used to calm particles

experiencing rapidly increasing velocities, leading to more stable test results. The non-physical damping scheme uses a sign function to decrease forces that increase the speed of a particle [87]. For a force $\vec{F}_w$ acting on an element at a given timestep, the force is adjusted by the quantity $\Delta\vec{F}_{dw}$:

$$\frac{(\Delta F)_{dw}}{F_w} \approx -\lambda_d sgn(F_w u_w), w \in (x, y, z) \tag{5.63}$$

where λ_d is a dimensionless user-defined damping coefficient. The minus sign on the right side of the equation ensures that a particle with increasing speed will have a damping force applied to it. For more details on the damping scheme, see [87].

Boundary Conditions

Clumps at the base of a sample were fixed in all six DOF: 3 DOF in translation and 3 DOF in rotation. Clumps at the top of a sample were assigned a constant 0.001 m/s velocity in the x-direction. These clumps were free to rotate about X, Y, Z axes and free to translate in Y, Z directions.

This velocity represents an extremely high strain rate of 14

Execution in YADE

The MATLAB image processing code (See Ch. 3 and Appendix B) writes a file called *yadeout.py* that executes in YADE. Each subsample from each test has its own *yadeout.py* file that builds the ice grain/bond network in YADE and executes the loading procedure. YADE code is divided into 6 sections:

1. Define test parameter variables:

   (a) Shear velocity (m/s)

(b) Ice material constitutive properties

(c) Damping Coefficient

(d) Data save interval (seconds)

2. Add clumps to simulation. For each clump (grain):

(a) Define center and radius of each sphere in clump

3. Define Physics Engines to use in simulation:

(a) Define collision detection engine

(b) Define physics engine

(c) Apply numerical damping coefficient

(d) Every 1000 timesteps, call routine AutoData that records simulation data:

i. Force

ii. Displacement

iii. Simulation Time

iv. # of cohesive bonds remaining in simulation

v. Total elastic strain energy in model

vi. CPU Time

(e) Every 20 minutes, save the state of the entire simulation.

4. For all bonds:

(a) Define sphere-sphere pair the bond is applied to

(b) Specify brittle-elastic bond model (no plasticity)

(c) Specify Bond Stiffness (k_n, k_s, k_r, k_{tw})

(d) Specify Bond Strength

5. Apply Boundary Conditions

   (a) Determine which clumps are at the top of the sample

   (b) Apply shear velocity to all of these clumps

   (c) Determine which clumps are at the base of the sample

   (d) Apply fixed boundary condition to all of these clumps

6. Run Simulation

For an annotated sample copy of *yadeout.py*, see appendix C.

DEM PARAMETRIC ANALYSIS

Due to the large number of parameters defined throughout the image processing and modeling steps of the project, sensitivity analyses were carried out to determine what effect several different parameters had on the overall behavior of the model, and to determine ideal values of these parameters.

Sensitivity to H-Min Parameter

The watershed segmentation depth threshold H_{min} (explained in Ch. 3) was adjusted to ascertain its effect on mechanical properties and energy dissipation in snow samples. The value of H_{min} affects segmentation, and by extension the mechanical response of the sample. The purpose of this sensitivity analysis is to decide on a useful value of H_{min} for modeling purposes. A lower H_{min} value will increase the number of bonds and grains segmented by the algorithm, while a higher value will decrease that number (Fig. 3.7). For computational purposes, it is preferable to use a higher value of H_{min} since models with fewer elements will run faster. However, a value of H_{min} that is too high will not create a mechanically realistic bond/grain network. Therefore, the objective was to do as little segmentation as possible while creating a realistic model.

This sensitivity analysis was run on a 2x2x2 mm cube of homogeneous rounded grains from sample 20160802 with 20 spheres per clump. H_{min} values of 0, 0.5, 1, 1.5, 2, 2.5 and 3 were considered. Visualizations of the multispheres in the YADE environment are given in Figure 6.1. Figure 6.2 shows how many bonds and grains were counted with each different H_{min} value.

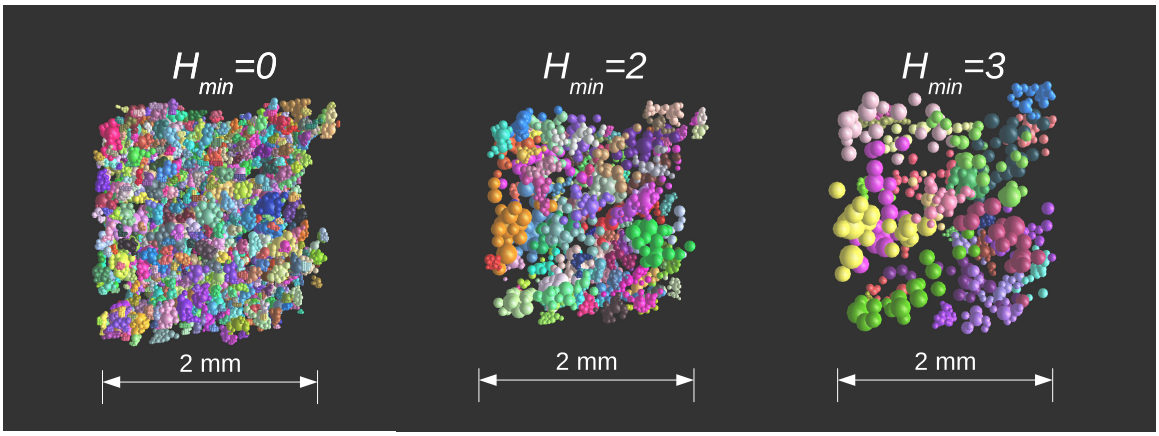


Figure 6.1: Multisphere assemblies for three different values of H_{min}. A lower H_{min} corresponds to more grains and bonds.

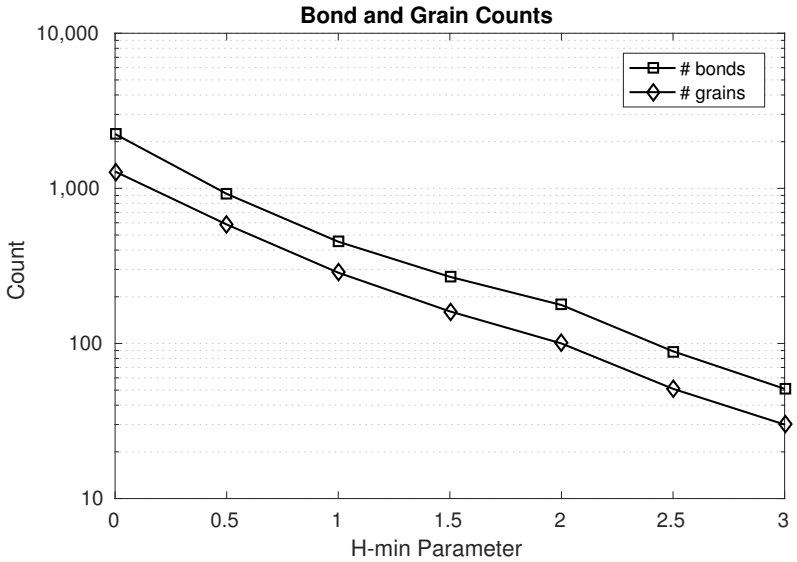


Figure 6.2: Number of bonds and grains generated with different H_{min} values in the watershed segmentation. Note the logarithmic scale on the y-axis.

The number of bonds varied from roughly 2,000 with $H_{min} = 0$ to 51 with $H_{min} = 3$. Number of grains ranged from about 1300 at imhmin=0 to 30 at imhmin=3. Interestingly, the coordination number stayed relatively constant at around 1.7.

Effect on Stress-Strain Relationship

As H_{min} was increased, segmentation was clearly affected and in turn the mechanical response of the sample was affected as well. Both stiffness and strength of the sample increased with increasing H_{min} (Fig. 6.3). This increase in strength and stiffness was accompanied by larger fluctuations in force as the sample failed. At low $H_{min}(\leq 1.5)$, there seemed to be a more gradual failure of the sample. With $H_{min} = 2$ there was one giant drop in force seen on the stress-strain curve. The increased stiffness and "pop" at failure may be attributable to loading and unloading in the sample occurring in just a few large bonds, rather than in many small bonds. Laboratory experiments (See chapter 2) were also characterized by sudden, rather than gradual, unloading events, and so for replicating the laboratory stress-strain relationship a higher (≥ 2) H_{min} value is suggested.

Effect on Energy Dissipation

Overall, varying H_{min} had little effect on cumulative energy dissipation, but it did affect the suddenness of dissipation. Figure 6.4 shows that the dissipation at $H_{min} = 1$ and $H_{min} = 1.5$ were nearly identical. Overall, $H_{min} = 2$ tracked these results but with fewer, more substantial dissipative events followed by longer intervals with no dissipation. Again, this seems to show that a less segmented sample stores its elastic strain energy in a few large bonds, rather than many small bonds.

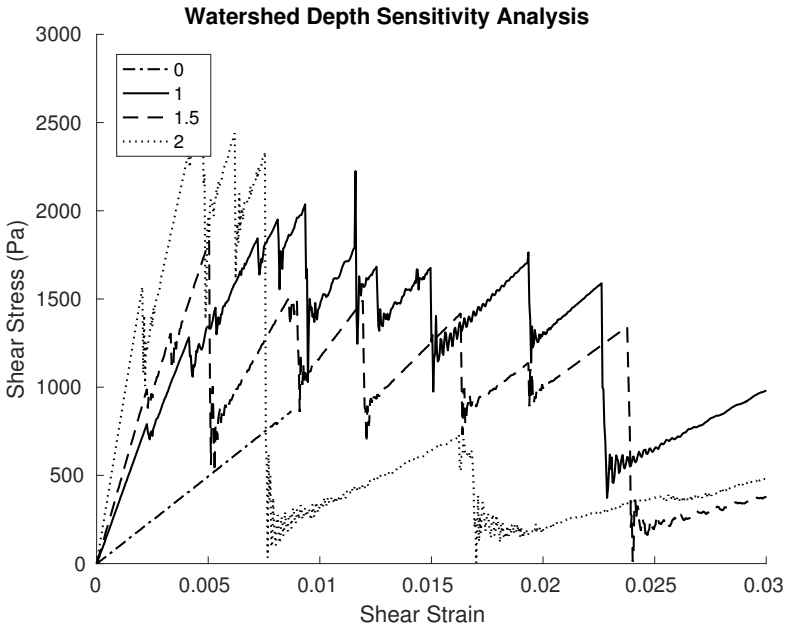


Figure 6.3: Comparison of stress-strain curves for the same sample after applying four different watershed depth criteria

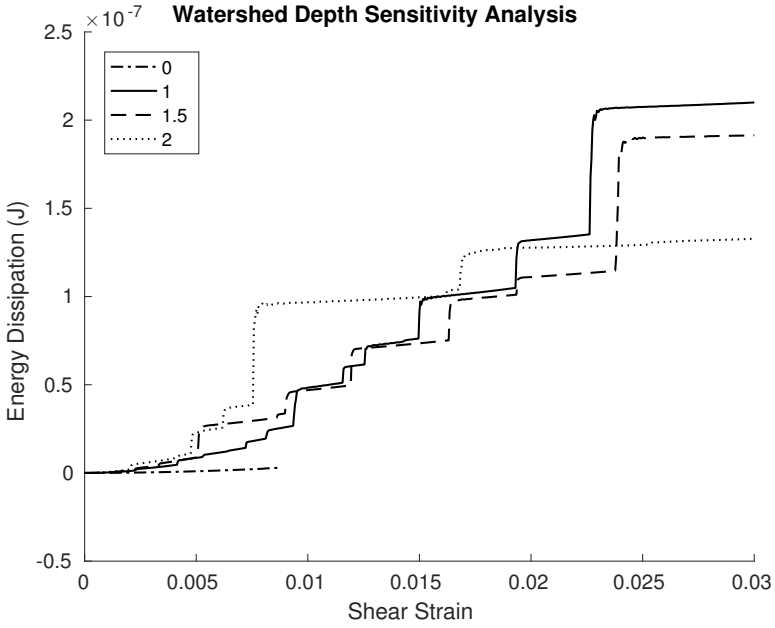


Figure 6.4: Comparison of dissipation curves for the same sample after applying four different H_{min} values

Computational Considerations

As H_{min} was decreased, the number of bonds and grains in each sample increased. More bonds and grains required more computational resources. The results of this analysis are summarized in Figure 6.5. All samples in the model were loaded at 1 mm/s. However, with fewer bonds and grains (Higher H_{min}), the model ran faster, and with more bonds and grains it ran slower. The model's velocity in real time was used as a benchmark for computer performance.

All samples were run on Montana State University's "Hyalite" high-performance Linux cluster running CentOS 6. Tests were run simultaneously on one node of the machine, an Intel Xeon E5-2650 V+ CPU with 16 physical cores. Each variation of the model (7 in total) was assigned two physical cores to take advantage of the multithreading option in YADE. The least segmented sample ($H_{min} = 3$) ran about 250 times as fast as the most segmented sample ($H_{min} = 0$). Given the huge increase in computational efficiency, the highest possible value of H_{min} that would still give a realistic mechanical response was desired. $H_{min} = 2$ was chosen.

Sensitivity to Number of spheres per clump

After the watershed segmentation, assembling the clumped multispheres is the next task in modeling microstructure geometry. An arbitrary number of spheres may be used to create each clump. With more spheres, the model showed convergent behavior. However, models with more spheres were more computationally expensive. Therefore, the smallest number of spheres per clump that still gives good representation of microstructure was required. This sensitivity analysis was run on one 3x3x3 mm subsample of homogeneous rounded snow from test 20160804. Figure 6.6 shows that both sample strength and modulus stabilize at 20 spheres per clump.

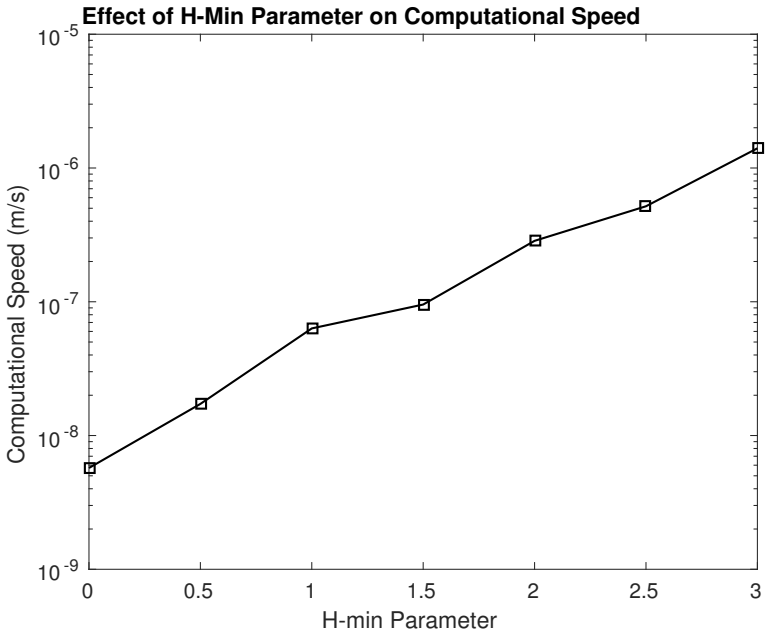


Figure 6.5: Comparison of computational speed of a sample after applying seven different watershed depths.

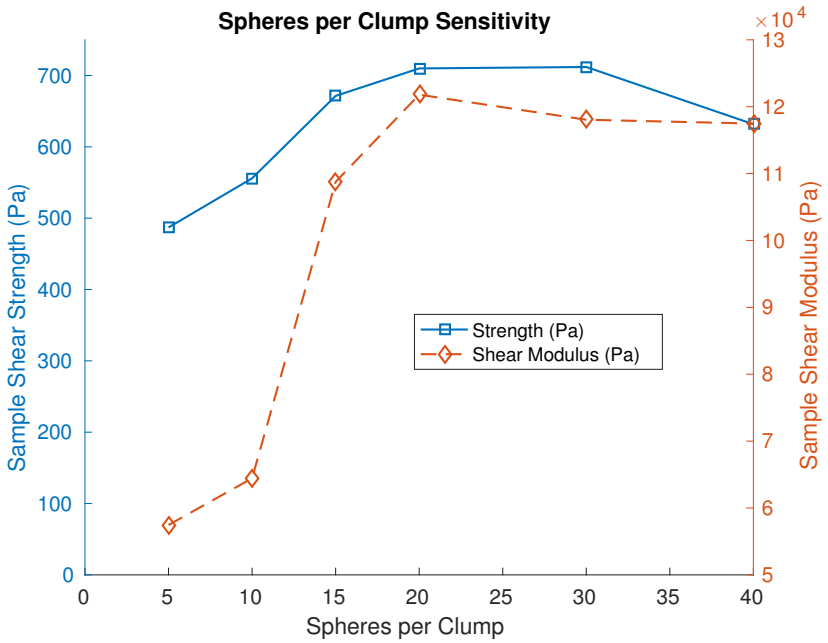


Figure 6.6: Comparison of bulk sample properties with variation of the number of spheres per clump

Energy dissipation during failure was also considered in this analysis. Dissipation did not vary appreciably as a function of the number of spheres per clump (Fig. 6.7). This is to be expected, as the image processing code creates the same bond network (a replica of the actual bond network, see Chapter 3) regardless of the number of spheres per clump. Since the number of spheres per clump did not affect dissipation, and strength and elastic modulus stablilized at 20 spheres per clump, all models were run with 20 spheres per clump.

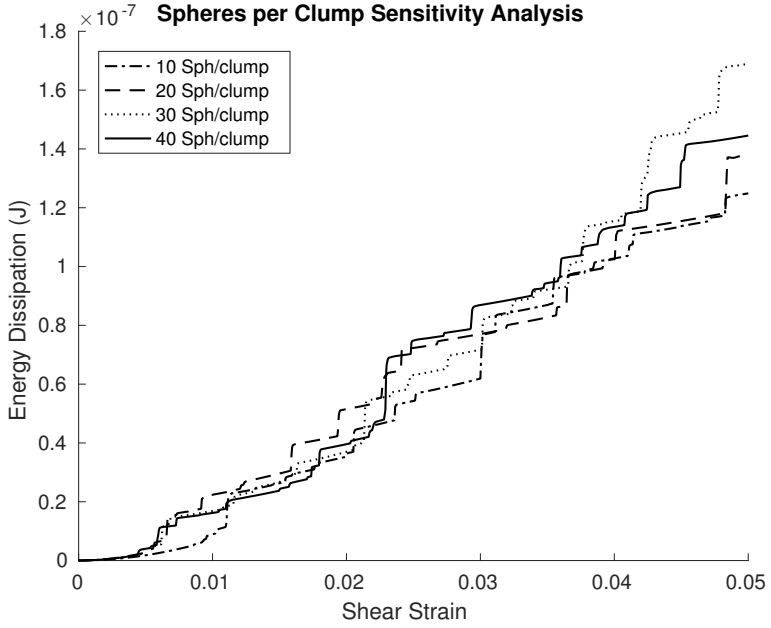


Figure 6.7: Number of spheres per clump has minimal effect on energy dissipation

Effect of Ice Network Constitutive Properties

For the model, the elastic modulus, poisson's ratio, density, and surface formation energy of ice had to be defined. Parametric analyses were performed for elastic modulus and density.

Elastic Modulus

Elastic modulus was paid particular attention. Numerical models of snow microstructure that use the "correct" elastic modulus of polycrystalline ice (about 9 GPa) tend to be too stiff (e.g. [89], [29]). This is to be expected since the elastic constitutive relations used here (Eqs. 5.27, 5.36,5.45,5.48) do not account for radii of spheres in contact; rather, only the bond is considered. Therefore, the elastic modulus of ice for use in the model was back-calculated from an analysis of 6 samples to optimize results in the elastic range. This was deemed acceptable because the purpose of the model is to analyze energy relevant to fracture, not elastic properties of samples.

In the constitutive approach used here, stiffness of any given bond is a function of E_{ice}. Therefore, for any snow sample comprised of many ice grains with modulus E_{ice}, the sample's shear modulus G_{snow} is proportional to E_{ice}:

$$G_{snow} \propto E_{ice} \tag{6.1}$$

Doubling the elastic modulus of the constituent material in the model will double the shear modulus of the entire sample and so forth. For six samples, values of E_{ice} were guessed for a first rough attempt to create an accurate linear elastic response over the entire sample. Results outside the linear elastic portion of the curve were not considered. These values were "corrected" to match the experimental result. For any given sample, let c be a correction factor and E^*_{ice} be an optimized elastic modulus of ice that, when used in the discrete element model, yields the correct linear elastic response of the snow grain network.

$$E^*_{ice} = cE_{ice} \tag{6.2}$$

Given Eq. 6.1, it follows that

$$G_{snow}* = cG_{snow} \tag{6.3}$$

To match the model to the experiments, let values with a * be the correct experimental values and the un-starred values are the modeled values that need correcting. Let τ and α be the values of shear stress and shear strain, respectively, along the stress-strain plot of the uncorrected model. Let τ^* and α^* be experimental shear stress and strain, respectively. The objective is to correct τ and α to match τ^* and α^* as closely as possible. Therefore, for any one point on the stress-strain curve, Eq. 6.3 becomes

$$\frac{\tau^*}{\alpha^*} = c\frac{\tau}{\alpha} \tag{6.4}$$

Also note that the strength of each bond scales with $\sqrt{E_{ice}}$, and therefore

$$\tau^* = \sqrt{c}\tau \tag{6.5}$$

Combining Eq. 6.5 and 6.4 yields

$$\alpha^* = \frac{\alpha}{\sqrt{c}} \tag{6.6}$$

Equations 6.5 and 6.6 may be used to correct one modeled data point to fit it to the experimental stress-strain curve. Of course, these equations will yield different values of c for each data point within one sample because neither the experimental nor numerical stress-strain curves is perfectly linear. Therefore, c was adjusted iteratively until the root mean square error of the linear portion of each sample was minimized. Optimized values of E_{ice} are given in Table 6.1.

Table 6.1: Values of E^*_{ice} from six different samples

Sample	$E^*_{ice}(Pa)$
20160802	1.22E08
20160803	1.31E08
20160804	1.38E08
20160809	7.70E07
20160916	6.34E08
20161120	5.45E08

The mean value of E^*_{ice} from all six samples, $2.74E08\,Pa$ was used as E_{ice} in the discrete element model.

Density

The density of ice is $\approx 917\,\frac{kg}{m^3}$. The density of ice used in the model is important for calculating kinetic energy terms, since kinetic energy is proportional to mass. Increased mass allows a longer critical timestep in the model (Eq. 5.14), resulting in faster runtimes for simulations. This numerical "trick" to speed up the simulation by increasing density has been used before in snow [29] with success and will be replicated here. It is argued that the kinetic energy in the model is insignificant relative to the work done and elastic strain energy in the model, and furthermore, altering the density of the material does little to affect the outcome of the model. A 3x3x3mm cube of snow from sample 20161120 is used here and densities of 1x,5x,10x and 20x the density of ice are tested. Parameters used for this sensitivity analysis are summarized in Table 6.3.

For the four densities of ice tested, energy terms were nearly identical for the range of strains considered. Figure 6.8 shows work done, strain energy, and kinetic energy for samples with densities $1 * \rho_{ice}, 5 * \rho_{ice}, 10 * \rho_{ice}$ and $20 * \rho_{ice}$. Work done is shown in blue, strain energy is shown in red, and kinetic energy in green. Sample

Table 6.2: Model parameters used for density sensitivity analysis

Test Sample	20161120
H_{min}	2.0
Spheres per clump	20
$E_{ice}\,(Pa)$	2.74E08
$Timestep$	$0.1t_{cr}$

density is denoted by line type (see legend). Varying the density makes so little difference in results that the plot is almost illegible, although it can be seen that the sample with density $20 * \rho_{ice}$ does deviate slightly from the rest.

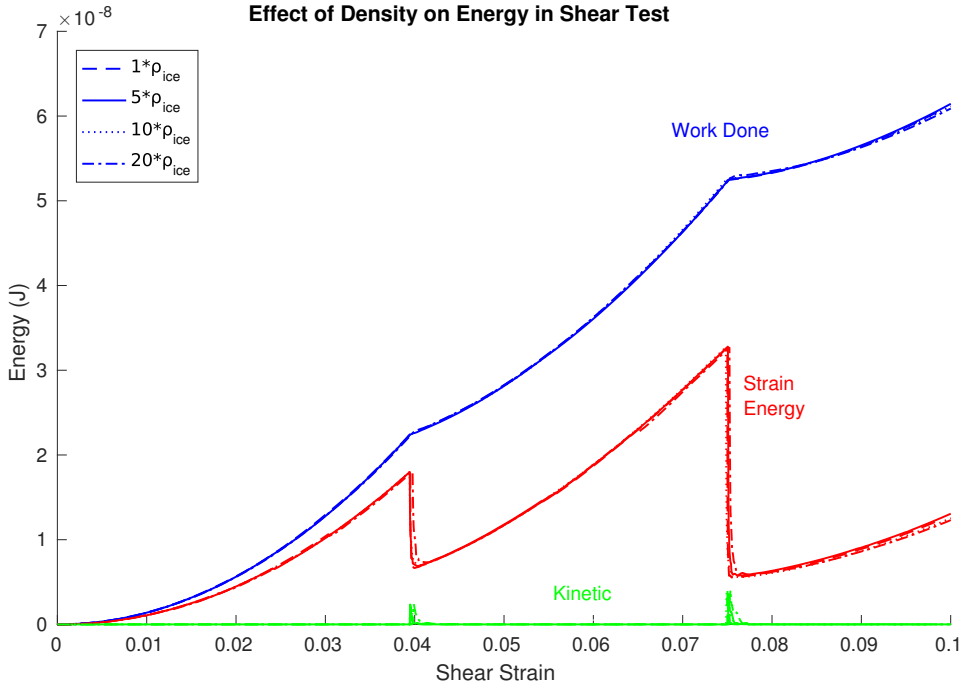


Figure 6.8: Ice density has minimal effect on work done, strain energy and kinetic energy.

A close-up of one bond breakage event that occurred at shear strain ≈ 0.075 is shown in Figure 6.9. As expected, all samples showed a steep drop in strain energy

as the bond broke. Simultaneously, all samples showed an abrupt increase in kinetic energy as some of the elastic energy stored in the bond was converted into kinetic energy. The peak value of kinetic energy was virtually identical among all samples. Damping ensued and eventually, kinetic energy in all samples returned to ≈ 0. The sample with density $20 * \rho_{ice}$ took longest to damp out the kinetic energy, while the sample with density $1 * \rho_{ice}$ was damped fastest. After damping was complete (shear strain ≈ 0.077), work done, strain energy, and kinetic energy became nearly identical across samples once again.

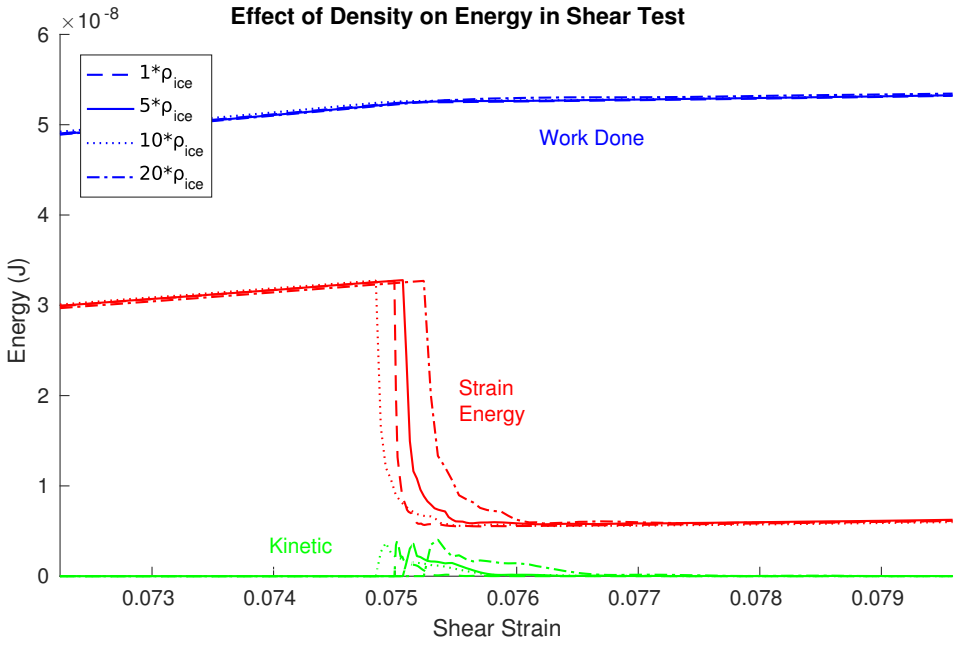


Figure 6.9: Close-up view of one bond breakage event

This analysis demonstrates that in the long run, density has minimal effect on energy terms measured within the modeled sample. However, the long span over which sample $\rho = 20 * \rho_{ice}$ was damped was undesirably excessive and possibly unrealistic. Therefore, a density of $10 * \rho_{ice}$ $(9170 kg/m^3)$ was used in the discrete element model as it had virtually no effect on energy within the sample.

Surface Formation Energy

The minimum surface formation energy for ice is $0.0757 J/m^2$ [37]. Using this value in the model would assume perfectly elastic behavior at the grain boundaries.

However, plastic phenomena at grain boundaries are well documented. [27], surface formation energy as high as $0.3 J/m^2$ has been found for granular ice [90]. As the DEM presented here seeks to create a realistic simulation of energy release, use of a realistic material property for ice was important. No attempt was made to calibrate or tweak this parameter. A value between the two extremes discussed here was chosen: $0.2 J/m^2$. Note that this is the energy required to form one surface, and so the constitutive property used in the model is

$$G_{c(ice)} = 2 \times 0.2\, J/m^2 = 0.4\, J/m^2 \tag{6.7}$$

Summary of Parameters Chosen

All parameters chosen in above sections are tabulated here.

Table 6.3: Model parameters used for density sensitivity analysis

H_{min}	2.0
Spheres per clump	20
$E_{ice}\,(Pa)$	2.74E08
$G_{c(ice)}\,(J/m^2)$	0.4
$\rho_{ice}\,(kg/m^3)$	9170
ν_{ice}	0.2
$Timestep$	$0.1t_{cr}$

DEM Validation Using Simple Geometry

When employing a large, complex numerical model, it is prudent to model a simple geometry with a known solution to ensure that the results from the model are reasonable. The model was validated by running simple 2-sphere geometry with a known bond size. The spheres were failed in shear and in tension to ensure that the energy released in the model is the same as that predicted by the constitutive relations in Ch. 5.

A simple two-sphere binary image stack was created (Fig. 6.10) and run through the image processing algorithm. These spheres were generated to mimic a CT scan of two spheres of radius $10^{-4}m$ connected by a bond of area $A = 10^{-10}m^2$. Image pixel scale was arbitrarily set at $10^{-5}m$. Spheres were placed so that their area of overlap (i.e. "bond") was exactly one pixel with an area of $10^{-10}m^2$. Images are arranged in the figure such that image 1 is at the top left corner, image 11 is at row 2 column 1, etc.

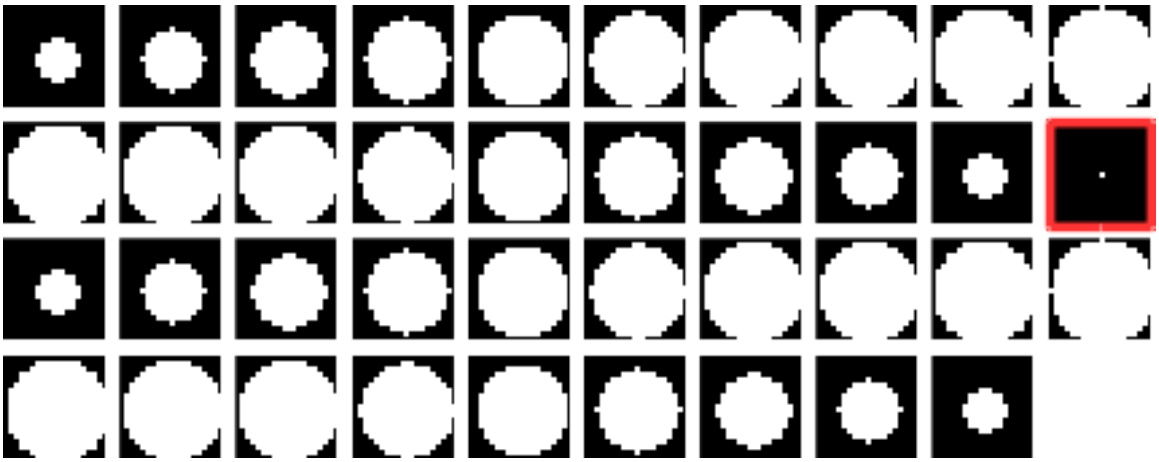


Figure 6.10: 39 binary images comprise the image stack for 2 spheres. The image with the 1-pixel "bond" is outlined in red.

Once the images are combined into a 3D binary image and segmented through watershed segmentation, they appear as two distinct spheres (Fig. 6.11) as expected. In the YADE simulation, these two spheres are two clumps comprised of one sphere each.

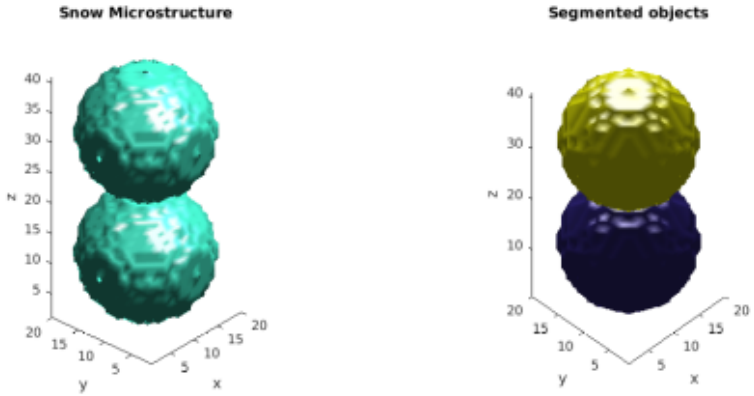


Figure 6.11: The binary images from Figure 6.10 before segmentation (left) and after (right).

The model is then run in YADE. Two screenshots from YADE (Fig. 6.12) show the spheres before (left) and after (right) shear failure. Note that bonds do not appear in this visualization.

Prediction of Energy Release

Bonds are modeled as linear elastic brittle springs. The energy stored in such a spring immediately prior to failure (and therefore the energy dissipated at failure) is

$$U = \frac{1}{2} k \delta_c^2 \tag{6.8}$$

where k is given by Eqs. 5.27 and 5.36 for normal and shear, respectively. δ_c is the relative displacement of sphere centers at failure. For normal stiffness, consider that there is a critical displacement $\delta_{c(n)}$ at which the bond breaks. $\delta_{c(n)}$ must occur

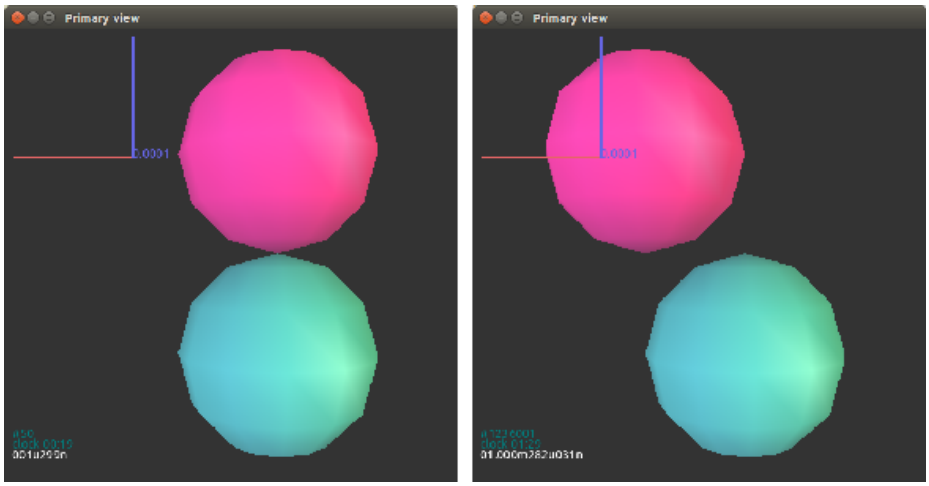


Figure 6.12: The simple two-sphere geometry is subjected to shear in the YADE environment. Initial configuration (left) and post-failure configuration (right)

when the normal force in the bond is at its maximum, defined by Eq. 5.45. Combining Eqs. 6.8, 5.27 and 5.45 yields the elastic energy stored at the instant before the bond breaks, and therefore the amount of elastic energy that must be released when bond i breaks:

$$U = 2G_c A_i \tag{6.9}$$

For the geometry considered, the model is expected to release $8 \times 10^{-11} J$ of elastic strain energy if the bond is broken in the normal direction.

Similarly, combining the relevant equations for shear (6.8,5.36 and 5.48) yields

$$U = \frac{4}{\pi} G_c A_i \tag{6.10}$$

For a bond i of area $A_i = 10^{-10} m^2$, energy release should be approximately $5.09 \times 10^{-11} J$ under shear failure.

The model shows exactly those values, shown in Figure 6.13. Initially, all work done on the sample goes into stored elastic energy in the bond. Then, when the bond breaks, elastic energy drops to zero. Other dissipative terms, such as kinetic energy and friction work, appear negligible.

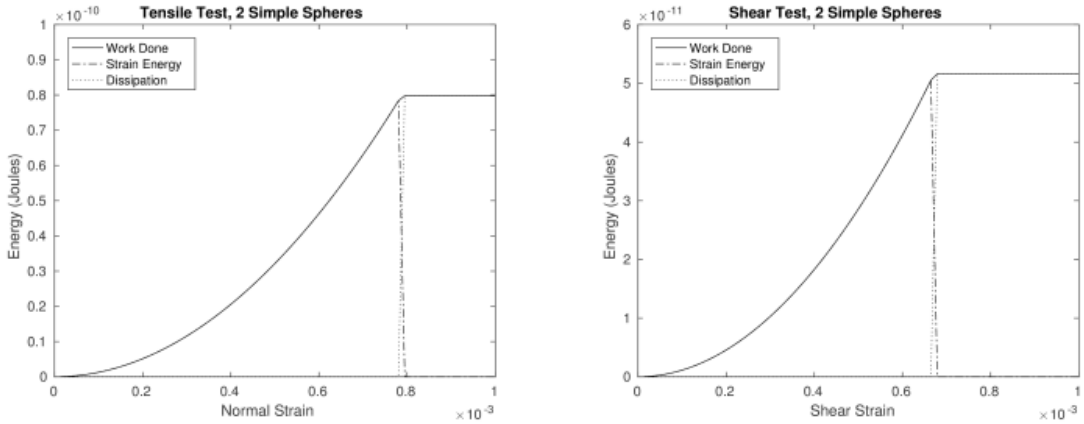


Figure 6.13: Energy released by failure of a simple sphere-sphere pair in tension (left) and shear (right).

The fact that the model dissipates the correct amount of energy implies that all steps of image processing and modeling relevant to bond calculations work properly. The following subroutines will all have an effect on modeled energy dissipation:

1. Watershed Segmentation: The watershed segmentation must choose the correct 1-voxel "bond" discussed above.

2. Bond Size Estimation: The bond sizing algorithm must correctly find that the 1-voxel "bond" output from watershed segmentation has cross-sectional area $10^{-10}m^2$.

3. Bond Stiffness and Strength: Values of k_n, k_t, P_c, Q_c in the model must be correct, or else Eqs. 6.9 and 6.10 would not correctly calculate modeled energy release.

DEM RESULTS

The model was validated through a series of laboratory shear experiments. In order to replicate fracture in natural snow, un-notched samples were used. It was hypothesized that notched samples provide an artificial stress concentration that may limit the size of the fracture process zone and not realistically simulate microcracking and/or damage accumulation. The idea was to let the naturally existing super-weak zones [3] within the sample initiate failure on their own, just as they would in an actual avalanche.

Experimental snow samples were labeled based on the date the experiment took place in a YYYYMMDD format. For example, 20160804 was tested on the 4th of August 2016. Multiple subsamples of microstructure were tested for all experiments. For example, subsample 20160804-7mm1 refers to a $7 \times 7 \times 7mm^3$ subsample from 20160804.

Spatial and Temporal Analysis of Bond Breakage Events

Both experiments and DEM were analyzed in a way that makes no assumptions about macroscopic failure modes. In this section, bond breakage events in the model are analyzed to determine whether a traditional fracture propagation model is applicable. To determine if the sample conforms to a traditional Mode I or Mode II model, numerical samples are tested against two criteria:

1. Bond failures must discernibly propagate in the $+x$ direction

2. There must be a discernible, finite Fracture Process Zone (FPZ)

Direction of Propagation

As the model ran, the location and time of failure of each bond was recorded. The coordinate of each bond failure was plotted against time of failure. In a conventional Mode II fracture, the crack propagates in the direction of loading (in this case the $+x$ direction). The correlation between simulation time and x coordinate was analyzed. A correlation with a positive slope would mean that fracture was moving in the $+x$ direction over time and therefore fit the pattern of Mode II fracture. A least squares line was fit to the data. The R^2 value of each fit line was recorded to determine if there was any correlation. R^2 values were negligible in all cases, meaning little or no linear correlation exists. Two samples are shown in Figs. 7.1 and 7.2, and similar plots for all samples are in Appendix D.

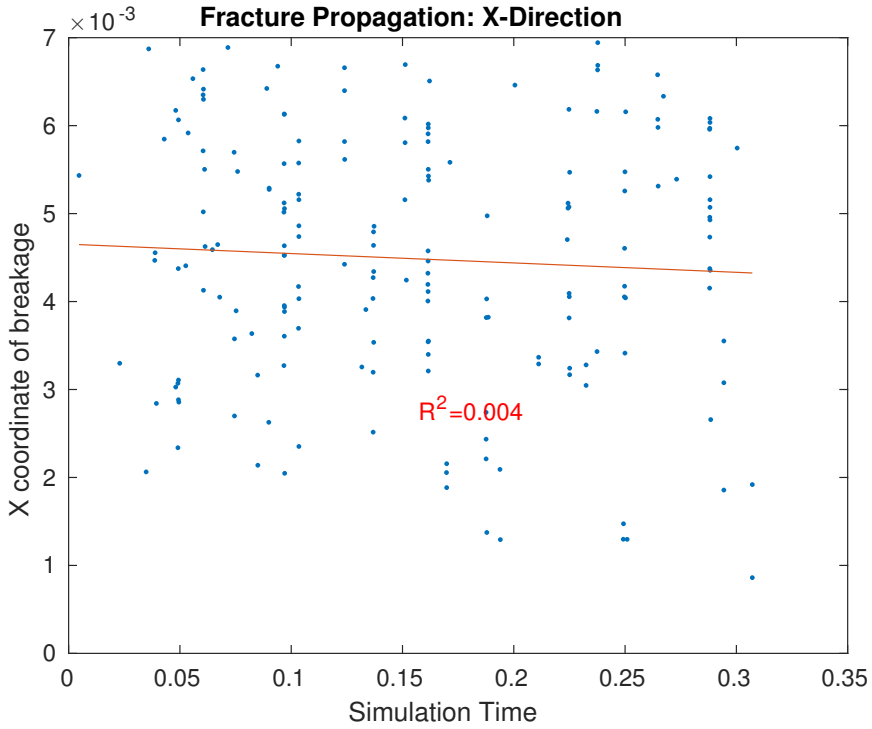


Figure 7.1: No correlation between bond breakage x coordinate and simulation time, indicating no process simulating crack extension. Homogeneous sample 20161120-7mm1.

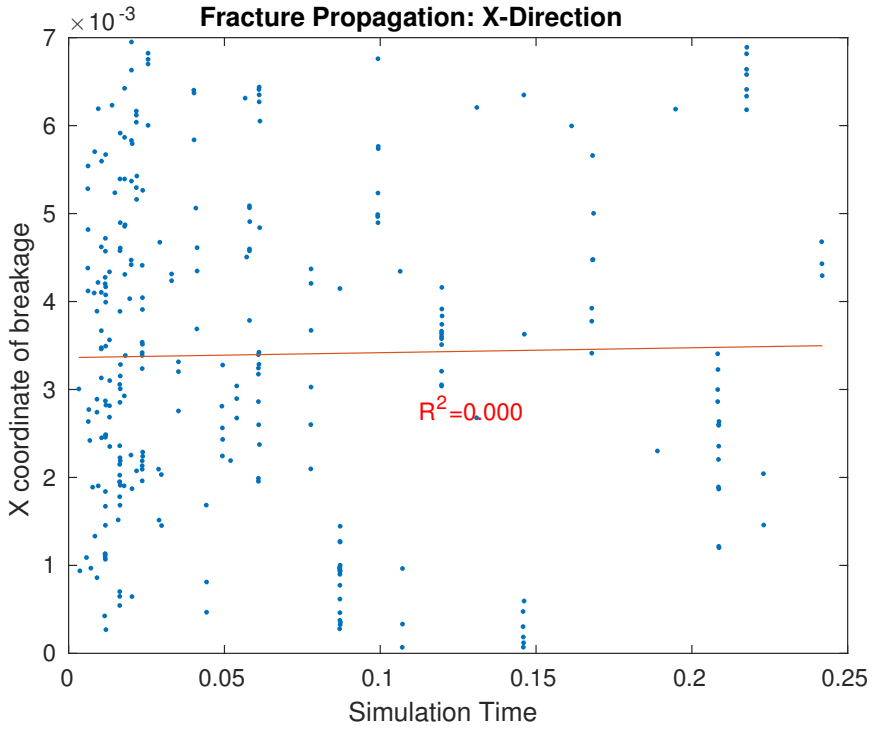


Figure 7.2: Little linear correlation between bond breakage x coordinate and simulation time, indicating no process simulating crack extension. Sample 20170127-7mm with buried SH.

All R^2 values for all numerical models are shown in Figs. 7.3 and 7.4.

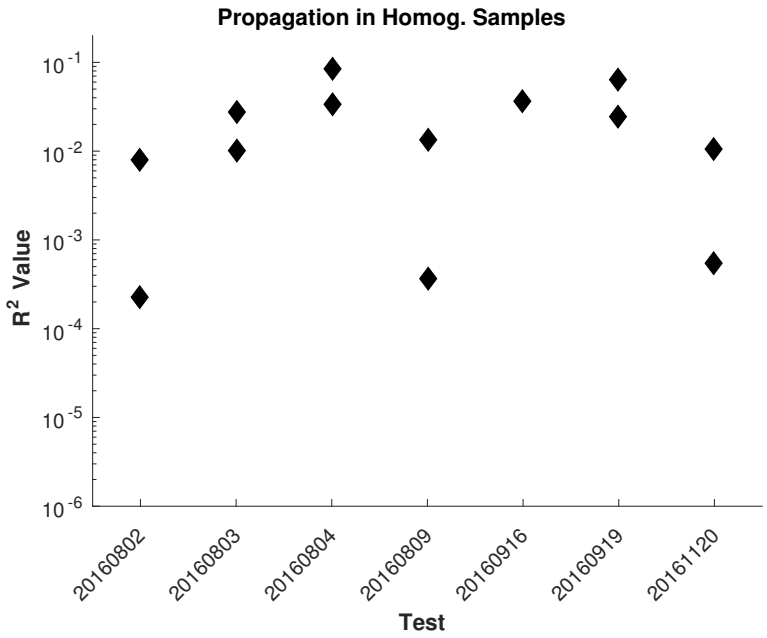


Figure 7.3: Plot of all R^2 values derived from plots like Fig. 7.1 in all homogeneous samples

In all cases, R^2 values were so low to be deemed insignificant, and it was concluded that these samples did not fit the criteria of a conventional Mode II fracture.

Size of FPZ

To determine whether or not each sample had a discernible fracture path, the z coordinates of all bond breakage events were considered. A perfectly brittle fracture will (theoretically) have a fracture process zone (FPZ) of negligible size. However, as damage increases, so may the size of the FPZ. In a granular material like snow, the size of the FPZ is defined by the spatial extent of bond breakage events.

To compare the height of the FPZ in numerical samples, the locations of all modeled bond breakage events were considered. The spatial extent in the z direction of the median 80% of all bond breakage events was measured. In other words, The

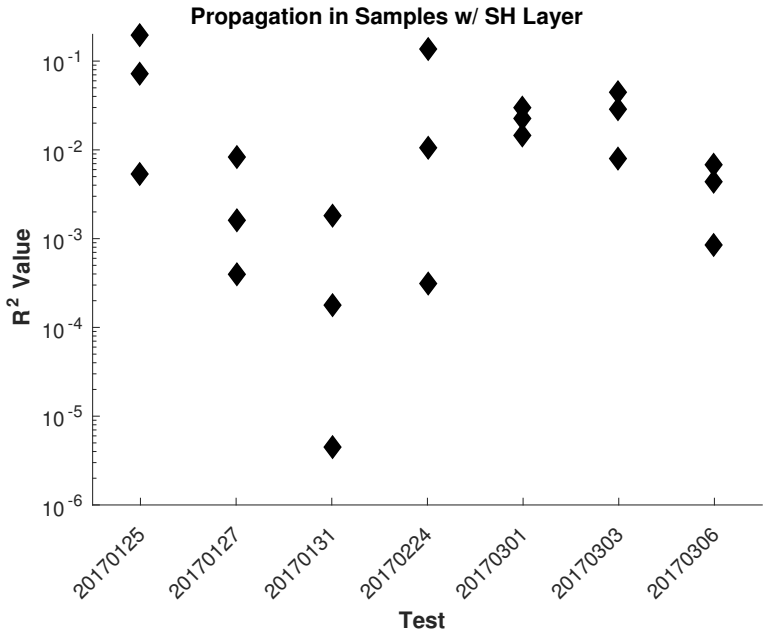


Figure 7.4: Plot showing all R^2 values derived from plots like Fig. 7.2 in all samples with SH layer

distance from the 10th percentile z coordinate to the 90th percentile z coordinate of all bond breakage events was considered. This distance will be referred to as the *10-90 span*. The 10-90 span was measured for all numerical samples. Samples with a larger span had a more diffuse failure process, while samples with a smaller 10-90 span had a more focused, fracture-like failure.

Most homogeneous samples had a large FPZ, that is, the 10-90 span was $\geq 80\%$ of sample height. That the span in which 80% of bond failures occured is itself 80% of sample height strongly indicates that failure of the sample happens across the entire volume.

Figure 7.5 plots all bond breakage events as a function of time in numerical subsample 20161120-7mm1. Each dot represents one bond breaking. It can be seen that bonds break throughout the height (z-direction) of the sample. In contrast, most samples with a buried weak layer showed significantly smaller FPZ height. FPZ

height was 20-70% of sample height. Consider figure 7.6, which shows Z coordinates of bond breakage events in sample 20170127-7mm with buried SH. Here, it can be seen that bond failure events are confined to a narrow band around the weak layer.

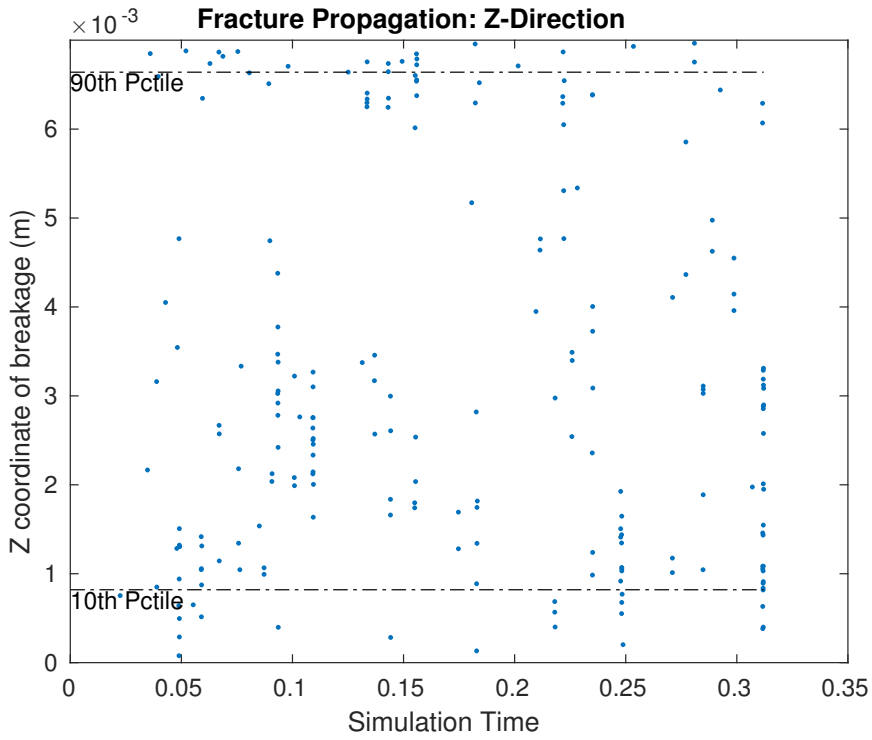


Figure 7.5: z coordinates of bond breakage events in homogeneous sample 20161120-7mm1.

To assess the FPZ height for all samples, 10-90 spans were compiled from all plots like Figs. 7.5 and 7.6 for all samples. Because the surface hoar growth process produces irregular surface hoar crystal sizes, the height of the 10-90 span was divided by total sample height to produce a normalized measure. Figs. 7.7 and 7.8 summarize results from all numerical models. In homogeneous samples, bond breakage was consistently a full-volume process, with the exception of sample 20160919.

In samples with buried weak layers, bond breakage was almost always confined to a small height within the sample, with the exception of sample 20170303. Sample

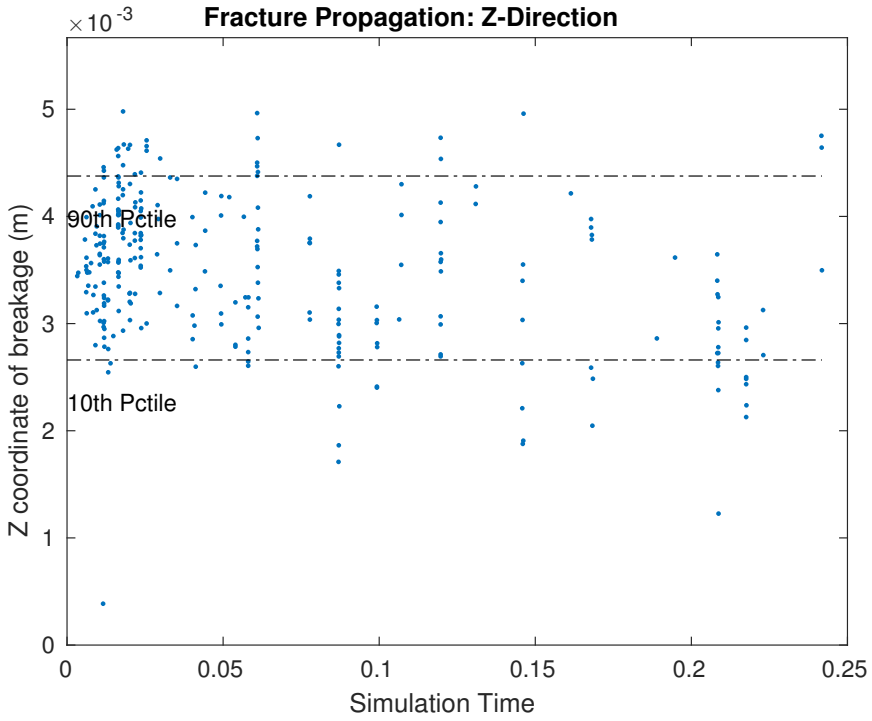


Figure 7.6: z coordinates of bond breakage events in sample 20170127-7mm with buried SH.

20170127 showed a very confined fracture process zone, while sample 20170306 showed that the 10-90 span was nearly 80% of sample height in one case. In summary, samples with buried SH showed a smaller spatial extent of bond breakage events, but results varied widely.

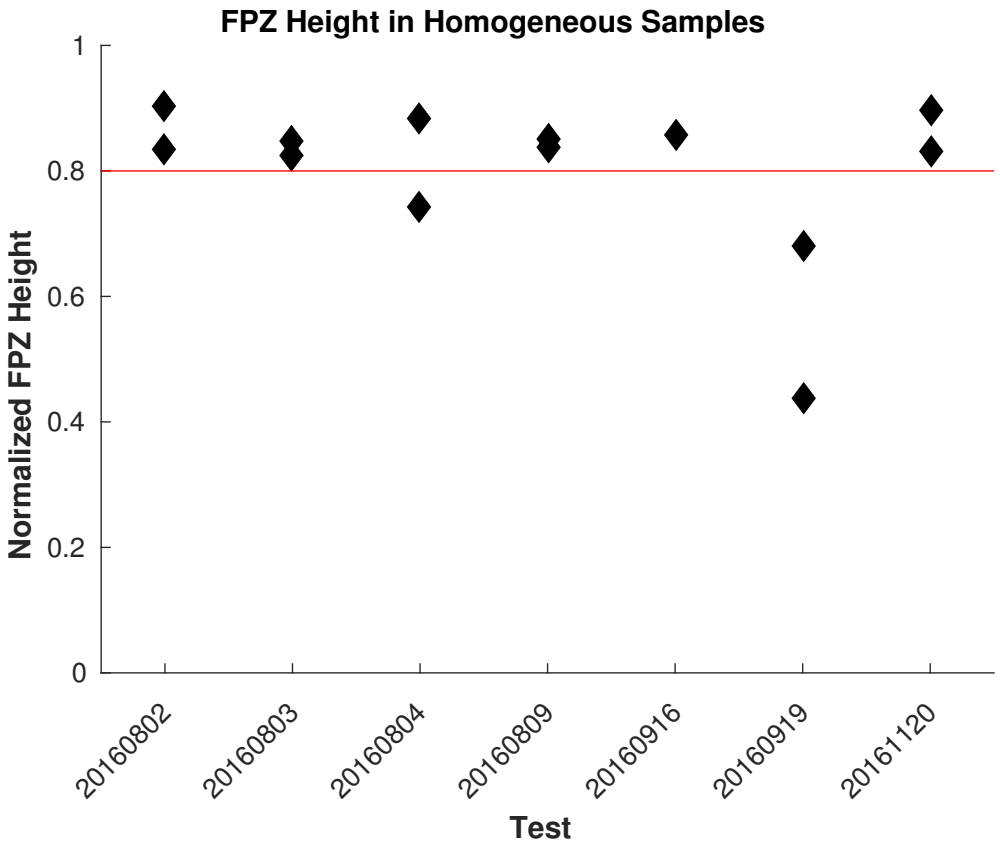


Figure 7.7: Normalized FPZ height for all homogeneous samples.

Representative Volume Element

Any numerical model has the objective of maximizing the amount of useful information gained while minimizing the computational cost. Therefore, the minimum representative volume element (RVE) had to be determined. Previous microstructural numerical models in snow have considered RVE's from $2.8^3 - 5.5^3\, mm^3$

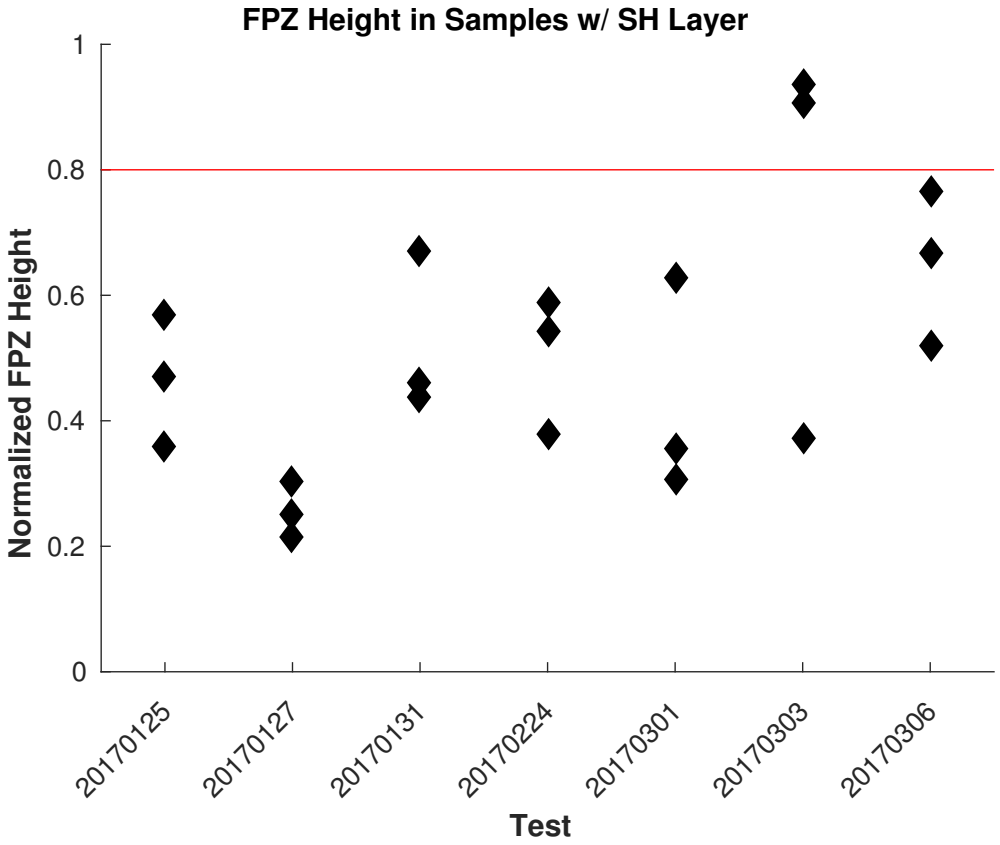


Figure 7.8: Normalized FPZ height for all samples with buried SH.

[31, 32, 89] and so $5^3\,mm^3$ was used as a starting point. However, shear fracture experiments in snow microstructure have not been considered before. Based on the results discussed above, where there was no clear propagation of fracture, it was hypothesized that the models were not large enough to capture the full failure process. Therefore, the largest RVEs possible were used. Whether or not these RVEs were adequate is discussed in the conclusions (Ch. 9).

From each experiment, two subsamples sized $7^3\,mm^3$ were attempted. In some cases, $7^3\,mm^3$ samples were unable to be processed due to memory limits and so $5^3 - 6^3\,mm^3$ subsamples were used.

Energy Dissipation

The main purpose of this project is to quantify energy released during failure of snow. The DEM was able to track the amount of total elastic energy stored in the sample. During deformation, all numerical samples saw a linear elastic portion wherein work done was approximately equal to stored elastic energy. Some work went into kinetic energy and was lost to damping, but these terms were consistently small.

As an example, consider numerical subsample 20160803-7mm1. This is a shear test on homogeneous sample 20160803, with a cubic subsample of dimensions $7x7x7mm$. The stress-strain curve appears linear until shear strain ≈ 0.002 (Fig. 7.9). However, analysis of model output tells us that the first bond breaks when shear strain $= 9.481E-4$. So, while the initial stress-strain relationship is roughly linear, dissipation of elastic energy starts before casual observation of the stress-strain curve might indicate.

This may be seen in a graph of energy terms plotted against shear strain. Fig. 7.10 shows the total work done on the sample and the total amount of elastic energy stored in all bonds. Dissipation is taken to be work done minus stored elastic energy. Because energy may be dissipated in several ways– through release of strain energy, grain-grain frictional work, or damping, this technique is a convenient way to account for energy lost through all dissipative mechanisms. Figures like 7.9 and 7.10 for all numerical model runs may be found in Appendix D

Comparison to Experiments

Energy dissipation was calculated for all experiments using the method outlined in Ch. 2. To compare energy dissipation in the small samples ($\approx 7\times7\times7mm$) used in the numerical model with the large samples used in experiments ($\approx 155\times155\times40mm$),

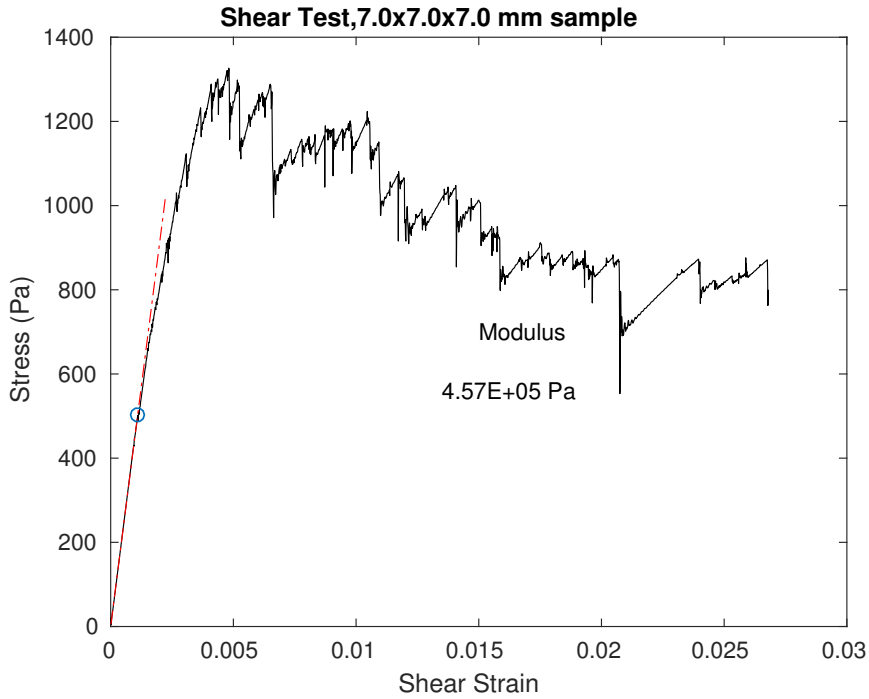


Figure 7.9: Shear stress-shear strain plot for numerical subsample 20160803-7mm1.

energy dissipation was scaled by sample size. Typically, energy release is reported as an energy release rate: energy dissipation divided by sample cross-sectional area. However, due to the evidence in this work of failure being a whole-volume process, energy release was divided by sample volume to create a valid comparison between model and experiment.

Energy dissipation in each modeled subsample was compared to energy dissipation in experiments. Although energy release rate is typically compared, the instant at which samples failed was not always obvious, and therefore one energy release rate for the sample could not be ascertained. Furthermore, it was desirable to examine the gradual damage accumulation throughout the loading of the sample, not just at the moment of failure.

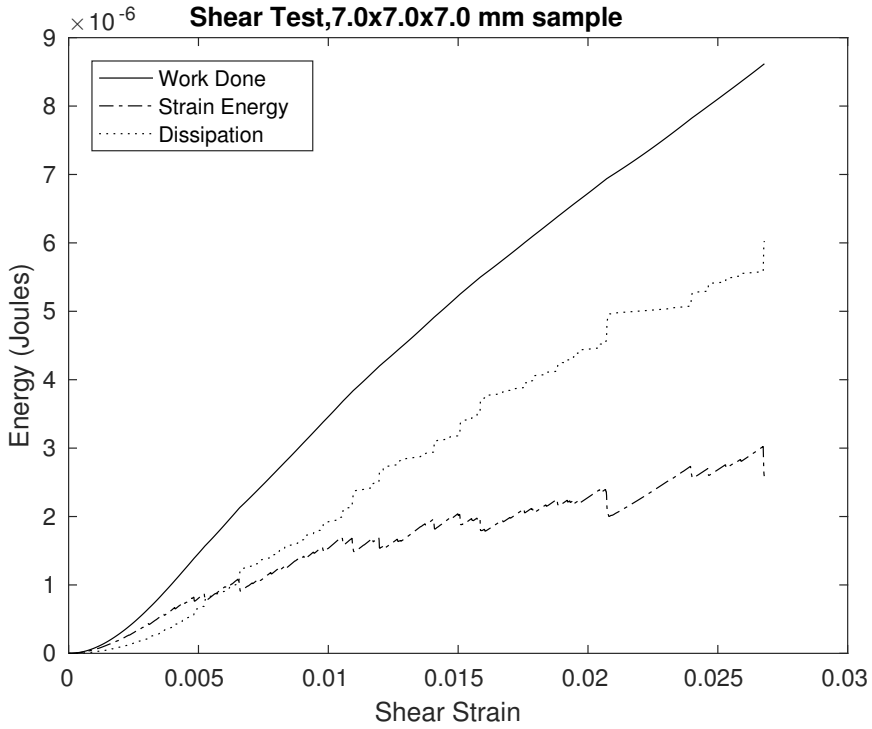


Figure 7.10: Plot of work done, stored elastic energy, and dissipation for numerical subsample 20160803-7mm1.

Therefore, energy dissipation of both experiment and model was plotted against shear strain. To evaluate goodness of model fit, a normalized root mean square error (NRMSE) technique was applied. For a sample with n data points y_i,

$$NRMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n\bar{y}^2}} \tag{7.1}$$

The character $\hat{y}_i$ refers to the "estimator" or, in this case, modeled value. The value y_i is the "correct" experimental value that corresponds to $\hat{y}_i$. $\bar{y}$ is the mean of all y_i.

For every subsample modeled, energy release vs. strain was compared to values of experimental energy release vs. strain. Since the experimental results were expressed as a range of possible values, y_i was taken to be the mean of the minimum and

maximum dissipation at a value of shear strain α_i. NRMSE was calculated over all points n in the numerical sample, since experimental samples typically strained the snow further than did the numerical models.

Homogeneous Samples

The typical fracture property G_c was calculated for experimental and numerical results. With this normalization technique (dividing energy release by cross-sectional area), the modeled results were consistently too low. However, it was noted that in the model failure is a whole-volume process. When energy release was normalized by volume, results were much better. An example from test 20160802 is shown in Figure 7.11. Similar figures for all tests are found in Appendix D.

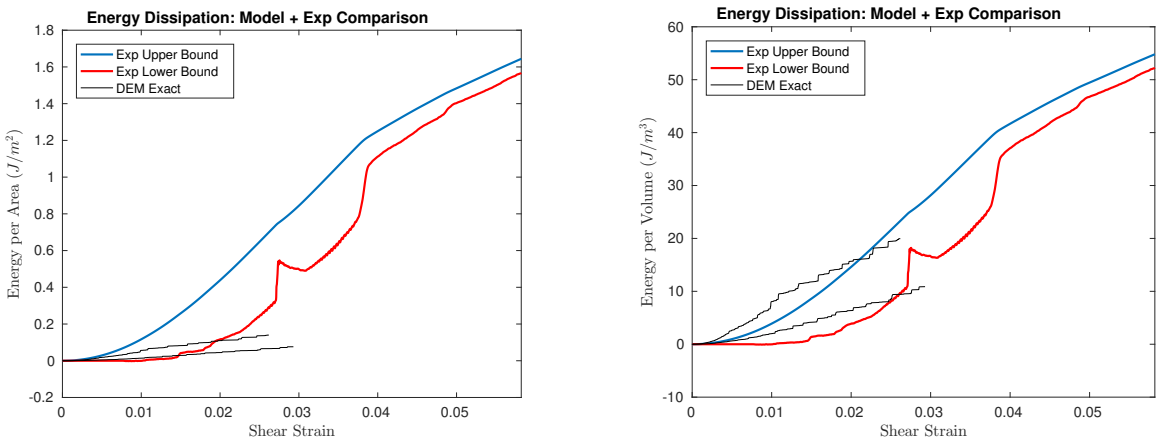


Figure 7.11: Dividing energy released by cross-sectional area (left) consistently yields low estimates from the model. Dividing energy by volume (right) yields consistently better results

By comparing NRMSE for energy per Area and NRMSE for energy per volume, it can be computed which model results better fit the data. Figure 7.12 compares NRMSE for all homogeneous samples. Black diamonds show NRMSE when energy release is normalized over volume, and blue dots show NRMSE when those same samples are normalized over area. A paired T-test of corresponding NRMSE values

rejects the null hypothesis ($p = 0.05$), meaning that the NRMSE is significantly lower for results normalized by volume than for results normalized by area.

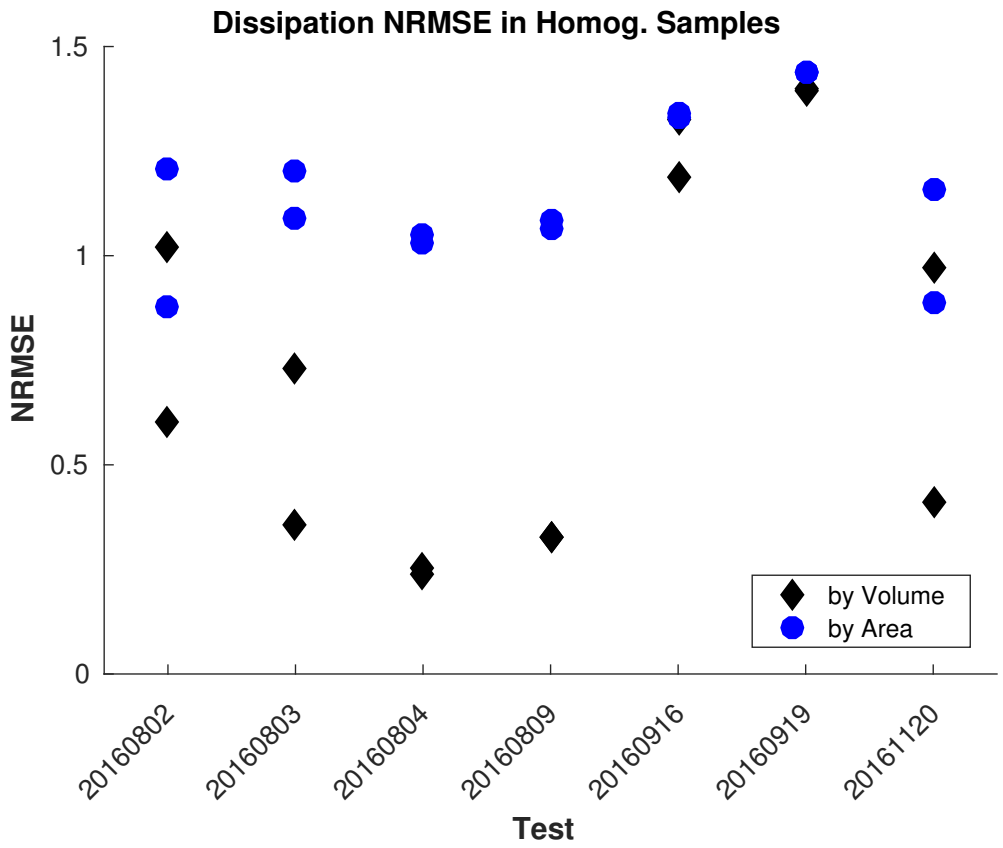


Figure 7.12: In homogeneous samples, normalizing by sample volume yielded lower NRMSE than normalizing by sample cross-sectional area

Samples with Buried SH

For surface hoar, it was hypothesized that the damage/failure process would be limited to the weak layer, and therefore dividing energy dissipation by sample cross-sectional area was a better way to compare experimental to model results. This assumption implies that, since damage is limited, LEFM may be a better approximation than for homogeneous samples. And indeed, some but not all numerical samples show a more brittle-elastic regime. For example, a DEM subsample

from test 20170127 shows a fairly linear-elastic regime with significantly less strain softening than seen in any homogeneous sample (Figure 7.13). Other samples, however, appeared to show significant strain softening and gradual unloading, just like homogeneous samples (Figure 7.14).

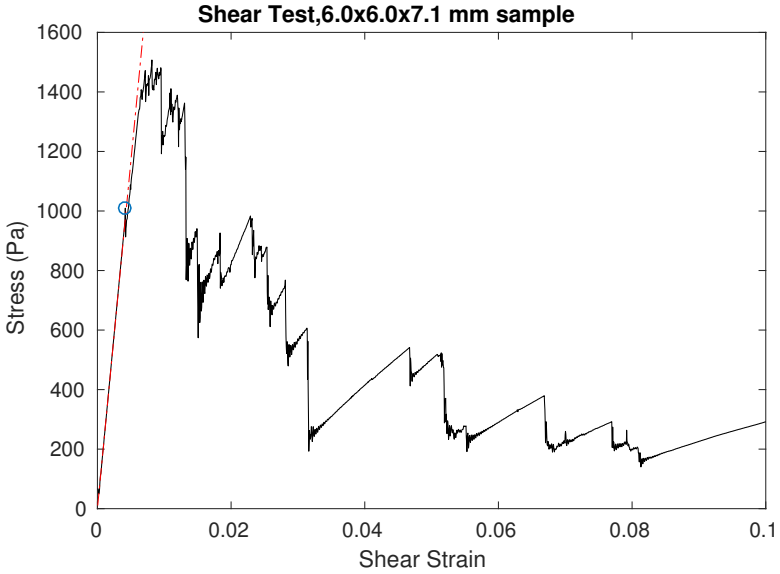


Figure 7.13: In this subsample from test 20170127, there is very little strain softening and and abrupt drop in force, behavior that closely approximates brittle failure

The extent to which the volumetric vs. area normalization technique applies can be examined. Figure 7.15 shows that one normalization technique is not necessarily better than the other. Detailed plots of energy release for all models are in Appendix D.

Across all modeled subsamples, normalizing by volume had mean NRMSE=0.563. Normalizing by area yielded mean NRMSE=1.403. In other words, across all SH samples, where damage was thought to be confined to a thin layer, NRMSE results show that the failure process is better explained as a whole-volume process.

It has already been hypothesized that when the failure/damage process occurs throughout the whole volume of a sample, then normalizing by volume is a good way

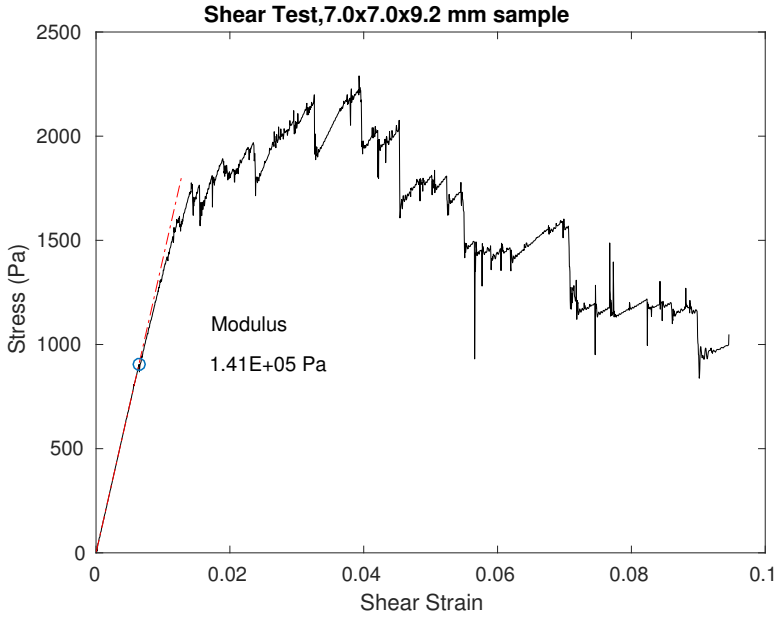


Figure 7.14: In this subsample from test 20170303, there is much strain softening and a gradual drop in force, suggesting fracture behavior that does not closely approximate brittle behavior

to make the model match the experimental results. Likewise, if the failure/damage process occurs in a small FPZ of limited height, then normalizing by area should yield a better result. If this is true, then samples with greater FPZ height should have higher NRMSE by area. A plot of NRMSE vs. FPZ height is given in Figure 7.16 for all modeled subsamples with surface hoar. There are three obvious samples with extremely high NRMSE. Two of these outliers occur in numerical subsamples where the 10-90 span is greater than 80% of sample height, indicating an obvious full-volume failure process. These two samples were deemed outliers, as they were SH samples where failure was not confined to the SH layer.

If these two outliers are discarded, a paired t-test does not reject the null hypothesis ($p = 0.05$). Therefore, there is no statistically significant difference between NRMSE based on area vs. volume normalization technique. Normalizing

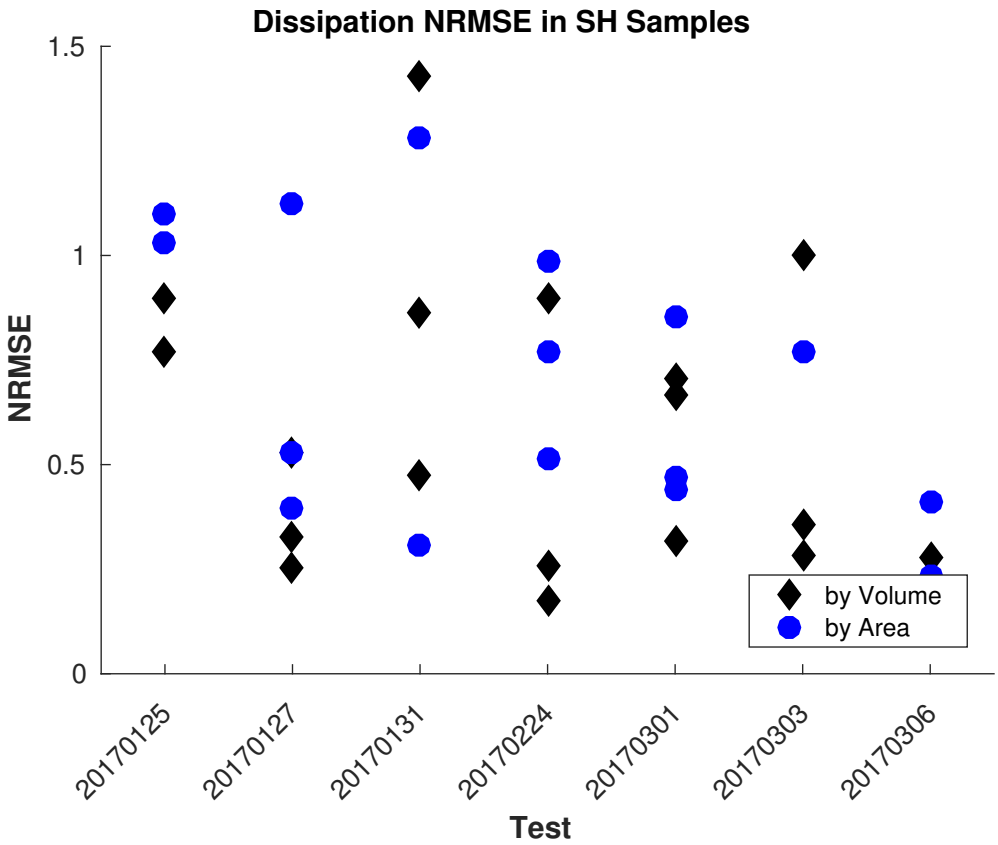


Figure 7.15: In samples with buried weak layer, normalizing by sample volume may yield a lower NRMSE than normalizing by sample cross-sectional area

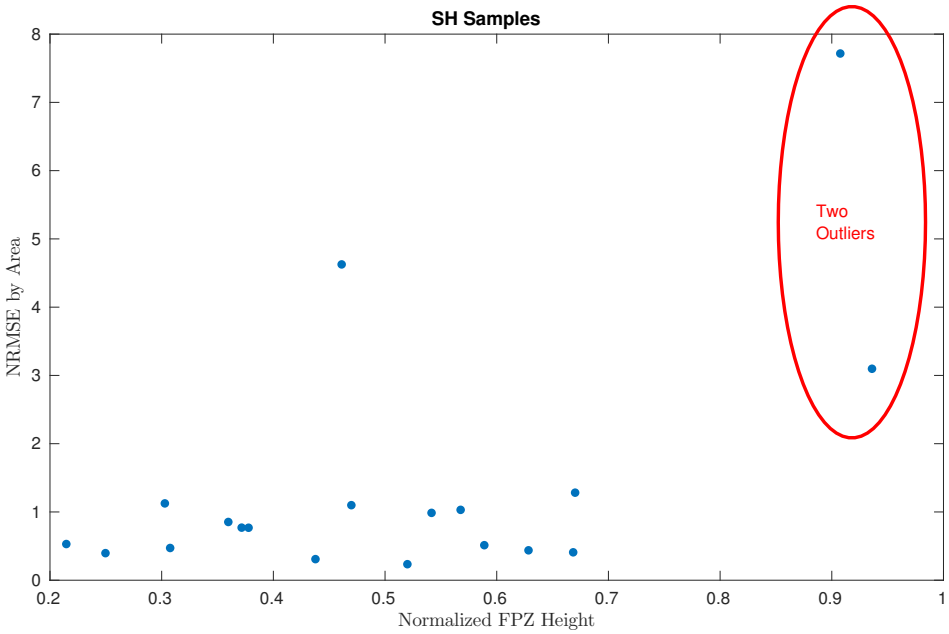


Figure 7.16: Of samples with buried SH where NRMSE was evaluated for an area normalization, mostly low NRMSE is observed with three very high results. Two of these results were considered outliers.

sample dissipation by volume yields about the same error as normalizing by area, indicating that overall, the failure process is neither whole-volume nor confined to a small FPZ. Rather, the reality lies somewhere in between. There is significant damage in the weak layer and in the slab. Furthermore, the existence of the two outliers can be taken to show that failure is not always confined to the weak layer.

Estimates of Energy Release Rate

Although the gradual failure behavior of the numercal DEM samples made quantifying G_c difficult, an attempt was made. Energy released at fracture propagation was taken to be the amount of energy dissipated when the sample exhibited its first major drop in stress as interpreted by the author. Results are summarized here. In general, for both homogeneous and layered samples, G_c estimated from DEM was an order of magnitude lower than G_c computed from laboratory experiments. This is to be expected, since the size effect of G_c has already been established. Figure 7.17 shows results for homogeneous samples, and Figure 7.18 shows results for samples with buried surface hoar.

Tensile Tests

Two numerical samples were subjected to tension in the DEM. While no laboratory tensile experiments were performed, numerical tests were nonetheless an interesting proof of concept that could be validated in the future. Two subsamples from test 20160804 (homogeneous) and two samples from test 20170125 (buried SH) were considered.

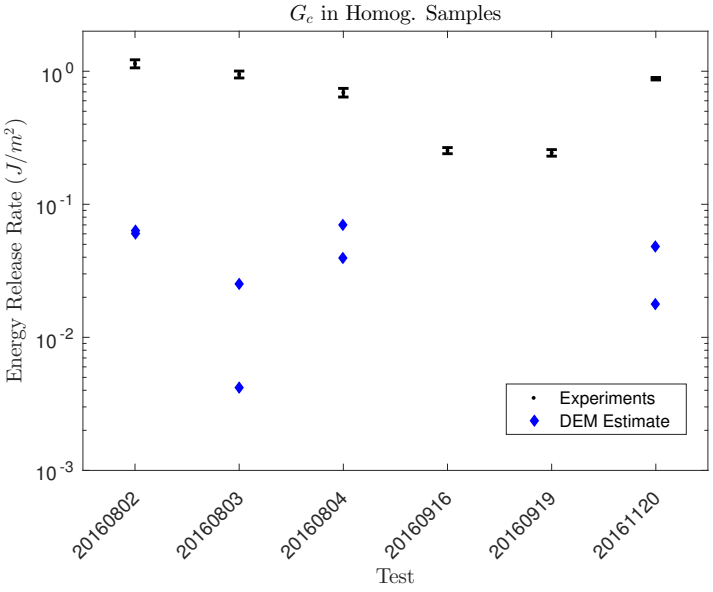


Figure 7.17: Experimental values of G_c are compared to results estimated from DEM for homogeneous samples.

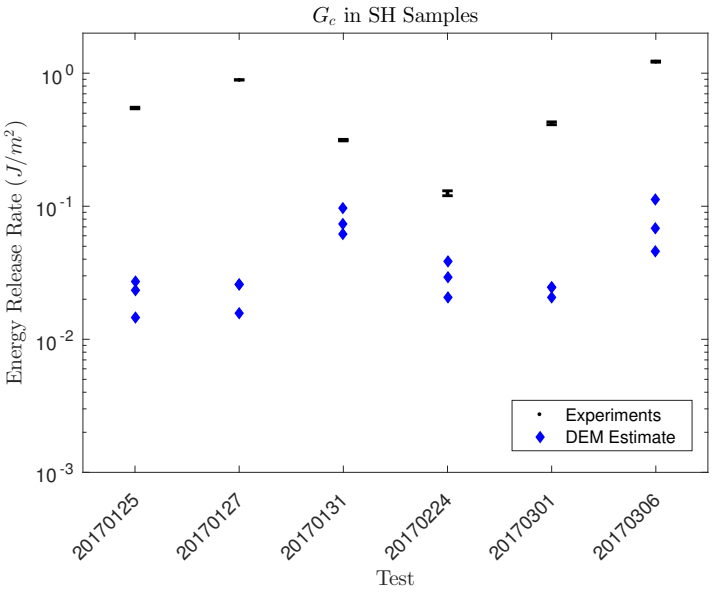


Figure 7.18: Experimental values of G_c are compared to results estimated from DEM for samples with buried SH.

Homogeneous Sample

Homogeneous sample 20160804 in tension showed results similar to shear tests. There was a linear portion of the stress-strain curve, followed by damage and strain softening. The exact instant of failure was not obvious, again showing that damage accumulation dominates the failure process, rather than fracture propagation. As in shear samples, bond breakage events did not clearly advance in the x direction. In both numerical subsamples, the 10-90 span of bond breakage events was approximately half of sample height, indicating a finite FPZ. Full results are in Appendix D.

Sample With Buried SH

Sample 20170125 with buried SH was also tested numerically in tension using the DEM. There was a linear portion of the stress-strain curve, followed by damage and strain softening. The exact instant of failure was not obvious, again showing that damage accumulation dominates the failure process, rather than fracture propagation. As in shear samples, bond breakage events did not clearly advance in the x direction. However, z coordinates of bond breakage events were confined to a narrow band. The 10-90 span was roughly half of sample height, indicating a finite FPZ. The SH samples were considerably weaker in tension than homogeneous samples. Full results are in Appendix D.

Discussion

Results show that for the sample sizes considered here ($5^3 - 6^3\, mm^3$ numerical samples, $15.5 \times 15.5 \times 2 - 6\, cm$ laboratory experiments), failure is a full-volume, damage accumulation process. Many numerical samples never showed the major drop in force

associated with fracture propagation, instead displaying elastic-plastic like stress-strain curves. Material around the tip of a crack within the FPZ characteristically exhibits plastic behavior. Because of this plastic behavior, it seems likely that the elastic-brittle assumption is not valid, at least for samples of this size. Even experimental samples consistently showed large amounts of strain softening and plastic behavior before failure.

ANALYSIS

Results are analyzed to see how they compare with other calculations of energy release, what microstructure may reveal generally about fracture in snow, and add physical insight to the snow fracture process.

Anticrack-Based Modeling from Field Experiments

van Herwijnen et al [85] have taken a novel work-energy approach to measure energy dissipation in the field during fracture. Using propagation saw tests (PST), they measure the change in gravitational potential energy of a collapsing slab and assume that this changing gravitational potential drives kinetic energy, elastic deformation in the slab, and fracture in the weak layer. Their field-based formulation of energy release is based on Heierli's anticrack model [34]. They posit that during crack propagation, total mechanical energy in the system is constant:

$$V_p = V_i(r) + V_e(r) + V_p(r) \tag{8.1}$$

The left side of Equation 8.1, V_p, is the initial gravitational potential energy in the slab and is a constant. $V_p(r)$ is gravitational potential energy as a function of crack length r during propagation, $V_i(r)$ is dissipation, and $V_e(r)$ is elastic strain energy.

The paper yields an interesting result: that more energy is dissipated in samples with thicker (taller) overlying slabs. In their experimental results, G_c varied by as much as an order of magnitude depending on overburden stress. They offer the explanation that weak layers subjected to a higher overburden pressure undergo more sintering and therefore require more energy to fail.

While the sintering explanation is plausible, another explanation is offered for these results: that some non-recoverable energy dissipation occurs within the slab, and therefore dissipation must increase with slab size. The van Herwijnen study presumes that slab deformation is purely elastic. This pure elastic assumption implies that, hypothetically, if gravity were removed from the system after failure, the slab would recover elastically. This supposition, while a straightforward consequence of the anticrack model, seems unlikely. Instead, there is likely unrecoverable energy dissipation (broken bonds, friction and grain movement) throughout the slab, possibly up to the slab depth.

Consider that in Equation 8.1, $V_e(r)$ is based on a linear elastic assumption, using an effective elastic modulus E^* as defined by Heierli [34]. If plasticity (damage) in the slab were considered, work done on the slab might be shown to increase, thereby requiring decreased $V_i(r)$ as per Equation 8.1.

Shear Model from Field Measurements

McClung [59] has produced estimates of G_{IIc} from field measurements based on a pure mode II quasi-brittle shear model. Again, the propagation saw test was used. Energy release rate calculation was based on Bazant et al's [5] size effect law for quasi-brittle, notched behavior.

$$G_{IIc} = \tau_N \sqrt{1 + \frac{D}{D_0}} \bar{\delta} \tag{8.2}$$

where D is slab depth, $\bar{\delta}$ is average weak layer slip, and τ_N is shear stress at the base of the slab (calculated from slope angle, slab depth and density). D_0 is a length scale that accounts for FPZ size, based on the work of Bazant and Planas for quasi-brittle fracture [4]:

$$D_0 = \left(\frac{2c_f}{\frac{L}{D} - \frac{c_f}{D}} \right) \tag{8.3}$$

where $2c_f$ is the size of the FPZ (McClung assumed 2-10 cm) and L is the length of the saw cut.

Interestingly, this calculation does not account for slab properties in the fracture calculation. It merely uses slab density and depth to compute overburden stress on the weak layer. However, the assumption that FPZ size is 2-10cm does imply that there could be some dissipation (bond breakage) in the slab. This study, unlike the van Herwijnen field study, is independent of any energy terms that may affect the slab. So, while the van Herwijnen results could have included dissipation in the slab, the McClung study isolates the weak layer dissipation.

McClung's results for G_{IIc} from PSTs are 0.04-0.1 J/m^2, significantly lower than van Herwijnen's 0.08-2.7 J/m^2. By eliminating the slab from energy considerations, McClung has focused on dissipation at (or at least near) the weak layer.

Work presented in this dissertation shows that slab damage is likely significant. The question then arises of what fracture energy is being computed. By assuming linear elastic slab behavior, van Herwijnen et al may have inadvertently counted slab dissipation in their computations of energy release, while McClung does not, implying an interesting question: Which term is relevant for avalanche initiation: weak layer energy release or total system (slab + weak layer) energy release? In the microstructural sense, there are no "layers," just individual grains and bonds.

The results here for samples with weak layers are closer to the results of van Herwijnen et al [85]. Values of G_c presented in this work make no effort to differentiate between slab dissipation and weak layer dissipation: they consider all dissipation.

Size Effect Law

McClung [61] estimates that about 40% of work done on a sample is dissipated during the fracture process. This differs from results here, where about 60-80% of work done on the sample is dissipated.

McClung's work was on notched samples, which display more brittle-like fracture character than un-notched samples [12]. Given that samples in this work were un-notched, it seems reasonable that more damage (non-elastic failure process) will occur here. This damage could occur in the slab or the weak layer.

McClung [61] calculates that homogeneous snow has a characteristic FPZ size of 1.5-3 m and that mode II energy release scales linearly with FPZ size:

$$G_{IIc} \approx 0.4 \left(\frac{32}{9\pi} \right) \omega \tag{8.4}$$

where ω is FPZ size. Given the large FPZ size calculated by McClung and Schweizer [60], it is reasonable to assume that the full volume of laboratory samples and numerical homogeneous specimens analyzed here are part of the FPZ. This supports the conclusion in chapter 7 that energy release rate scales linearly with specimen height in homogeneous snow. Figure 8.1 shows a good linear fit ($R^2 = 0.906$) of G_{IIc} vs. sample height for laboratory experiments.

Layered Samples

Bazant et al [5] also estimate that snow shear (Mode II, notched samples with weak layers, quasi-brittle, with cohesive crack faces) fracture toughness increases with slab height T_s in field measurements, althouth they assume that slab deformation is elastic. They conclude that

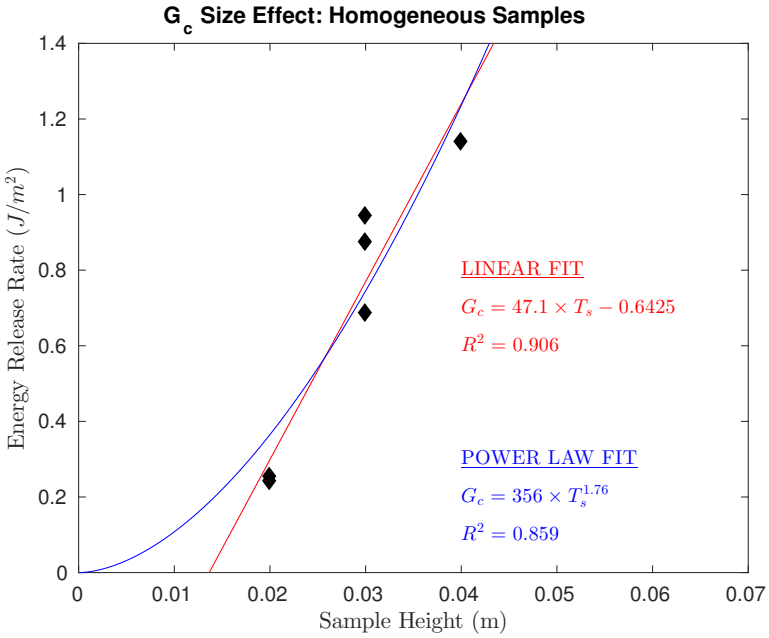


Figure 8.1: Fracture size effect in homogeneous laboratory samples

$$K_{IIC} \propto T_s^{1.8} \tag{8.5}$$

which, when combined with Eq. 1.5 yields

$$G_{IIc} \propto T_s^{3.6} \tag{8.6}$$

In Ch. 7 of this work, energy release is scaled by volume, implying that

$$G_c \propto T_s^1 \tag{8.7}$$

In experiments, both homogeneous samples and samples with buried surface hoar showed a positive correlation between sample height and energy release rate. Using the assumption that G_c scales linearly with sample height produces good results for

homogeneous samples ($R^2 = 0.906$), Fig. 8.1) and finds a weaker correlation for samples with buried surface hoar ($R^2 = 0.377$, Fig. 8.2).

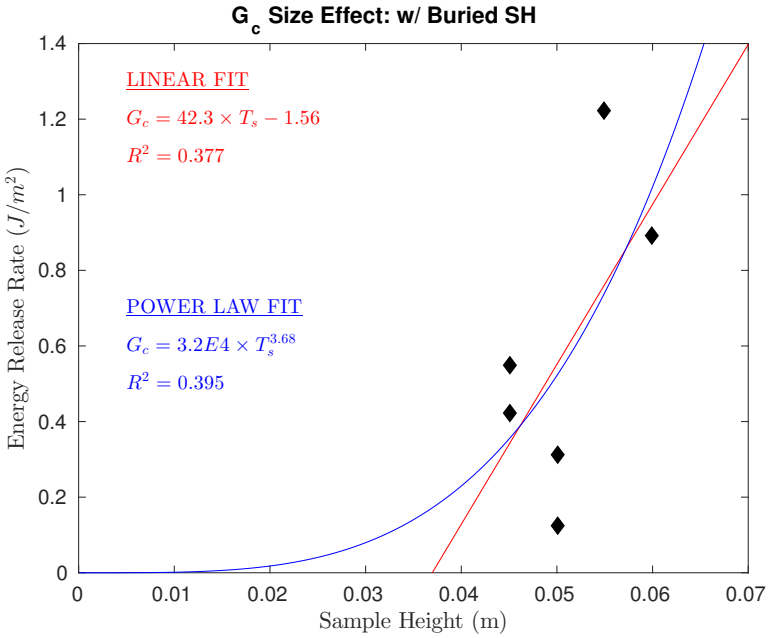


Figure 8.2: Fracture size effect in samples with buried surface hoar

Figures 8.1 and 8.2 also show the results of a power law fit.

$$G_c = aT_s^b \tag{8.8}$$

For homogeneous samples, power law fit yeilded $b = 1.76$, while for samples with buried surface hoar, $b = 3.68$, very close to Bazant et al's [5] $b = 3.6$ for field samples with weak layers. R^2 values for power law were very similar to R^2 for linear fit. However, samples with weak layers tested in this work were 2-6 cm tall, and Bazant's work was done on slabs in the field. So, while the results here fit nicely with Bazant et al's results, they should not be taken as an independent verification.

There may be several reasons that the size effect law does not fit as well (lower R^2) for SH samples as for homogeneous samples. First, homogeneous samples were

shorter (smaller T_s) than SH samples, and displayed a tendency to have a larger, more active damage process. So, with more damage occurring throughout a smaller volume, a situation where G_{IIc} scales with height seems more likely.

Within samples containing buried SH, there were more variables to contend with. First, SH growth in the laboratory is a fickle process that does not produce identical crystals each time, so there will inherently be more variation from test to test. Secondly, the watershed segmentation process used may not work as well on SH crystals as it does on rounded grains. For more discussion of this issue, see section 9.

The effect of sample height on G_{IIc} in numerical samples was not studied due to the small and very similar heights of all samples (5-7mm).

Size of FPZ and Uncertainty

It must be noted that estimates of FPZ size in snow can be large: McClung [58] estimates FPZ size of 5-10 cm for snow with weak layers and 1.5-3 m for homogeneous snow [61]. 10 cm is larger than any sample, numerical or physical, considered in this work. The evidence here does not contradict McClung's estimates of FPZ size. Laboratory shear experiments were performed on samples of height less than 5 cm, and numerical subsamples were an order of magnitude smaller. It was concluded that a full-volume damage accumulation process was applicable in both homogeneous and surface hoar samples. Testing of larger samples is required to determine if the fracture process zone is limited to 5-10 cm, or if it is in fact larger.

Numerical samples are especially limited in this respect. The tallest was 1 cm and most were on the order of 6-7mm, some of the largest microstructural samples of snow ever analyzed for shear fracture. With the right expertise, the model could likely be made more computationally efficient and permit the use of larger numerical samples.

McClung's 5-10 cm estimate is for notched samples with weak layers. Based on the prevalence of full-volume damage accumulation in snow samples here, the FPZ in un-notched homogeneous slabs is likely much larger.

Cohesive Crack Faces vs Large FPZ

The interesting result of increasing G_{IIc} with increasing slab depth could be driven by several mechanisms. In snow, cohesive crack faces have been considered in macrostructural numerical models [48, 54, 55]. These models assume that as the slab collapses, it makes cohesive contact with the substrate. As overburden pressure increases, so does cohesion, meaning a crack is more resistant to propagation. Then, fracture toughness and energy release rate are effectively increased.

However, work here implies that increasing G_{IIC} with slab depth can be explained by damage accumulation. In future modeling efforts, it may be desirable to account for cohesion as well in order to compute the contributions of both mechanisms to increased G_{IIc}.

Consideration of Minimum Energy Path

The minimum energy fracture path (See chapter 4) was found to consistently under-predict G_c compared to experiments. However, the minimum energy calculation does offer some useful insights. Minimum critical energy release rate $G_{c(min)}$ calculated by the minimum energy path can be considered the energy that would be released during fracture if LEFM were perfectly applicable. By comparing the experimental $G_{c(exp)}$ to $G_{c(min)}$, the amount of work that goes into pseudoplastic/damage behavior outside the primary fracture path can be computed.

As $G_{c(exp)}$ for a given experiment approaches its numerically calculated $G_{c(min)}$ value(s), damage must be less significant.

Figures 8.3 and 8.4 compare $G_{c(exp)}$ and $G_{c(min)}$ for all samples (Each experiment corresponds to 1-3 numerical subsamples). The minimum energy calculation consistently predicted $G_{c(min)}$ 1-2 orders of magnitude lower than the actual $G_{c(exp)}$.

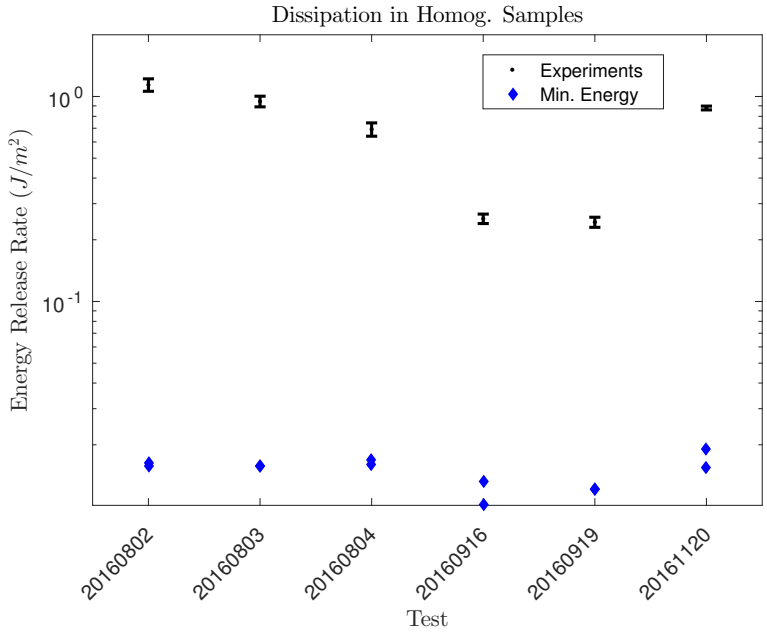


Figure 8.3: Minimum Cut G_c compared to actual experimental G_c in homogeneous samples

The ratio $G_{c(exp)}/G_{c(min)}$ is shown in figure 8.5 for all samples. A ratio of 1 would indicate that the experiment demonstrated perfect LEFM behavior, while higher ratios indicate significant damage accumulation outside the primary fracture path. On average, the ratio for homogeneous samples was 44.4 and the ratio for surface hoar samples was 29.3. Both of these numbers indicate significant damage accumulation.

Since surface hoar experiments had a lower ratio, they more closely approximated LEFM than did homogeneous samples. The $G_{c(exp)}/G_{c(min)}$ ratios for homogeneous snow were compared to the ratios for surface hoar samples using a t-test, and the null

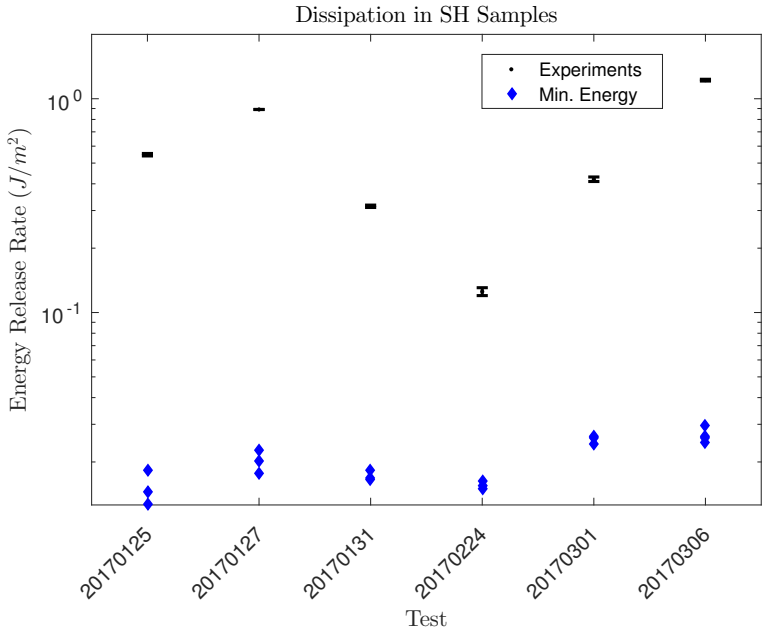


Figure 8.4: Minimum Cut G_c compared to actual experimental G_c in samples with buried SH

hypothesis was rejected (p=0.03), indicating that the difference between homogeneous and surface hoar results is statistically significant.

Surprisingly, the value of $G_{c(min)}$ does not vary significantly between homogeneous samples and surface hoar samples. All values of $G_{c(min)}$ were around $0.01 - 0.02\, J/m^2$.

Stress Concentrations: Effect on FPZ

Bazant et al [5] conclude that LEFM becomes an increasingly acceptable approximation in snow as the notch length increases. In LEFM, stress around the crack tip is $\sigma \propto a^{0.5}$ (Eq. 1.2). It is reasonable to think that by increasing notch size and concentrating stress, failure will more likely be confined to a small spatial zone of high stress.

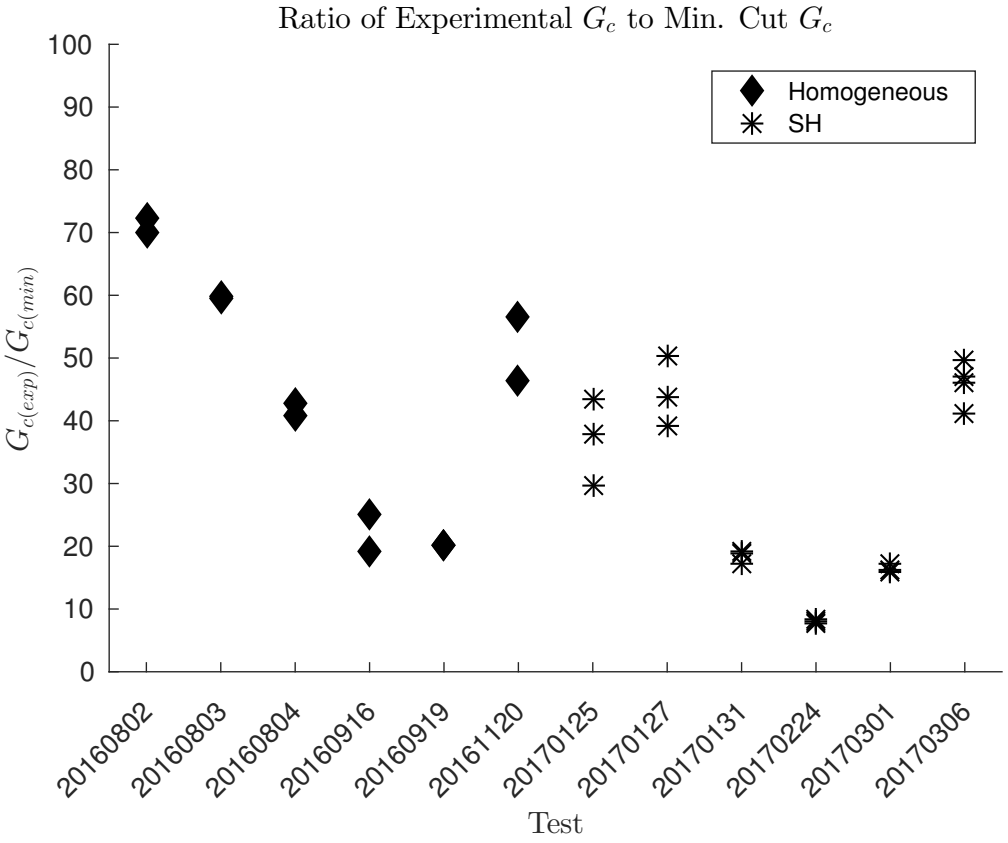


Figure 8.5: Ratio of experimental $G_{c(exp)}$ to minimum cut $G_{c(min)}$.

The stress concentration concept also fits with data found in the analysis of the minimum energy path. The minimum cut energy in surface hoar did not vary significantly from minimum cut energy in homogeneous samples. In this sense, surface hoar is not any "weaker" than rounded grains, yet samples with a slab-surface hoar *system* showed less damage.

However, weak layers are more compliant and less dense than rounded grains [89]. Hutchinson and Suo [40], in their work on fracture in laminates, offer an applicable explanation: elastic mismatch. They consider a crack at the interface of two elastic materials (Fig. 8.6) with elastic moduli E_1 and E_2.

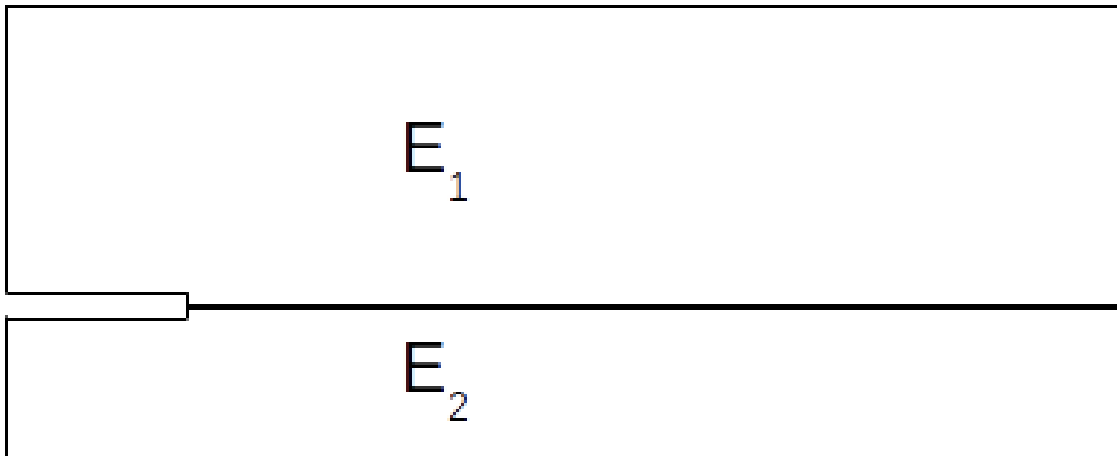


Figure 8.6: Interface crack in two elastic materials with different elastic moduli

These two materials are subject to an elastic mismatch parameter α:

$$\alpha = \frac{E_1 - E_2}{E_1 + E_2} \tag{8.9}$$

In this analysis, $\alpha \to 1$ if $E_1 \gg E_2$, $\alpha \to -1$ if $E_2 \gg E_1$, and $\alpha = 0$ if $E_1 = E_2$

For a shear crack at a bilayer interface,

$$G_c \propto 1 - \alpha \tag{8.10}$$

Consider a configuration in snow where the slab has modulus E_1 and a weak layer has modulus E_2. So, as $\alpha \to 1$, that is, as the ratio of slab modulus to weak layer modulus increases, energy release rate for a crack at the interface approaches 0. So, as the elastic mismatch increases, the stress intensity factor at the interface must increase. Even though the bonds in the "weak layer" are not weaker by the minimum cut analysis, the interface between weak layer and slab creates a high stress concentration, effectively weakening the structure. Heierli [34] made the assumption that $\alpha = 1$, although his model does not directly consider an elastic modulus for the weak layer.

Conceptually ,the concept of elastic mismatch driving the failure process in snow is readily observed in the field. Birkeland et al [8] have shown that both weak layer and slab properties affect fracture propagation. They refer to the ability of a slab to "communicate" fracture to the weak layer. They find that as a weak layer's time of burial increases, a denser slab is required to propagate fracture. Weak layer bonds strengthen over time as they sit buried beneath overburden pressure. Denser slabs tend to have higher elastic moduli. Therefore, the results of Birkeland et al can be interpreted to say that tougher weak layers require stiffer slabs for propagation. This concept is justified by the interface crack concepts discussed here, that energy release rate decreases as $\alpha \to 1$.

Schweizer et al [71] find similar results from a field study: As weak layer energy release rate increases, so does the elastic modulus of the slab. Again, this follows the same concept: That as the weak layer strengthens, the slab must stiffen in order to propagate fracture.

In a similar finding, van Herwijnen et al [85] find that slabs with a lower elastic modulus were less likely to be associated with full fracture propagation in propagation saw tests.

In Micro-CT specimens collected here, the layered approach was eschewed in favor of a more detailed microstructural analysis. At the scale of micro-CT scans, there were not clearly defined layers. However, there were density gradients. Weak layers were less dense and slabs were more dense

It is known that elastic modulus generally increases with density, and volume fraction of each numerical subsample is easily computed from CT data. Similar to how greater elastic mismatch decreases energy release rate in layered composites, it was hypothesized that specimens with a greater microstructural density mismatch would show more linear-elastic behavior and comparatively less damage. The volume fraction of each layer ("slice") in the CT reconstructions was computed. The microstructural density mismatch parameter ψ is proposed as a convenient microstructural parameter:

$$\psi = \frac{\rho_{max}}{\rho_{min}} \tag{8.11}$$

where ρ_{max} is the density of the densest crack face-parallel slice, and ρ_{min} is the density of the least dense crack-parallel slice in the numerical subsample. A perfectly uniform specimen would have $\psi = 1$, whereas a specimen with a dense slab overlying a non-dense weak layer would have $\psi \gg 1$.

The density mismatch parameter ψ was shown to correlate with energy release. As expected, a greater density mismatch was associated with decreased energy release rate, showing that ψ could be a useful microstructural parameter (Figure 8.7). The fit shows that higher values of ψ (higher density mismatch) yield energy release closer to $G_{c(min)}$.

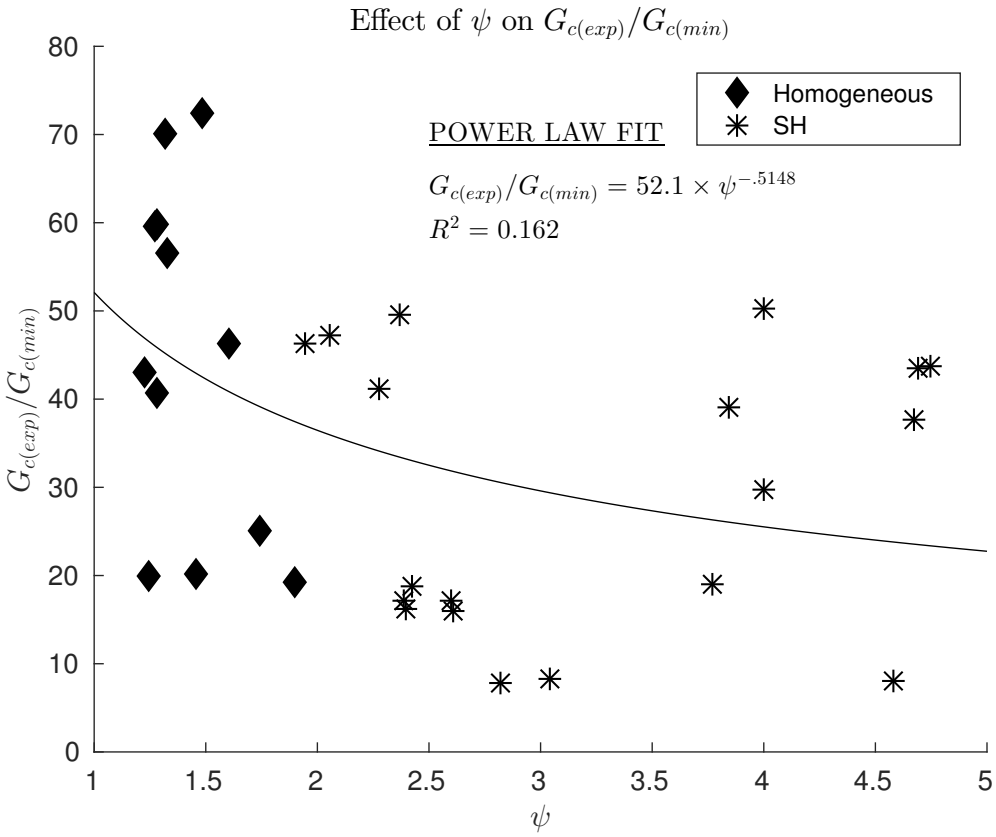


Figure 8.7: Ratio of actual G_c to $G_{c(min)}$ as a function of ψ.

CONCLUSIONS

Both the experimental and numerical results of this study show that damage accumulation is the dominant mechanism in the failure of snow, both in the slab and in the weak layer. While LEFM may be a usable approximation in many cases, especially with notched samples, it must be noted that the actual microstructural failure process is not linear elastic.

Experiments consistently showed pseudoplastic damage accumulation behavior: As samples were strained, load would increase, level off, and then drop. Microstructurally, this pseudoplasticity is caused by widespread bond breakage in the samples that is not necessarily constrained to the weak layer.

Discrete element modeling also showed pseudoplasticity as samples were loaded, although there was not always an obvious drop in force as the sample failed. Rather, the failure process appeared more gradual. This could be due to the small size of microstructural samples, which may be smaller than the FPZ in snow. In homogeneous snow, bond failure occurred throughout the volume of each sample. In snow with a buried weak layer, damage was more confined to the weak layer, although there were several instances where this was not the case and a compelling case is made for significant energy dissipation in the slab as well.

The minimum energy path computation showed interesting corroborating evidence for a widespread damage process. Because the minimum energy model consistently underestimated energy release rate by 1-2 orders of magnitude relative to laboratory experiments, it is concluded that damage outside a primary intergranular fracture path accounts for the vast majority of energy released during fracture. In homogeneous samples, a greater portion of energy is released outside the minimum

energy path than in layered samples, indicating that layered samples have a more limited FPZ and may be more closely approximated by LEFM.

From the results in Chapter 7, several consistent trends develop. First, energy release scales better by *volume* than by area, at least in homogeneous samples and possibly in layered samples as well. This is consistent with multiple published results that show increasing energy release with increasing sample size (e.g. [85], [71]). It also agrees nicely with Sigrist et al's observation [74] that fracture toughness (and therefore G_c) increases with sample size "possibly up to the scale of a slab avalanche." This work does not consider samples of that scale, but does not contradict Sigrist's conclusion either.

The scale observations also fit in nicely with what is known about avalanche initiation in the field. Indeed, a skier standing on a thick slab is less likely to trigger an avalanche than a skier standing on a thin slab. Thumlert and Jamieson [83] have measured the "stress bulb" created by a skier or snowmobiler moving on snow, and show that further down in the snowpack, the skier or rider induces less stress.

Another way to view this result is in terms of work done on the snowpack. A skier standing on the slope does work to deform the snow immediately underneath his or her skis. However, if the slab is thick, the work done by the skier is inadequate to deform the slab down to the weak layer and cause failure. Indeed, it seems reasonable that for the skier to do work on the weak layer and cause failure, he or she must cause bond failure through the entire depth of the slab in order to perturb the weak layer and initiate a propagating fracture.

Direction of Energy/Loading

From the minimum energy computation, it is seen that the theoretical minimum energy cost to fail a sample is very low. But, experiments and DEM show that

significantly more energy is required to actually fail a sample. As elastic mismatch and notch size increase, the work done on the snowpack is directed into a more confined spatial zone, increasing the propensity of the sample to display brittle failure. However, even in samples with a buried weak layer, there is often damage outside the weak layer. This suggests an interaction of weak layer and slab during the fracture process. The importance of weak layer/slab interaction is well established in field studies, and here that interaction is analyzed through microstructure.

Fracture in snow seems to be function of how microstructural elements (and in turn, their respective layers) work together to direct a load to a confined spatial zone and propagate fracture. It is not enough to have a weakness in the system, the system must transmit load or work/energy to that weakness in order to propagate failure in the snowpack. The more efficiently the slab/weak layer system transmits load to the weakness, the more closely the system approximates brittle failure. It was established that the ratio of experimental $G_{c(exp)}$ to minimum cut $G_{c(min)}$ correlates with how closely the system may approximate LEFM. The density mismatch parameter ψ was proposed as a microstructural parameter that predicts how closely fracture propagation mimics a brittle elastic regime.

All model and experimental results for G_c point consistently to the presence of damage accumulation. Analysis of the minimum cut, discrete element analysis, and laboratory experiments all showed the importance of damage accumulation. All values of G_c from all models and experiments are summarized in Figure 9.1. The DEM showed the importance of damage by physically recording all damage (bond breakage events) and demonstrating that damage is not always constrained to a planar-like fracture, even in samples with weak layers. The minimum energy path showed the dominance of damage mechanisms by severely under-predicting the amount of energy released relative to laboratory experiments. Taken together, these three different

approaches complement one another to highlight the importance of nonlinear failure processes in snow.

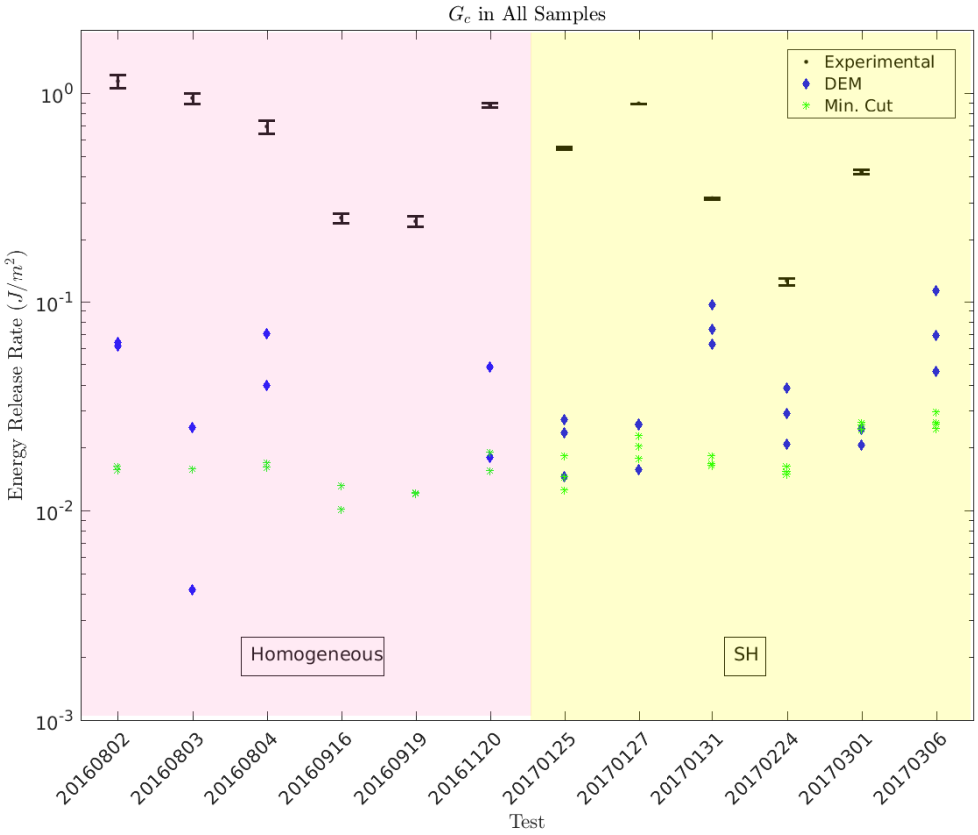


Figure 9.1: Summary of G_c for all tests and all models

Considerations for future models

Cohesive crack faces are a popular consideration in snow fracture models that were not considered here. Crack face friction was part of the DEM, but adhesion was not. It may be desirable in the future to consider adhesion and fast sintering processes that occur between ice grains during the fracture process.

This study also shows that the model performed better (lower NRMSE) on homogeneous, rounded-grain samples than on samples with buried surface hoar. This

may be due in part to the segmentation method chosen. The watershed segmentation method works best on spherical particles. While this may be acceptable in rounded grains, the flat, angular nature of surface hoar crystals makes them less suitable for watershed segmentation. A segmentation algorithm for snow that demarcates grain/bond boundaries by a criteria of local curvature and constriction has been proposed for snow [30], and is claimed to give a more realistic mechanical response in discrete element models.

REFERENCES CITED

[1] S. Amberger, M. Friedl, C. Goniva, S. Pirker, and C. Kloss. Approximation of objects by spheres for multisphere simulations in dem. In *European Congress on Computational Methods in Applied Sciences and Engineering.*, 2012.

[2] R. Ashton. A review and analysis of research on the testretest reliability of professional judgment. *Journal of Behavioral Decision Making*, 13(3):277–294, 2000.

[3] H. Bader and B. Salm. On the mechanics of snow slab release. *Cold Regions Science and Technology*, 17(3):287–300, 1990.

[4] Z. Bazant and J. Planas. Fracture and size effect. *Concrete and Other Quasibrittle Materials*, 16, 1998.

[5] Z. Bazant, G. Zi, and D. McClung. Size effect law and fracture mechanics of the triggering of dry snow slab avalanches. *Journal of Geophysical Research*, 2003.

[6] L. Berlyand, M. Rintoul, and S. Torquato. First-passage percolation, semi-directed bernoulli percolation and failure in brittle materials. *Journal of Statistical Physics*, 91(3-4):603–623, 1998.

[7] N. Biggs, E. Lloyd, and R. Wilson. *Graph Theory.* Clarendon Press, 1976.

[8] K. Birkeland, A. vanHerwijnen, E. Knoff, M. Staples, E. Bair, and R. Simenhois. The role of slabs and weak layers in fracture arrest. In *Proceedings of the 2014 International Snow Science Workshop*, 2014.

[9] K. W. Birkeland. Terminology and predominant processes associated with the formation of weak layers of near-surface faceted crystals in the mountain snowpack. *Arctic and Alpine Research*, 30(2):193–199, 1998.

[10] K. W. Birkeland, R. F. Johnson, and D. S. Schmidt. Near-surface faceted crystals formed by diurnal recrystallization: A case study of weak layer formation in the mountain snowpack and its contribution to snow avalanches. *Arctic and Alpine Research*, 30(2):200–204, 1998.

[11] J. Bones and E. Adams. Controlling crystal habit in a small scale snowmaker. In *International Snow Science Workshop*, 2009.

[12] C. Borstad and D. McClung. Numerical modeling of tensile fracture initiation and propagation in snow slabs using nonlocal damage mechanics. *Cold Regions Science and Technology*, 69(2-3):145–155, 2011.

[13] D. Broek. *ELementary Engineering Fracture Mechanics.* Martinus Nijhoff Publishers, 1982.

[14] Bruker MicroCT. *NRecon User Manual*, 2016.

[15] S. Colbeck. Growth of faceted crystals in a snow cover. Technical Report 29, Cold Regions Research and Engineering Laboratory, Hanover, N.H., 10 1982.

[16] S. Colbeck. A review of seasonal snow metamorphism. *Review of Geophysics and Space Physics*, 20(1):45–61, 1987.

[17] P. Cundall and O. Strack. A discrete numerical model for granular assemblies. *Geotechnique*, 29(1):47–65, 1979.

[18] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the Association for Computing Machinery*, 19(2):248–264, 1972.

[19] F. Flin, J. Brzoska, D. Coeurjolly, R. Pieritz, B. Lesaffre, C. Coleou, P. Lamboley, O. Teytaud, G. Vignoles, and J. Delesse. Adaptive estimation of normals and surface area for discrete 3-d objects: Application to snow binary data from x-ray tomography. *IEEE Transactions on Image Processing*, 14(5):585–596, 2005.

[20] L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[21] L. Ford and D. Fulkerson. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982.

[22] S. Freiman and J. Mecholsky. *The Fracture of Brittle Materials: Testing and Analysis*. John Wiley & Sons, 2012.

[23] T. Fukuzawa and H. Narita. An experimental study on the mechanical behavior of a depth hoar layer under shear stress. In *International Snow Science Workshop*, Breckenridge, CO, 1992.

[24] J. Gaume, A. vanHerwijnen, G. Chambon, N. Wever, and J. Schweizer. Snow fracture in relation to slab avalanche release: critical state for the onset of crack propagation. *The Cryosphere Discussions*, 2016.

[25] L. Gibson and M. Ashby. *Cellular Solids: Structure and Properties*. Cambridge University Press, 2 edition, 1999.

[26] D. Gleich. *Matlab BGL v2.1*, 4 2007.

[27] D. Goldsby and D. Kohlstedt. Grain boundary sliding in fine-grained ice i. *Scripta Materialia*, 1997.

[28] A. Griffith. The phenomena of rupture and flow in solids. *Philosophical Transactions of the Royal Society of London*, 1920.

[29] P. Hagenmuller. *Modlisation du comportement mcanique de la neige partir dimages microtomographiques*. PhD thesis, UNIVERSIT DE GRENOBLE, 2014.

[30] P. Hagenmuller, G. Chambon, F. Flin, S. Morin, and M. Naaim. Snow as a granular material: assessment of a new grain segmentation algorithm. *Granular Matter*, 2014.

[31] P. Hagenmuller, G. Chambon, and M. Naaim. Microstructure-based modeling of snow mechanics: a discrete element approach. *The Cryosphere*, 9(5):1969–1982, 2015.

[32] P. Hagenmuller, T. Theile, and M. Schneebeli. Numerical simulation of microstrural damage and tensile strength of snow. *Geophysical Research Letters*, 41(1):86–89, 2014.

[33] J. Heierli, P. Gumbsch, and D. Sherman. Anticrack-type fracture in brittle foam under compressive stress. *Scripta Materialia*, 67(1):96–99, 2012.

[34] J. Heierli, P. Gumbsch, and M. Zaiser. Anticrack nucleation as triggering mechanism for snow slab avalanches. *Science*, 321:240–243, July 2008.

[35] H. Hertz. *Miscellaneous Papers*. Macmillan and Co, Ltd., 1896. Authorized English translation.

[36] R. Hill. A general theory of uniqueness and stability in elastic-plastic solids. *Journal of the Mechanics and Physics of Solids*, 6(3):236 – 249, 1958.

[37] P. Hobbs. *Ice Physics*. Oxford University Press, 2010.

[38] E. Holm. Surface formation energy for intergranular fracture in two-dimensional polycrystals. *Journal of the American Ceramic Society*, 81(3):455–459, 1998.

[39] E. A. Holm and P. M. Duxbury. Three-dimensional materials science. *Scripta Materialia*, 54(6):1035–1040, 2006.

[40] J. Hutchinson and Z. Suo. *Mixed Mode Cracking in Layered Materials*. Academic Press Inc, 1992.

[41] G. Irwin, J. Kies, and H. Smith. Fracture strengths relative to onset and arrest of crack propagation. In *Proc. ASTM*, volume 58, pages 640–657, 1958.

[42] D. Jauffres, C. Martin, A. Lichtner, and L. Bordia. Simulation of the elastic properties of porous ceramics with realistic microstructures. *MOdelling and Simulation in materials science and engineering*, 2012.

[43] D. Jauffres, C. Martin, A. Lichtner, and R. Bordia. Simulation of the toughness of partially sintered ceramics with realistic microstructures. *Acta Materialia*, 2012.

[44] G. Jefferson, G. Haritos, and R. McMeeking. The elastic response of a cohesive aggregate–a discrete element model with coupled particle interaction. *Journal of the Mechanics and Physics of Solids*, 2002.

[45] Johnson. *Contact Mechanics*. Cambridge University Press, 1985.

[46] H. Kirchner, G. Michot, H. Narita, and T. Suzuki. Snow as a foam of ice: Plasticity, fracture and the brittle-to-ductile transition. *Philosophical Magazine A*, 81(9):2161–2181, 2001.

[47] H. Kirchner, G. Michot, and J. Schweizer. Fracture toughness of snow in shear and tension. *Scripta Materialia*, 46(6):425–429, 2002.

[48] H. Kirchner, G. Michot, and J. Schweizer. Fracture toughness of snow in shear under friction. *Physical Review E*, 66(2), 2002.

[49] H. Kirchner, G. Michot, and T. Suzuki. Fracture toughness of snow in tension. *Philosophical Magazine a-Physics of Condensed Matter Structure Defects and Mechanical Properties*, 80(5):1265–1272, 2000.

[50] E. N. Landis, E. N. Nagy, and D. T. Keane. Microstructure and fracture in three dimensions. *Engineering Fracture Mechanics*, 70(78):911 – 925, 2003.

[51] A. LeBaron and D. Miller. An energy-based microstructural constitutive model for fracture in snow. In *International Snow Science Workshop*, Banff, AB, Canada, 2014.

[52] X. Liu, C. Martin, G. Delette, and D. Bouvard. Elasticity and strength of partially sintered ceramics. *Journal of the Mechanics and Physics of Solids*, 2010.

[53] S. Luding. Introduction to discrete element methods. *European Journal of Environmental and Civil Engineering*, 2008.

[54] P. Mahajan and S. Joshi. Modeling of interfacial crack velocities in snow. *Cold Regions Science and Technology*, 51(23):98 – 111, 2008. International Snow Science Workshop (ISSW) 2006.

[55] P. Mahajan and S. Senthil. Cohesive element modeling of crack growth in a layered snowpack. *Cold Regions Science and Technology*, 40(12):111 – 122, 2004.

[56] D. McClung. Shear fracture precipitated by strain softening as a mechanism of dry slab avalanche release. *Journal of Geophysical Research: Solid Earth*, 1979.

[57] D. McClung. A physical theory of snow gliding. *Canadian Geotechnical Journal*, 18(1):86–94, 1981.

[58] D. McClung. Dry snow slab quasi-brittle fracture initiation and verification from field tests. *Journal of Geophysical Research*, 114(114):1–2, 2009.

[59] D. McClung. Mode ii fracture parameters of dry snow slab avalanche weak layers calculated from the cohesive crack model. *International Journal of Fracture*, 193(2):153 – 169, 2015.

[60] D. McClung and J. Schweizer. Fracture toughness of dry snow slab avalanches from field measurements. *Journal of Geophysical Research-Earth Surface*, 111(F4), 2006.

[61] D. M. McClung. Fracture energy applicable to dry snow slab avalanche release. *Geophysical Research Letters*, 34(2):n/a–n/a, 2007. L02503.

[62] D. M. McClung. The critical size of macroscopic imperfections in dry snow slab avalanche initiation. *Journal of Geophysical Research: Earth Surface*, 116(F3):n/a–n/a, 2011. F03003.

[63] F. Meyer. Topographic distance and watershed lines. *Signal Processing*, 1994.

[64] D. Miller, E. Adams, and R. Brown. A microstructural approach to predict dry snow metamorphism in generalized thermal conditions. *Cold Regions Science and Technology*, 37(3):213–226, 2003.

[65] H. Narita. Mechanical behavior and structure of snow under uniaxial tensile stress. *Journal of Glaciology*, 26(94):275–282, 1980.

[66] Palmer and Rice. The growth of slip ssurface in the progressive failure of over-consolidated clay. *Proceedings of the Royal Society of London*, 1973.

[67] D. Potyondy and P. Cundall. A bonded-particle model for rock. *International Journal of Rock Mechanics and Mining Sciences*, 41(8):1329 – 1364, 2004. Rock Mechanics Results from the Underground Research Laboratory, Canada.

[68] I. Reiweger, J. Schweizer, J. Dual, and H. J. Herrmann. Modelling snow failure with a fibre bundle model. *Journal of Glaciology*, 55(194):997–1002, 2009.

[69] C. Scapozza. *ENTWICKLUNG EINES DICHTE- UND TEMPERATURABHNGIGEN STOFFGESETZES ZUR BESCHREIBUNG DES VISKOELASTISCHEN VERHALTENS VON SCHNEE*. PhD thesis, EIDGENSSISCHEN TECHNISCHEN HOCHSCHULE ZRICH, 2004.

[70] P. Schaerer and D. McClung. *The Avalanche Handbook*. The Mountaineers Books, 2006.

[71] J. Schweizer, B. Reuter, A. vanHerwijnen, B. Richter, and J. Gaume. Temporal evolution of weak layer and slab properties in view of snow instability. *The Cryosphere Discussions*, 2016.

[72] J. Schweizer, A. van Herwijnen, and B. Reuter. Measurements of weak layer fracture energy. *Cold Regions Science and Technology*, 69(23):139 – 144, 2011. International Snow Science Workshop 2010 Lake Tahoe.

[73] I. Shames and F. Cozzarelli. *Elastic and Inelastic Stress Analysis*. Taylor & Francis, 1997.

[74] C. Sigrist, J. Schwiezer, H. Schindler, and J. Dual. On size and shape effects in snow fracture toughness measurements. *Cold Regions Science and Technology*, 43(1-2):24–35, 2005.

[75] C. Sigrist, J. Schwiezer, H. Schindler, and J. Dual. The energy release rate of mode ii fractures in layered snow samples. *International Journal of Fracture*, 139(3-4):461–475, 2006.

[76] V. Smilauer. *Yade Documentation 2nd ed*. The Yade Project, 2015. http://yade-dem.org/doc/.

[77] P. Soille. *Morphological Image Analysis*. Springer-Verlag, 1999.

[78] S. Somiya, editor. *Handbook of Advanced Ceramics*. Elsevier, 2013.

[79] W. Spychalski, K. Kurzydlowski, and B. Ralph. Computer study of inter-and intragranular surface cracks in brittle polycrystals. *Journal of the American Ceramic Society*, 49(1):45–53, 2002.

[80] B. Stanton. Visible bidirectional-reflectance measurements for rounded grain and surface hoar snow crystal morphologies. Master's thesis, Montana State University, 2013.

[81] K. Sternlof, J. Rudnicki, and D. Pollard. Anticrack inclusion model for compaction bands in sandstone. *Journal of Geophysical Research: Solid Earth*, 2005.

[82] H. Tada, P. Paris, and G. Irwin. *The Stress Analysis of Cracks Handbook*. American Society of Mechanical Engineers, 3rd edition, 2000.

[83] S. Thumlert and B. Jamieson. Stress measurements in the snow cover below localized dynamic loads. *Cold Regions Science and Technology*, 2014.

[84] B. Tremper. *Staying Alive in Avalanche Terrain*. The Mountaineers Books, Seattle, WA, 2 edition, 2008.

[85] A. vanHerwijnen, J. Gaume, E. Bair, B. Reuter, K. Birkeland, and J. Schweizer. Estimating the effective elastic modulus and specific fracture energy of snowpack layers from field experiments. *Journal of Glaciology*, 62(236):9971007, Dec 2016.

[86] A. vanHerwijnen and B. Jamieson. High-speed photography of fractures in weak snowpack layers. *Cold Regions Science and Technology*, 43(1):71–82, 2005.

[87] V. Šmilauer et al. Dem formulation. In *Yade Documentation 2nd ed*. The Yade Project, 2015. http://yade-dem.org/doc/.

[88] D. Walters and E. Adams. Micromechanical applications to radiation recrystallized snow. In *International Snow Science Workshop*, Banff, AB, Canada, 2014.

[89] D. J. Walters. *A MICROSTRUCTURAL INVESTIGATION OF RADIATION RECRYSTALLIZED SNOW LAYERS*. PhD thesis, Montana State University, August 2016.

[90] Y. Wei, R. Adamson, and J. Dempsey. Ice/metal interfaces: fracture energy and fractography. *Journal of Materials Science*, 31:943–947, 1996.

APPENDICES

APPENDIX A

IMAGE PROCESSING MATLAB CODE

Wrapper Script *yadebuild.m*

```
clear
close all

t=cputime
%% Enter parameters
filepath='RegSpheres/spheres'; %Directory of images PLUS any
    prefix on image names.
scale=1e-5; %Scale in meters per voxel
firstim=1 %Images must be numbered sequentially. This tells
    matlab where to start.
xrange=[1:20];  %X voxels to import from images
yrange=[1:20];
zrange=[1:40];
numsphrs=1; %Number of spheres per grain to use for approximation
imhmin=2; %Depth parameter for watershed segmentation
sphplot=0; %Plot spheres? 1 if yes, 0 if no
roverlap=0.15; %How much to overlap spheres in multisphere
    approximation?
ef=1.0; %Sphere enlargement factor
velo=[0.001,0,0];
lowspeed=[0.001,0,0]; %Slow strain speed (m/s) during fracture.
    For stability.
sigmaice=2.8e6;
tauice=1.25e6;
ogplot=0; %Plot original grains? 1 for yes, 0 for no
```

```
Gamma=2*0.2; %2*surface formation energy of ice
alpha=2 %Fracture shape factor, =2 for spherical particles.

%% Write file recording all test parameters

fileID1=fopen('/mnt/lustrefs/work/anthony.lebaron/yadeoutput/
    OPTBATCH25/simplespheres/2spheres/testparams.txt','w');
fprintf(fileID1,'%-s\n',strcat('Filepath: ',filepath));
fprintf(fileID1,'%-s\n',strcat('Scale (meters): ',num2str(scale))
    );
fprintf(fileID1,'%-s\n',strcat('First image in stack: ',num2str(
    firstim)));
fprintf(fileID1,'%-s\n',strcat('X Range: ',num2str(min(xrange)),'
    :',num2str(max(xrange))));
fprintf(fileID1,'%-s\n',strcat('Y Range: ',num2str(min(yrange)),'
    :',num2str(max(yrange))));
fprintf(fileID1,'%-s\n',strcat('Z Range: ',num2str(min(zrange)),'
    :',num2str(max(zrange))));
fprintf(fileID1,'%-s\n',strcat('Spheres per grain: ',num2str(
    numsphrs)));
fprintf(fileID1,'%-s\n',strcat('Watershed segmentation depth: ',
    num2str(imhmin)));
fprintf(fileID1,'%-s\n',strcat('Sphere overlap: ',num2str(
    roverlap)));
fprintf(fileID1,'%-s\n',strcat('Enlargement Factor: ',num2str(ef)
    ));
fprintf(fileID1,'%-s\n',strcat('Test Velocity: ',num2str(velo)));
```

```
fprintf(fileID1,'%-s\n',strcat('Slow Velocity: ',num2str(lowspeed
    )));
fprintf(fileID1,'%-s\n',strcat('Real sigma ice: ',num2str(
    sigmaice)));
fprintf(fileID1,'%-s\n',strcat('Real tau ice: ',num2str(tauice)))
    ;
fprintf(fileID1,'%-s\n',strcat('Gamma(2*surf. energy of ice): ',
    num2str(Gamma)));
fprintf(fileID1,'%-s\n',strcat('alpha (fracture particle shape
    parameter): ',num2str(alpha)));
fclose('all');

%% Run GrainsOut routine
[CCgrains,RealArea,RealLeng,bondnbr1,bondnbr2,sphmaster]=
    GrainsOut(filepath,scale,firstim,xrange,yrange,zrange,0,0,
    numsphrs,imhmin,sphplot,roverlap,ogplot);
%filepath, scale, first image to use, xrange, yrange, zrange,
    bondgrafx, grafxout, numsphrs, imhmim, sphplot, roverlap, ogplot

%% Create new Yade .py file, set view specification. This bilds
    yadeout.py, NOT efoptimize
fileID=fopen('/mnt/lustrefs/work/anthony.lebaron/yadeoutput/
    OPTBATCH25/simplespheres/2spheres/yade.py','w'); %Open/create
    file
fprintf(fileID,'%-s\n','#Parameters for user input');
```

```
fprintf(fileID,'%-s\n',strcat('velo=(',num2str(velo(1)),',',
    num2str(velo(2)),',',num2str(velo(3)),')␣#velocity␣of␣test␣(m/
    s)'));
fprintf(fileID,'%-s\n','axisxsxn=2␣#Axis␣about␣which␣we␣want␣to␣
    take␣x-sectional␣area.␣Usually␣2');
fprintf(fileID,'%-s\n','maxstr=0.5␣#Maximum␣sample␣strain␣(when␣
    to␣stop␣simulation)');
fprintf(fileID,'%-s\n','velmag=(velo[0]**2+velo[1]**2+velo[2]**2)
    **0.5␣#magnitude␣of␣velocity');
fprintf(fileID,'%-s\n',strcat('sampsize=(',num2str((max(xrange)-
    min(xrange)+1)*scale),',',num2str((max(yrange)-min(yrange)+1)*
    scale),',',num2str((max(zrange)-min(zrange)+1)*scale),')'));
fprintf(fileID,'%-s\n','␣');
fprintf(fileID,'%-s\n','nu=0.2␣#Poissons␣ratio␣of␣ice');
fprintf(fileID,'%-s\n','Eeq=2.74e8␣#youngs␣modulus␣of␣ice');
fprintf(fileID,'%-s\n','Ep=Eeq/(1-nu**2)␣#Eprime');
fprintf(fileID,'%-s\n','Epp=2*Eeq/((2-nu)*(1+nu))␣#E␣double␣prime
    ');
fprintf(fileID,'%-s\n','Gamma=2*0.2␣#2*surface␣form␣energy');
fprintf(fileID,'%-s\n','␣');
fprintf(fileID,'%-s\n','damping=0.05␣#Numerical␣damping␣
    coefficient');
fprintf(fileID,'%-s\n','saveinterval=1200␣#Time␣between␣file␣
    saves␣(seconds)');
fprintf(fileID,'%-s\n','ts=0.1␣#Fraction␣of␣critical␣timestep␣to␣
    use␣as␣timestep.␣0.01-0.1␣generally␣work␣well.');
fprintf(fileID,'%-s\n','␣');
```

```
fprintf(fileID,'%-s\n','#Graphics Options');
fprintf(fileID,'%-s\n','#yade.qt.Controller()');
fprintf(fileID,'%-s\n','#v=yade.qt.View()');
fprintf(fileID,'%-s\n','#v.viewDir=(0,-1,0)');
fprintf(fileID,'%-s\n','#v.upVector=(0,0,1)');

fprintf(fileID,'%-s\n',' ');
fprintf(fileID,'%-s\n','import numpy');
fprintf(fileID,'%-s\n','from yade import utils,plot');

%% Material definition for ice using CohFrictMat
fprintf(fileID,'%-s\n','O.materials.append(CohFrictMat(young=Eeq
   *0.01,poisson=1,density=9170,frictionAngle=0.01,isCohesive=
   True,normalCohesion=1e12,shearCohesion=1e12,etaRoll=10,
   momentRotationLaw=True,alphaKr=5,alphaKtw=5,label="ice"))');

%% Write grains (multispheres) into yade.py

sphmastcop=sphmaster; %Creat a copy of sphere master list to work
    with

for i = 1:max(sphmastcop(:,5)) %For all clumps in sphere master
   list

    if isempty(find(sphmastcop(:,5)==i)) %If there is no such
       clump, skip it
        continue
```

```
end

fprintf(fileID,'%-s\n','racol=utils.randomColor()'); %
    Generate random color for clump i


numstring=num2str(i,'%d'); %Clump ID# as string
clumpname=strcat('cl',numstring,'id,s',numstring,'id=0.bodies
    .appendClumped([');
fprintf(fileID,'%-s\n',clumpname); %Line to create new clump
    variable

firstsph=find(sphmastcop(:,5)==i,1); %row of first sphere in
    clump j
lastsph=find(sphmastcop(:,5)==i,1,'last');
for j=firstsph:lastsph; %For all spheres in clump i

    spcmd=strcat('sphere([',num2str(sphmastcop(j,1)),',',
        num2str(sphmastcop(j,2)),',',num2str(sphmastcop(j,3)),
        '],',num2str(sphmastcop(j,4)),',color=racol),'); %
        Assemble the sphere command text string
    fprintf(fileID,'\t%s\n',spcmd); %Tab, sphere command
        string,then new line

    clear spcmd

end
```

```
    fprintf(fileID,'\t%s\n',']))');
    sphmastcop(firstsph:lastsph,:)=[]; %Erase entry in sphmastcop
        . Will this make future iterations of for loop faster?
end


fclose('all'); %Closes the current file so we may append it
%% Write interactions into yadeints.py
fileID=fopen('/mnt/lustrefs/work/anthony.lebaron/yadeoutput/
    OPTBATCH25/simplespheres/2spheres/yadeints.py','w'); %Open/
    create file
for i = 1:length(RealArea) %For all bonds
    if RealLeng(i)==0 %Patch for occasional zero-size bonds
        RealLeng(i)=0.01; %Will create a very soft bond. Minimal
            effect on energy.
        RealArea(i)=1e-10; %Very small bond. Weak. Minimal effect
            on energy.
    end
    numstring=num2str(i-1); %String with bond ID#
    bondname=strcat('int',numstring,'=createInteraction(',num2str
        (bondnbr1(i)),',',num2str(bondnbr2(i)),')');
    fprintf(fileID,'%-s\n',bondname);

    fprintf(fileID,'%-s\n',strcat('int',numstring,'.phys.Fragile=
        True'));
```

```
        fprintf(fileID,'%-s\n',strcat('int',numstring,'.phys.
            normalAdhesion=2*(Ep*3.14159*Gamma)**0.5*',num2str(sqrt(
            RealArea(i)/3.14159)^1.5)));
        fprintf(fileID,'%-s\n',strcat('int',numstring,'.phys.
            shearAdhesion=2**1.5*(Epp*Gamma)**0.5*',num2str(sqrt(
            RealArea(i)/3.14159)^1.5)));
        fprintf(fileID,'%-s\n',strcat('int',numstring,'.phys.kn=Ep*',
            num2str(sqrt(RealArea(i)/3.14159))));
        fprintf(fileID,'%-s\n',strcat('int',numstring,'.phys.ks=Epp*'
            ,num2str(sqrt(RealArea(i)/3.14159)))); %Jauffres et al
            2012 shear stiffness
        fprintf(fileID,'%-s\n',strcat('int',numstring,'.phys.kr=0.25*
            Ep*',num2str(sqrt(RealArea(i)/3.14159)),'**3')); %bending
            (rotational) stiffness
        fprintf(fileID,'%-s\n',strcat('int',numstring,'.phys.ktw=0.5*
            Epp*',num2str(sqrt(RealArea(i)/3.14159)),'**3')); %
            Twisting stiffness
        fprintf(fileID,'%-s\n',strcat('int',numstring,'.phys.
            cohesionBroken=False'));
        fprintf(fileID,'%-s\n',strcat('int',numstring,'.phys.
            normalForce=(0,0,0)'));
        fprintf(fileID,'%-s\n',strcat('int',numstring,'.phys.
            shearForce=(0,0,0)'));
        %fprintf(fileID,'%-s\n',strcat('int',numstring,'.phys.unp=int
            ',numstring,'.geom.penetrationDepth'));
    end
    fclose('all'); %Closes the current file so we may append it
```

```
%% Add engines to the script
fclose('all'); %Closes the current file so we may append it

system('cat /mnt/lustrefs/work/anthony.lebaron/yadeoutput/
    OPTBATCH25/simplespheres/2spheres/yade.py scriptfrags/engines.
    py /mnt/lustrefs/work/anthony.lebaron/yadeoutput/OPTBATCH25/
    simplespheres/2spheres/yadeints.py scriptfrags/postscript.py >
    /mnt/lustrefs/work/anthony.lebaron/yadeoutput/OPTBATCH25/
    simplespheres/2spheres/yadeout.py')



%% Calculate Minimum Energy Path
%GrainsOut will produce a graph WITHOUT the top/bottom "master
    nodes"
%Add those nodes
%Then calc path

u=CCgrains.NumObjects+1; %Label of master start node
v=CCgrains.NumObjects+2; %Label of master end node

%Determine which grains are "From" but not "To," these connect to
     the top
%master node
Tops=setdiff(From,To); %Array of all grains that are in "From"
    but not "To"
To=[To,Tops];
```

```
for i=length(From)+1: length(To);
    From(i)=u; %Creates a new node labeled as 1 more than # of
        grains
    Area(i)=100000; %Arbitrarily high number that won't limit
        flow
end

%Determine which grains are in "To" but not in "From," these
    connect to
%bottom master node

Bottoms=setdiff(To,From); %Array of nodes that are in "to" but
    not "from"
From=[From,Bottoms]; %Append From to include bottom nodes
for i=length(To)+1:length(From)
    To(i)=v; %Creates a new node labeled as 2 more than # of
        grains
    Area(i)=100000;
end

%Sometimes zeros end up in From, To, and Area, remove them.
From(From==0)=[];
To(To==0)=[];
Area(Area==0)=[];

%Eliminate neighborless bonds.
i=1;
```

```
while i<=length(From)
    if From(i)==0
        From(i)=[];
        To(i)=[];
        Area(i)=[];
    elseif To(i)==0
        From(i)=[];
        To(i)=[];
        Area(i)=[];
    else
        i=i+1;
    end
end

%Assemble sparse graph matrix
Graph=sparse(From,To,Area,v,v);

%Calculate Minimum Cut using MatlabBGL
cd('matlab_bgl') %Change directory to use Matlab Boost Graph
    Laboratory
[flowval,cut,R,F] = max_flow(Graph,u,v);
cd ../
Cutenergy=flowval*scale^2*Gamma

%[topcut,botcut]=CutGraphics(cut,CCgrains,Graph,CCbonds);
```

```
proctime=cputime-t

fileID1=fopen('/mnt/lustrefs/work/anthony.lebaron/yadeoutput/
    OPTBATCH25/simplespheres/2spheres/mincut.txt','w');
fprintf(fileID1,'%-s\n',strcat('Minimum Cut Energy: ',num2str(
    Cutenergy)));
fclose('all');
```

grainsout.m subroutine

This is where the bulk of the image processing happens.

```
%% Starts with raw binary CT data and end up with array of grain
    voxels
%Basically the first ~20 lines of Scan2Cut.m
%20151010 Added subroutine to convert grains to multispheres
%Filepath includes beginning of filename, e.g. 'C:\Images\ImFile'
%Scale in meters per voxel e.g. 14.88e-6 for 14.88 micron
    resolution
%firstim is file number of


function [CCgrains,RealArea,RealLeng,bondnbr1,bondnbr2,sphmaster
    ]=GrainsOut(Filepath,scale,firstim,imsizex,imsizey,imsizez,
    bondgrafx,grafxout,numsphrs,imhmin,sphplot,roverlap,ogplot); %
    Filepath for images, scale (meters per pixel), image size as 3
     separate arrays specifying pixels in each dimension, 1 for
    bond/grain graphics, 1 for .stl and image stacks out,numsphrs
    is # of spheres per grain
```

```
%sphplot=1 to plot multi-colored clumps of spheres
%ogplot=1 to plot original grains on same plot as clumps
%failures=0; %Count the number of grains for which multisphere
    routine fails
%Eice=1e9; %Young's modulus of ice
%% Import Images

bw=BinStackImprt(Filepath,'%04d','.bmp',firstim,imsizex,imsizey,
    imsizez); %Builds binary array from image stack
%bw=1;

disp('Image Import complete!')
assignin('base','ccbw',bwconncomp(bw));
%% Perform Watershed Segmentation
[bonds,CCgrains,labelgrn,~,gracols]=WatershedSeg(bw,imhmin,0,
    ogplot); %Performs watershed segmentation on bw, 0/1 on bond-
    grain grafix, 0/1 on multicolor grain plot
disp('Segmentation Complete!')
stats=regionprops(CCgrains);
assignin('base','CCgrains',CCgrains)
%% Compute total bond areas and bond center-center lengths and
    bond neighbors
%CCbonds=bwconncomp(bonds);
%assignin('base','ccbonds',CCbonds);
%clear bonds
%Use GraphBUild2
```

```
[Graph,u,v,CCbonds,nbr1,nbr2,Area,normal,pixLeng]=GraphBuild2(
    bonds,CCgrains,labelgrn); %pixLeng is center-center bond
    lengths in voxels
RealLeng=pixLeng.*scale;
RealArea=Area.*scale^2;
assignin('base','CCbonds',CCbonds);
assignin('base','Graph',Graph)
assignin('base','Area',Area)
%% Cleanup bonds. Combine multiples. Sometimes multiple bonds
    connect the same two grains.
i=1; %Start counter
while i<length(nbr1) %Goes through every pair


    j=i+1;
    while j<=length(nbr1) %Compares nbr1(i) and nbr2(i) to ALL
        other pairs
        if nbr1(i)==nbr1(j) && nbr2(i)==nbr2(j) %If bond i
            connects same two grains as bond j, combine them
                RealArea(i)=RealArea(i)+RealArea(j);
                RealArea(j)=[];
                nbr1(j)=[];
                nbr2(j)=[];
                RealLeng(j)=[];
        else
            j=j+1; %If bond is not redundant, move on
        end
```

```
    end
    i=i+1; %Move on to next i

end
%% Create multispheres

%lastrow=0; %Counter for each row in the sphere master list
sphmaster=[]; %zeros(CCgrains.NumObjects*numsphrs,5);

colors=colormap; %Stores default colormap used in WatershedSeg
    routine

if sphplot==1
    figure
end
disp('Beginning parallel for loop to compute grain sphere clumps'
    )

parfor i=1:CCgrains.NumObjects %For all grains
    cmap=colors(ceil(gracols(i)*64),:); %Create a random color
        for this grain. For graphics purposes
    [sphtab]=multisphere(CCgrains.PixelIdxList{i},[CCgrains.
        ImageSize(1),CCgrains.ImageSize(2),CCgrains.ImageSize(3)],
        numsphrs,ogplot,cmap,sphplot,roverlap);
    %[sphtab,failure]=multisphere(CCgrains.PixelIdxList{i},[
        CCgrains.ImageSize(1),CCgrains.ImageSize(2),CCgrains.
        ImageSize(3)],numsphrs,ogplot,cmap,sphplot,roverlap,rsph);
```

```
        %Compute the sphere table for each grain. Pixel IDs,image
        size,spheres per grain, grafx=1 for isosurface on, =0 for
        off
%       if failure==1
%           failures=failures+1 %Count number of failed grains
%       else
          s(i).clumpnum=i;
          s(i).spheres=cat(2,sphtab,zeros([length(sphtab(:,1)),1])+
              i); %Add a 5th column with grain # to sphtab
%           for j=1:length(sphtab(:,1)) %For each sphere
%               sphmaster(lastrow+j,1)=sphtab(j,1); %Assemble
    master list of sphere centers and radii
%               sphmaster(lastrow+j,2)=sphtab(j,2);
%               sphmaster(lastrow+j,3)=sphtab(j,3);
%               sphmaster(lastrow+j,4)=sphtab(j,4);
%               sphmaster(lastrow+j,5)=i; %Column #5 is grain
    number
%           end
%           lastrow=lastrow+length(sphtab(:,1));
%     end
end
%for i = 1 : CCgrains.NumObjects %Assemble structured array into
    a normal array that can be read by following algorithms
    % i

     %a=s(i).spheres
     sphmaster=cat(1,s.spheres); %Combine s.spheres into 1 table
```

```
sphmaster(:,1:4)=sphmaster(:,1:4).*scale;
%end

if sphplot==1
    saveas(gcf,'sphereapprox.png')
    saveas(gcf,'sphereapprox.fig')
end
%failures

%% Clean up sphmaster to eliminate spheres of 0 radius
i=1;
while i <= length(sphmaster(:,1));
    if sphmaster(i,4)==0;
        sphmaster(i,:)=[];
    else
        i=i+1;
    end

end

%% Add a 6th column to sphmaster. This will compute the element #
    assigned to that sphere by Yade

for i =1:length(sphmaster(:,1))
    sphmaster(i,6)=i-1+(sphmaster(i,5)-1); %Spheres are numbered
        0,1,2,etc
end
```

```
%% For each bond i with grains l and m, identify the two closest
    spheres and connect them.
bondnbr1=zeros(length(nbr1),1); %Create empty arrays to store
    bond neighbor spheres
bondnbr1=zeros(length(nbr2),1);
assignin('base','CCbonds',CCbonds);
for i=1:length(RealArea) %for all bonds

    l=nbr1(i); %identify neighbor l
    lcent=stats(l).Centroid.*scale; %identify center of grain l (
        meters)
    m=nbr2(i); %Identify neighbor m
    mcent=stats(m).Centroid.*scale; %center of grain m
    sphcentsl=sphmaster(sphmaster(:,5)==l,1:4); %Identify sphere
        centroid locations and radii in grain l
    sphcentsm=sphmaster(sphmaster(:,5)==m,1:4); %identify sphere
        centroid locations and radii in grain m
    [sphl,sphm]=findclosphere(lcent,mcent,sphcentsl,sphcentsm); %
        Determine which sphere in l is closest to m and vice versa
    nbr1row=find(ismember(sphmaster(:,1:3),sphl,'rows')); %
        Returns row index of sphere in l to be bonded
    bondnbr1(i)=sphmaster(nbr1row(1),6);
    nbr2row=find(ismember(sphmaster(:,1:3),sphm,'rows')); %
        Returns row index of sphere in m to be bonded. Why does it
```

```
        sometimes return multiples? Putting multiple spheres in
        the same place??!
    bondnbr2(i)=sphmaster(nbr2row(1),6);
end

end %Ends function at top
```

GraphBuild2.m

```
%% Script to build graph from segmented images
function [Graph,u,v,CCbonds,From,To,Area,normal,pixLeng]=
    GraphBuild2(bonds,CCgrains,labelgrn)
%% Identify and label Discrete grains and bonds. Commented out b/
    c this is now done externally

Size=size(bonds) %Size of array in voxels

bonds(1:2,:,:)=0; %Six planes to erase bond voxels on edge of
    image. Prevents edge problems when analyzing neighborhoods.
    Could be improved?
bonds(:,1:2,:)=0;
bonds(:,:,1:2)=0;
bonds(Size(1)-1:Size(1),:,:)=0;
bonds(:,Size(2)-1:Size(2),:)=0;
bonds(:,:,Size(3)-1:Size(3))=0;

CCbonds=bwconncomp(bonds); %Lists voxels in each bond,
```

```
GrnStat=regionprops(CCgrains,'Centroid'); %Stores centroids of
    each grain

assignin('base','CCbondsDebug',CCbonds) %Writes a copy of CCbonds
     to workspace in case function doesn't finish

%% Identify neighbor grains of each bond and top/bottom
    relationship for flow directions
bondnum=1; %start a counter for the while loop

while bondnum<=CCbonds.NumObjects %For all bonds
     bondnum;
     [NbrGrains]=NbrFind(bondnum,labelgrn,Size,CCbonds);

     %At this point, we have the numerical "names" of each grain
         touching
     %bond i. We assume flow from high z to low z.

     if length(NbrGrains)>2 %If a bond touches 3 or 4 grains,
         there is more work to be done!
          %sprintf('multigrain routine triggered')
          bondnum;
          [NbrGrains,CCbonds]=multigrain(bondnum,CCbonds,labelgrn);
          %NbrGrains=NbrGrains(1:2); %Temporary patch- not the best
               idea
     end
```

```
    if length(NbrGrains)<2 %If a bond has 1 or 0 neighbors, it is
        a floater and should be eliminated.
        %sprintf('Bond detected with 0 or 1 neighbors')
        bondnum=bondnum+1;
        continue
    end

    GrainPair=NbrGrains; %Grain Neighbors must be whittled down
        to pairs...add later

    [Fromi,Toi]=FromTo(GrainPair,GrnStat); %Decides which grain
        is the "from" and which is the "to"



    From(bondnum)=Fromi; %Adds value to directed graph array
    To(bondnum)=Toi; %Adds value to directed graph array
    [Area(bondnum),normal(bondnum,:)]=PlaneFit(CCbonds.
        PixelIdxList{bondnum},CCbonds.ImageSize);


    %Area(bondnum)=length(CCbonds.PixelIdxList{bondnum}); %Values
        for bond area, set to # of voxels for testing/debugging.
        Refine later.
    bondnum=bondnum+1;
end
assignin('base','From',From)
```

```
assignin('base','To',To)



%% Center-center lengths
stats = regionprops(CCgrains); %Individual grain stats
pixLeng=zeros(CCbonds.NumObjects,1);
for i=1:CCbonds.NumObjects %For all bonds
    %i;
    if From(i)~=0
        if To(i)~=0
            stats(From(i)).Centroid;
            stats(To(i)).Centroid;
            pixLeng(i)=norm(stats(From(i)).Centroid-stats(To(i)).
                Centroid);
        end
    end
end

%% Add Top and Bottom Master Nodes
u=CCgrains.NumObjects+1; %Label of master start node
v=CCgrains.NumObjects+2; %Label of master end node
%
% %% Determine which grains are "From" but not "To," these
    connect to the top
% %master node
```

```
% Tops=setdiff(From,To); %Array of all grains that are in "From"
    but not "To"
% To=[To,Tops];
%
% for i=length(From)+1: length(To);
%     From(i)=u; %Creates a new node labeled as 1 more than # of
    grains
%     Area(i)=100000; %Arbitrarily high number that won't limit
    flow
% end
%
%% Determine which grains are in "To" but not in "From," these
    connect to
%%bottom master node
%
% Bottoms=setdiff(To,From); %Array of nodes that are in "to" but
    not "from"
% From=[From,Bottoms]; %Append From to include bottom nodes
% for i=length(To)+1:length(From)
%     To(i)=v; %Creates a new node labeled as 2 more than # of
    grains
%     Area(i)=100000;
% end
%
%%Sometimes zeros end up in From, To, and Area, remove them.
% From(From==0)=[];
% To(To==0)=[];
```

```
% Area(Area==0)=[];


%% Eliminate neighborless bonds.
i=1;
while i<=length(From)
    if From(i)==0
        From(i)=[];
        To(i)=[];
        Area(i)=[];
    elseif To(i)==0
        From(i)=[];
        To(i)=[];
        Area(i)=[];
    else
        i=i+1;
    end
end


%% Assemble sparse graph matrix
Graph=sparse(From,To,Area,v,v);
```

PlaneFit.m

For one bond, approximates bond as a plane and computes area using technique of Flin et al.

```
% Function to input bond voxels, output area of the best fit (
    least squares) plane
```

```
%Modified to run on simple arrays for C translation- NO cells
    because not C
%compatible

function [area,unorm]=PlaneFit(CCbondlist,ImSize) %CCbondlist is
    CCbonds.PixelIdxList{bondnum}, ImSize is CCbonds.ImageSize
%% Create an array the size of the original image that contains
    the voxels
% of the one bond of interest
%bondnum
Bond= false(ImSize);  %Preallocate zeros, logical array to save
    memory
for i=1:length(CCbondlist) %Plot bond of interest from PIxel Id
    List
     Bond(CCbondlist(i))=1;
end

%% Trim the array Bond so that the many planes of zeros are
    eliminated. To save
% memory. And time? 7/7/14 Scan2Cut seems to go three times as
    fast without
% this step...
BondSize=size(Bond); %1x3 array of size of bond array

% for x=BondSize(1):-1:1
%
%       if any(any(Bond(x,:,:)))==0
```

```
%            Bond(x,:,:)=[];
%        end
% end
%
% for y=BondSize(2):-1:1
%
%        if any(any(Bond(:,y,:)))==0
%            Bond(:,y,:)=[];
%        end
% end
%
% for z=BondSize(3):-1:1
%
%        if any(any(Bond(:,:,z)))==0
%            Bond(:,:,z)=[];
%        end
% end

%assignin('base','BondDebug',Bond);
%% Create X,Y,Z arrays of bond points

% n=0; %counter set to 0
% for x=1:length(Bond(:,1,1))
%        for y=1:length(Bond(1,:,1))
%            for z=1:length(Bond(1,1,:))
%                if Bond(x,y,z)==1 %Creating X,Y,Z column arrays
    with coords of bond voxels
```

```
%                       n=n+1;
%                       X(n)=x;
%                       Y(n)=y;
%                       Z(n)=z;
%                   end
%               end
%           end
% end
%
% X
% Y
% Z

%7/7/2014 Replace for loops (above) with find command
lins=find(Bond); %Outputs linear indices of bond voxel locations
[X, Y, Z]=ind2sub(BondSize,lins(:)); %Stores X Y Z array of bond
    vox


%% Find bond unit normal based on least squares
InvMat=[sum(X.*X),sum(X.*Y),sum(X);sum(X.*Y),sum(Y.*Y),sum(Y);sum
    (X),sum(Y),length(X)];%Matrix to be inverted: must test for
    singularity first

if max(Y)==min(Y) %If plane is vertical, plane equation function
    doesn't work
     area=length(Y); %If plane vertical, area is # of voxels
```

```
    unorm=[0 1 0];
elseif max(Z)==min(Z) %If bond falls in constant z value, same
    area=length(Z);
    unorm=[0 0 1];
elseif max(X)==min(X) %If bond falls in constant X value, same
    area=length(X);
    unorm=[1 0 0];
elseif det(InvMat)== 0
        sprintf('Plane Fitting Matrix is Singular. Skipping bond'
            )
        unorm=[0 0 1];
        area=0;
else  %If plane is not neatly vertical/horizontal, bring out the
   big guns... Best Fit it!
    %Find plane A,B,C for equation z=Ax+By+C

    ABC=InvMat\[sum(X.*Z);sum(Y.*Z);sum(Z)]; %Least squares
        equation for a plane
    %Unit Normal
    normal=[ABC(1),ABC(2),-1]; %A normal to the plane
    unorm=normal/norm(normal); %Unit normal to the plane

%% Thin the surface

    surface=Bond; %Surface array; some voxels will be eliminated
```

```
    if max(abs(unorm))==abs(unorm(1)) %If max plane normal is in
        x direction
        surface=sum(Bond(1:length(Bond(:,1,1)),:,:));
    elseif max(abs(unorm))==abs(unorm(2)) %If max plane normal is
        in y direction
        surface=sum(Bond(:,1:length(Bond(1,:,1)),:));
    elseif max(abs(unorm))==abs(unorm(3)) %If max plane normal is
        in z direction
        %sprintf('Max normal in Z')
        surface=sum(Bond(:,:,1:length(Bond(1,1,:)),:));
    else
        sprintf('surface thinning gone awry')
        %bondnum
    end
%assignin('base','unormdebug',unorm)
%assignin('base','surfacedebug',surface)


    %Now we have a 3D array of surface voxels (essentially a
        thinned
    %surface to work with Flin's algorithm), the intermediate
        projection
    surfvox=nnz(surface); %# of surface voxels is equal to number
        of nonzero elements in surface array
    %In the future, could only count surface voxels rather than
        create
    %new array, but for now will keep array for visualization.
```

```
        ratio=1/max(abs(unorm)); %Ratio of "true" surface area to
            area projected onto functional plane (X,Y or Z)
        area=surfvox*ratio;


end


end %Associated with invoking function
```

APPENDIX B

MINIMUM ENERGY CODE

GraphBuild2.m

MATLAB Function to

```
%% Script to build graph from segmented images
function [Graph,u,v,CCbonds,From,To,Area,normal,pixLeng]=
    GraphBuild2(bonds,CCgrains,labelgrn)
%% Identify and label Discrete grains and bonds. Commented out b/
    c this is now done externally

Size=size(bonds) %Size of array in voxels

bonds(1:2,:,:)=0; %Six planes to erase bond voxels on edge of
    image. Prevents edge problems when analyzing neighborhoods.
    Could be improved?
bonds(:,1:2,:)=0;
bonds(:,:,1:2)=0;
bonds(Size(1)-1:Size(1),:,:)=0;
bonds(:,Size(2)-1:Size(2),:)=0;
bonds(:,:,Size(3)-1:Size(3))=0;

CCbonds=bwconncomp(bonds); %Lists voxels in each bond,
GrnStat=regionprops(CCgrains,'Centroid'); %Stores centroids of
    each grain

assignin('base','CCbondsDebug',CCbonds) %Writes a copy of CCbonds
     to workspace in case function doesn't finish
```

```
%% Identify neighbor grains of each bond and top/bottom
    relationship for flow directions
bondnum=1; %start a counter for the while loop

while bondnum<=CCbonds.NumObjects %For all bonds
    bondnum;
    [NbrGrains]=NbrFind(bondnum,labelgrn,Size,CCbonds);

    %At this point, we have the numerical "names" of each grain
        touching
    %bond i. We assume flow from high z to low z.

    if length(NbrGrains)>2 %If a bond touches 3 or 4 grains,
        there is more work to be done!
        %sprintf('multigrain routine triggered')
        bondnum;
        [NbrGrains,CCbonds]=multigrain(bondnum,CCbonds,labelgrn);
        %NbrGrains=NbrGrains(1:2); %Temporary patch- not the best
            idea
    end

    if length(NbrGrains)<2 %If a bond has 1 or 0 neighbors, it is
        a floater and should be eliminated...WHY WOULD THIS
       HAPPEN?!?!?! Probably due to trimming...
        %sprintf('Bond detected with 0 or 1 neighbors')
        bondnum=bondnum+1;
        continue
```

```
    end

    GrainPair=NbrGrains; %Grain Neighbors must be whittled down
        to pairs...add later

    [Fromi,Toi]=FromTo(GrainPair,GrnStat); %Decides which grain
        is the "from" and which is the "to"



    From(bondnum)=Fromi; %Adds value to directed graph array
    To(bondnum)=Toi; %Adds value to directed graph array
    [Area(bondnum),normal(bondnum,:)]=PlaneFit(CCbonds.
        PixelIdxList{bondnum},CCbonds.ImageSize);


    %Area(bondnum)=length(CCbonds.PixelIdxList{bondnum}); %Values
         for bond area, set to # of voxels for testing/debugging.
        Refine later.
    bondnum=bondnum+1;
end
assignin('base','From',From)
assignin('base','To',To)
```

```
%% Center-center lengths
stats = regionprops(CCgrains); %Individual grain stats
pixLeng=zeros(CCbonds.NumObjects,1);
for i=1:CCbonds.NumObjects %For all bonds
    %i;
    if From(i)~=0
        if To(i)~=0
            stats(From(i)).Centroid;
            stats(To(i)).Centroid;
            pixLeng(i)=norm(stats(From(i)).Centroid-stats(To(i)).
                Centroid);
        end
    end
end

%% Add Top and Bottom Master Nodes
u=CCgrains.NumObjects+1; %Label of master start node
v=CCgrains.NumObjects+2; %Label of master end node
%
% %% Determine which grains are "From" but not "To," these
    connect to the top
% %master node
% Tops=setdiff(From,To); %Array of all grains that are in "From"
    but not "To"
% To=[To,Tops];
%
% for i=length(From)+1: length(To);
```

```
%       From(i)=u; %Creates a new node labeled as 1 more than # of
    grains
%       Area(i)=100000; %Arbitrarily high number that won't limit
    flow
% end
%
%% Determine which grains are in "To" but not in "From," these
    connect to
%%bottom master node
%
% Bottoms=setdiff(To,From); %Array of nodes that are in "to" but
    not "from"
% From=[From,Bottoms]; %Append From to include bottom nodes
% for i=length(To)+1:length(From)
%       To(i)=v; %Creates a new node labeled as 2 more than # of
    grains
%       Area(i)=100000;
% end
%
% %Sometimes zeros end up in From, To, and Area, remove them.
% From(From==0)=[];
% To(To==0)=[];
% Area(Area==0)=[];

%% Eliminate neighborless bonds.
i=1;
while i<=length(From)
```

```
        if From(i)==0
            From(i)=[];
            To(i)=[];
            Area(i)=[];
        elseif To(i)==0
            From(i)=[];
            To(i)=[];
            Area(i)=[];
        else
            i=i+1;
        end
    end

    %% Assemble sparse graph matrix
    Graph=sparse(From,To,Area,v,v);
```

APPENDIX C

SAMPLE YADE CODE

This is the python script *yadeout.py* file used in the simple two-sphere model example in Chapter 6.

```
## This is sample code for the very simple case of two spheres
    bonded to one another.
## It shows how materials, test parameters, spheres, clumps and
    bonds are added to the simulation

#Parameters for user input
velo=(0.00001,0,0) #velocity of test (m/s)
axisxsxn=2 #Axis about which we want to take x-sectional area.
    Usually 2
maxstr=0.5 #Maximum sample strain (when to stop simulation)
velmag=(velo[0]**2+velo[1]**2+velo[2]**2)**0.5 #magnitude of
    velocity
sampsize=(0.0002,0.0002,0.0004)

nu=0.2 #Poissons ratio of ice
Eeq=2.74e8 #youngs modulus of ice
Ep=Eeq/(1-nu**2) #E*
Epp=2*Eeq/((2-nu)*(1+nu)) #E**
Gamma=2*0.2 #2* surface form energy

damping=0.05 #Numerical damping coefficient
saveinterval=5 #Time between file saves (seconds)
ts=0.1 #Fraction of critical timestep to use as timestep.
    0.01-0.1 generally work well.
```

```
#Graphics Options (Comment out to run on cluster)
yade.qt.Controller()
v=yade.qt.View()
v.viewDir=(0,-1,0)
v.upVector=(0,0,1)

import numpy #Loads numerical python package for use with python
    routines (at end of file)
from yade import utils,plot #Loads yade utilities and plot
    packages
O.materials.append(CohFrictMat(young=Eeq*0.01,poisson=1,density
    =9170,frictionAngle=0.01,isCohesive=True,normalCohesion=1e12,
    shearCohesion=1e12,etaRoll=10,momentRotationLaw=True,alphaKr
    =5,alphaKtw=5,label="ice")) #Create the default material. All
    bodies created will be of this ice material.

## Create clump 1, consisting of one sphere ##
racol=utils.randomColor() #Make each clump its own random color
cl1id,s1id=O.bodies.appendClumped([
        sphere([0.00011,0.00011,0.00011],0.0001,color=racol), #[x
            ,y,z coordinate of center],radius,color of sphere (
            meters)
        #If there were more spheres in this clump, they would go
            here
        ])

## Create clump 2, consisting of one sphere ##
```

```
racol=utils.randomColor()
cl2id,s2id=O.bodies.appendClumped([
        sphere([0.00011,0.00011,0.00031],0.0001,color=racol),
        ])

v.center(median=True) #Center viewing window to fit all bodies (
    This line must be commented out to run on the cluster since
    cluster has no GUI)

#############
## ENGINES ##
#############
O.trackEnergy=True


O.engines=[
    ForceResetter(), #Resets forces at every step. See Yade
        documentation
    InsertionSortCollider([Bo1_Sphere_Aabb(label='bo1s')]), #Use '
        axis-aligned bounding box' technique for contact detection.
         For details, see Yade documentation
    InteractionLoop(
    [Ig2_Sphere_Sphere_ScGeom6D(label='ig2ss')], #Loads engine for
         spherical geometries (Yade can work with other shapes, I
        chose spheres)
```

```
    [Ip2_CohFrictMat_CohFrictMat_CohFrictPhys(setCohesionNow=False
        ,label="cohesiveIp")], #Use Cohesive Frictional Material
        physics engine
    [Law2_ScGeom6D_CohFrictPhys_CohesionMoment(useIncrementalForm=
        True,label='law2')]
    ),
    NewtonIntegrator(damping=damping), #Applies forces and updates
         positions. Applies numerical damping routine.
    SnapshotEngine(iterPeriod=50000,fileBase='sheartest-',label='
        snapshooter',firstIterRun=1), #Save a screenshot every
        50,000 timesteps
    PyRunner(command='brokenbonds,intactlist=AutoData(numint,
        brokenbonds,intactlist)',iterPeriod=1000), #Runs AutoData
        function every 1000 timesteps to record state of test
    PyRunner(command='AutoSave(brokenbonds)',realPeriod=
        saveinterval) #Saves all data from AutoData every
        saveinterval seconds. 15 seconds for this simple example,
        usually 1200 seconds (20 minutes) on larger models
]

## 1 step to initialize network ##
for i in O.bodies: #For all bodies
    i.dynamic=False #Hold all bodies (spheres) in non-dynamic (0
        acceleration) state until bonds are applied
for i in O.interactions: #For all interactions
```

```
    O.interactions.erase(i.id1,i.id2) #Erases any interactions
        that the software may have assigned by default.
        Precautionary measure.


## Now we bond our clumps together
## by creating interactions with properties based on bond size
## Bond size was calculated using Flin et al's method

int0=createInteraction(2,0) #Create an interaction (bond) between
    sphere 2 and sphere 0. Sphere 0 is a member of clump 1 and
int0.phys.Fragile=True #Setting attribute phys.Fragile makes this
    bond brittle elastic
int0.phys.normalAdhesion=2*(Ep*3.14159*Gamma)**0.5*1.3401e-08 #
   Normal adhesion (strength) assigned
int0.phys.shearAdhesion=2**1.5*(Epp*Gamma)**0.5*1.3401e-08 #Shear
    strength assigned
int0.phys.kn=Ep*5.6419e-06 #Normal stiffness of bond
int0.phys.ks=Epp*5.6419e-06 #Shear stiffness
int0.phys.kr=0.25*Ep*5.6419e-06**3 #Bending stiffness
int0.phys.ktw=0.5*Epp*5.6419e-06**3 #Twisting Stiffness
int0.phys.cohesionBroken=False #Sets the initial state of the
   bond to cohesive
int0.phys.normalForce=(0,0,0) #By manually setting forces=0, this
    initial geometry becomes the equilibrium state of the bond
int0.phys.shearForce=(0,0,0)
```

```
## Put network in equilibrium ##
# When manually specifying bond properties, YADE also requires
    explicit setting of equilibrium state
numint=0
for i in O.interactions: #For all interactions
    i.phys.unp=i.geom.penetrationDepth #Sets the current
        penetration depth as equilibrium state
    i.phys.normalForce=(0,0,0)
    i.phys.cohesionDisablesFriction=True #Disables friction for
        cohesive bonds.
    numint+=1
print(numint) #Prints the initial # of bonds in the system
#DO NOT ADJUST numint after initial finding. It must stay
    constant for function AutoData to work

for x in range(1,50): #50 steps to remove any non-equilibrium
    forces
    O.step()
    for i in O.interactions:
        i.phys.unp=i.geom.penetrationDepth #Sets the current
            penetration depth as equilibrium state
        i.phys.normalForce=(0,0,0)

for i in O.bodies: #Now that we're definitely in equilibrium,
    allow the simulation to be dynamic (permit nonzero
    accelerations)
    i.dynamic=True
```

```
## Export master .txt file with initial positions of bonds ##
bondmaster=[0,0,0,0,0] #columns are id1,id2,x,y,z
for i in O.interactions:
    currentbond=[i.id1,i.id2,i.geom.contactPoint[0],i.geom.
        contactPoint[1],i.geom.contactPoint[2]]
    bondmaster=numpy.vstack((bondmaster,currentbond))
numpy.savetxt('bondmaster.txt',bondmaster,delimiter=',',fmt=['%i'
    ,'%i','%.6E','%.6E','%.6E'])


################################################
## BOUNDARY CONDITIONS AND APPLIED VELOCITY ##
################################################

O.dt=ts*utils.PWaveTimeStep() #Set the timestep as a coefficient
    of critical timestep. utils.PWaveTimeStep() is a function that
     calculates system critical timestep.
bb=uniaxialTestFeatures(axis=axisxsxn) #From YADE example
    uniaxial.py
negIds,posIds,axis,crossSectionArea=bb['negIds'],bb['posIds'],bb[
    'axis'],bb['area'] #Handy functions that find which spheres
    are at base and top (for application of BC's)
negclumpMembs=[]
for x in range(1,50): #50 steps for smoother acceleration as
    velocity is applied
    for i in negIds:
```

```
        clnum=O.bodies[i].clumpId #Identifies the clump this
            sphere is a member of
        O.bodies[clnum].state.blockedDOFs='xyzXYZ' #Clumps in
            negId are fixed
        negmembsi=O.bodies[clnum].shape.members.keys() #all
            members in the same clump as sphere i
        for j in negmembsi:
            try:
                negclumpMembs.index(j) #search negclumpMembs to
                    see if this sphere is already in list

            except ValueError: #If not already in list
                negclumpMembs.append(j) #add to negclumpMembs

posclumpMembs=[]     #Blank list of members of positive clumps
    .
for i in posIds:
    atopsphere=i
    clnum=O.bodies[i].clumpId #Identifies the clump this
        sphere is a member of
    O.bodies[clnum].state.vel=tuple(veloc*x/50 for veloc in
        velo) #Slowly increase velocity of boundary elements
        for smoother results
    O.bodies[clnum].state.blockedDOFs='xyzXYZ' #Clumps in
        posId are fixed for simple shear
    posmembsi=O.bodies[clnum].shape.members.keys() #all
        members in the same clump as sphere i
```

```
        for j in posmembsi:
            try:
                posclumpMembs.index(j) #search posclumpMembs to
                    see if this sphere is already in list

            except ValueError: #If not already in list
                posclumpMembs.append(j) #add to posclumpMembs
    O.step()

for i in negIds:
    clnum=O.bodies[i].clumpId #Identifies the clump this sphere
        is a member of
    O.bodies[clnum].state.blockedDOFs='xyzXYZ' #Clumps in negId
        are fixed
    negmembsi=O.bodies[clnum].shape.members.keys() #all members
        in the same clump as sphere i
    for j in negmembsi:
        try:
            negclumpMembs.index(j) #search negclumpMembs to see
                if this sphere is already in list

        except ValueError: #If not already in list
            negclumpMembs.append(j) #add to negclumpMembs

posclumpMembs=[]     #Blank list of members of positive clumps.
for i in posIds:
    atopsphere=i
```

```
    clnum=O.bodies[i].clumpId #Identifies the clump this sphere
        is a member of
    O.bodies[clnum].state.vel=velo #Apply fixed velocity to upper
         clumps
    O.bodies[clnum].state.blockedDOFs='xyzXYZ' #Clumps in posId
        are fixed for simple shear
    posmembsi=O.bodies[clnum].shape.members.keys() #all members
        in the same clump as sphere i
    for j in posmembsi:
        try:
            posclumpMembs.index(j) #search posclumpMembs to see
                if this sphere is already in list

        except ValueError: #If not already in list
            posclumpMembs.append(j) #add to posclumpMembs


##########################
## PYRUNNER ROUTINE(S) ##
##########################

brokenbonds=numpy.array([0,0,0,0],ndmin=2) #Columns: bondname,id1
    ,id2,O.time (simulation time). Empty array to be appended.
intactlist=range(0,numint) #List of intact cohesive bonds.
    Updated in function AutoData to track which bonds are broken.
    Initially, all bonds are intact
```

```
def AutoSave(brokenbonds): #Saves data every saveinterval seconds
    plot.saveDataTxt('loadhist.txt',vars=('d','F','time','
        cohbonds','strNrg','KE','normEnergy','shearEnergy','
        bendEnergy','twistEnergy','compTime','nvDamp'))
    numpy.savetxt('brokenbonds.txt',brokenbonds,delimiter=',',fmt
        =['%i','%i','%i','%.6E'])
    starttime=O.realtime #reset counter
    saveVars('pyVars',atopsphere=atopsphere,posclumpMembs=
        posclumpMembs,sampsize=sampsize,maxstr=maxstr)  #save
        relevant variables
    O.save('savedsim.yade.gz') #Save simulation. These files can
        be large (50 MB) so don't use unless you think you'll need
         it.
    #if compTime > 84000:      #Stops the simulation after 23.5
        hours. This prevents SLURM from killing the job in the
        middle of a save at 24 hours
        #O.pause()
        #exit()


def AutoData(numint,brokenbonds,intactlist):#Plotting routine
    based on https://answers.launchpad.net/yade/+question/248562
    #Also counts broken bonds
    #bondfile=open('bondbreak.csv','a') #'a' opens the file for
        appending
```

```
#del brokenbonds
#brokenbonds
for i in intactlist: #For all bonds not already known to be
    broken, we test to see if they are broken now
     bondname='int'+str(i)
     #print bondname
     try:

          if eval(bondname).phys.cohesionBroken: #if the bond
              has attribute cohesionBroken

               print "cohesionBroken detected"
               recentbroken=[i,eval(bondname).id1,eval(bondname)
                   .id2,O.time]
               brokenbonds=numpy.vstack((brokenbonds,
                   recentbroken)) #Appends the latest broken bond
                    to the array for export
               intactlist.remove(i) #Remove bond i from the list
                    of intact bonds


     except AttributeError: #If bond does not have
         cohesionBroken option, then also not a cohesive bond
         anymore
          print "Attribute Error"
          print 'bond number'+bondname
          recentbroken=[i,eval(bondname).id1,eval(bondname).id2
              ,O.time]
```

```
            brokenbonds=numpy.vstack((brokenbonds,recentbroken))
                #Appends the latest broken bond to the array for
                export
            intactlist.remove(i) #Remove bond i from the list of
                intact bonds

cohbonds=len(intactlist) #Number of cohesive bonds
normEnergy=law2.normElastEnergy() #Elastic Energy
shearEnergy=law2.shearElastEnergy()
bendEnergy=law2.bendingElastEnergy()
twistEnergy=law2.twistElastEnergy()


d=O.time*velmag #O.bodies[atopsphere].state.pos[0]-O.bodies[
    atopsphere].state.refPos[0] #X Displacement of upper
    clumps
plot.addData(
d=d,
F=utils.sumForces(posclumpMembs,(-1,0,0)),#utils.
    sumFacetNormalForces([tri3])+utils.sumFacetNormalForces([
    tri4]), #sum facet normal forces on tri1 axis 2 (z axis)
#F=utils.sumForces([1],(1,0,0)), #Use this line with the
    simple 2 spheres
time=O.time,
cohbonds=cohbonds,
normEnergy=normEnergy,
shearEnergy=shearEnergy,
```

```
        bendEnergy=bendEnergy,
        twistEnergy=twistEnergy,
        strNrg=normEnergy+shearEnergy+bendEnergy+twistEnergy,
        KE=O.energy['kinetic'],
        compTime=O.realtime,
        nvDamp=O.energy['nonviscDamp']
        )
        if d/sampsize[0] > maxstr:      #(abs(d)/sampsize[0])>maxstr:
            #after 20% strain stop and write output file
            O.pause()
            plot.saveDataTxt('loadhist.txt',vars=('d','F','time','
                cohbonds','strNrg','KE','normEnergy','shearEnergy','
                bendEnergy','twistEnergy','compTime','nvDamp'))
            exit()
        return brokenbonds,intactlist
plot.plots={'d':('F')}
plot.plot()

## Start it up! ##
#O.run() #These two lines should be uncommented to make code run
    automatically on cluster. For running on my PC, commenting
    these lines gives me manual start/pause controls of simulation
    .
#O.wait()
```

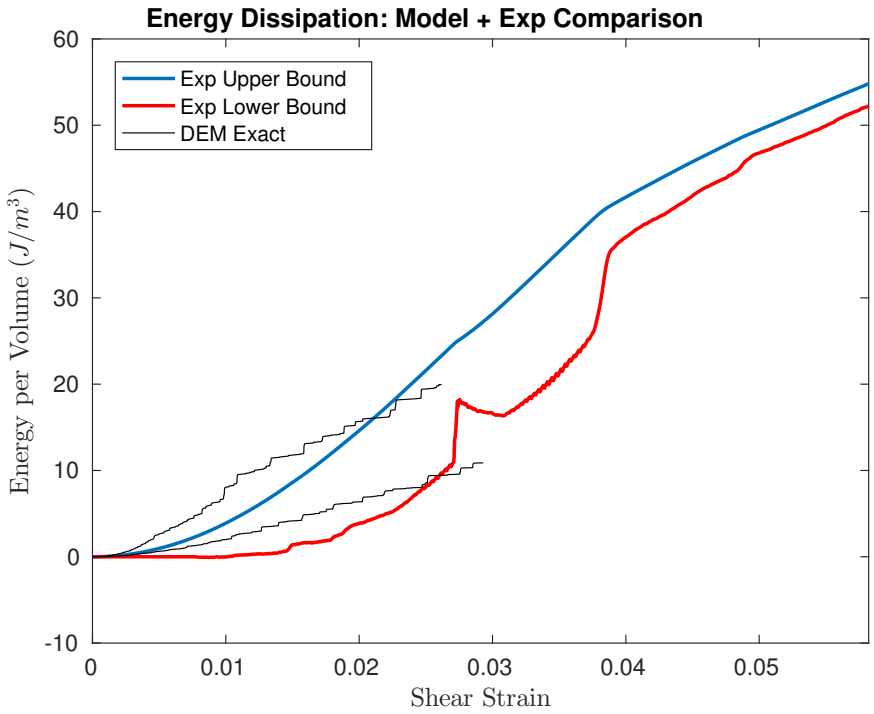
APPENDIX D

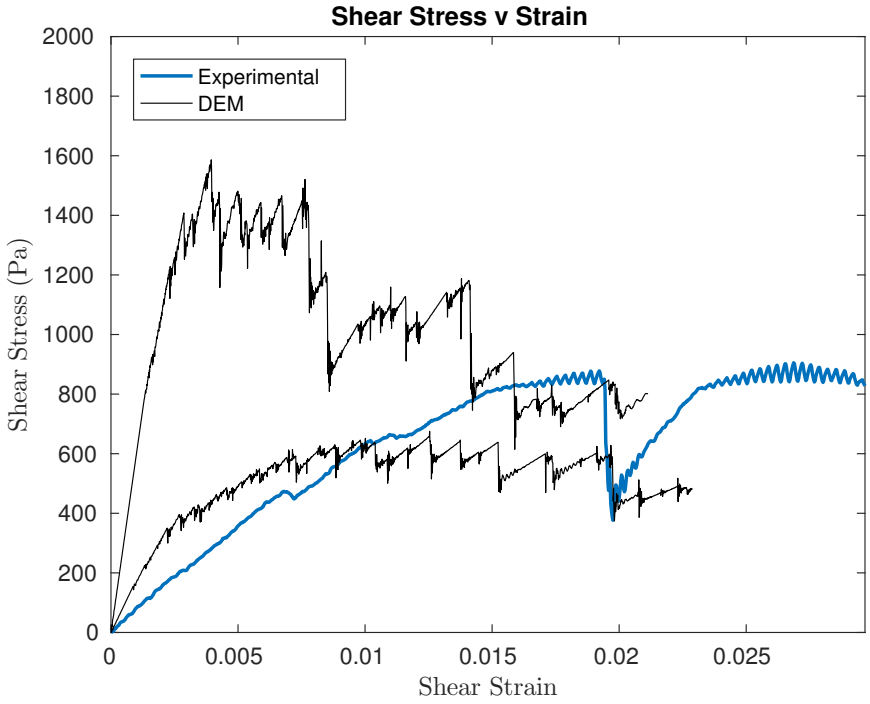
ALL RESULTS

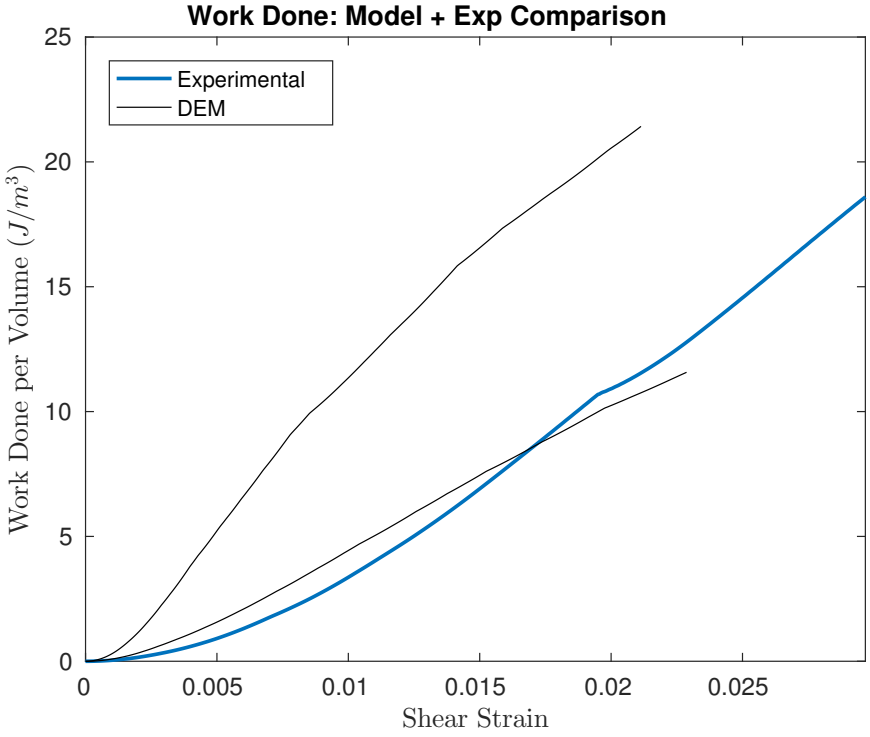
Tests are named by the date on which the test took place in a YYYYMMDD format.

Test 20160802: Homogeneous

Experimental vs Numerical Comparison

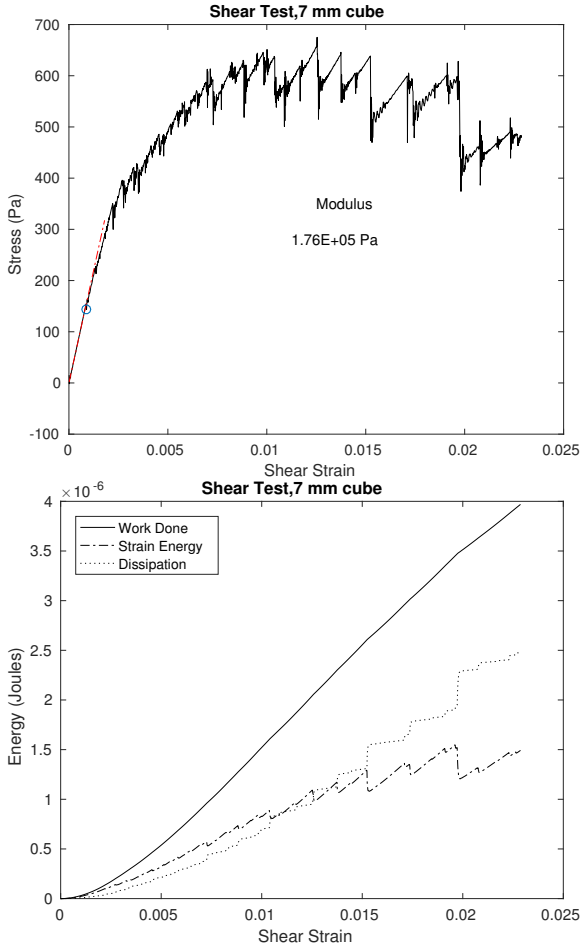


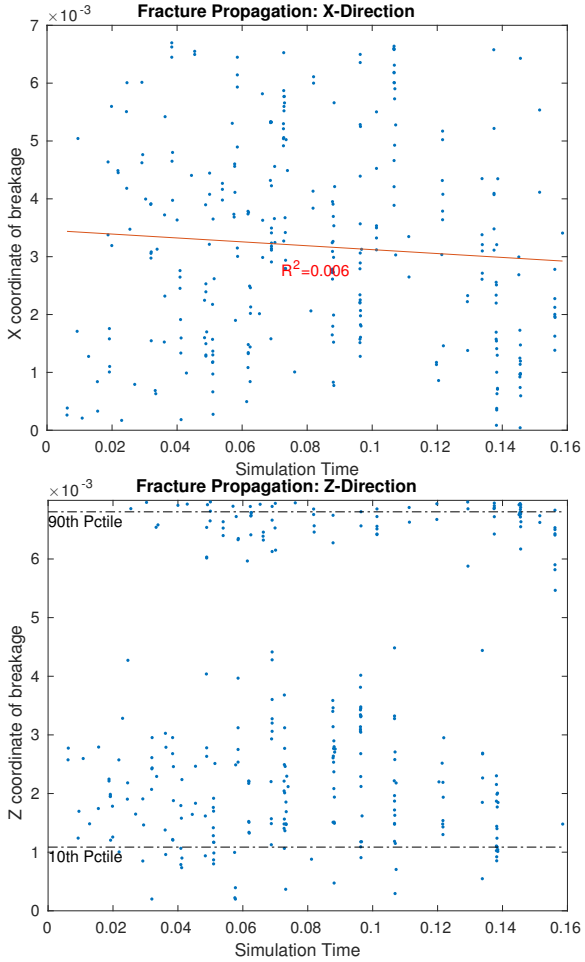

Shear Stress v Strain
Experimental
DEM
Shear Stress (Pa)
Shear Strain
2000
1800
1600
1400
1200
1000
800
600
400
200
0
0
0.005
0.01
0.015
0.02
0.025


Work Done: Model + Exp Comparison
Experimental
DEM
Work Done per Volume (J/m^3)
Shear Strain
25
20
15
10
5
0
0
0.005
0.01
0.015
0.02
0.025

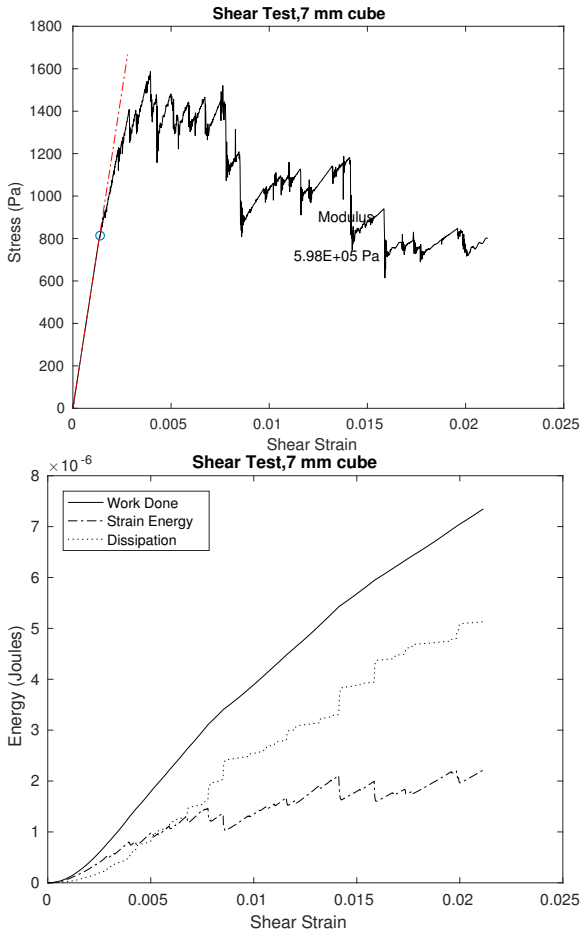
Detailed Numerical Results

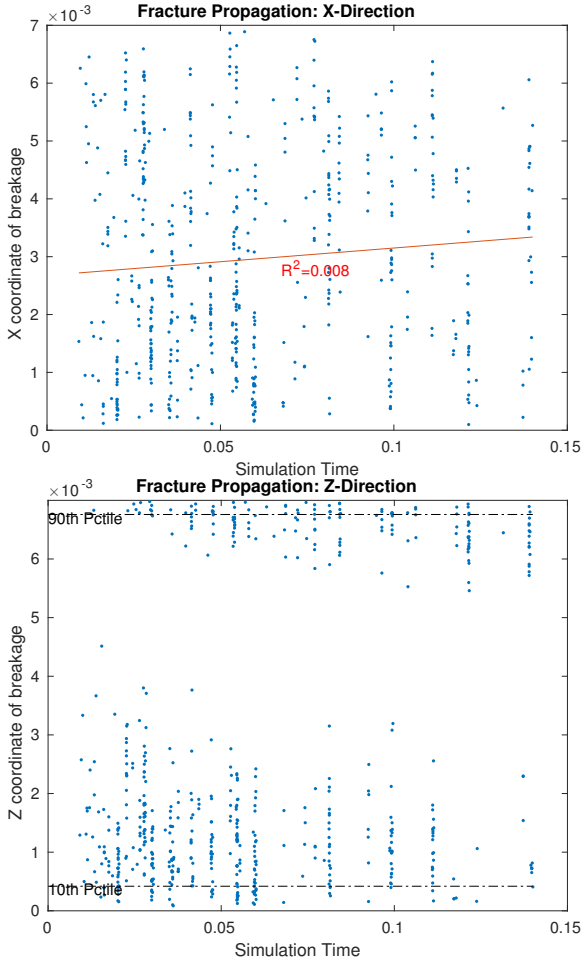
Numerical Subsample 7mm1:




Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.006
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

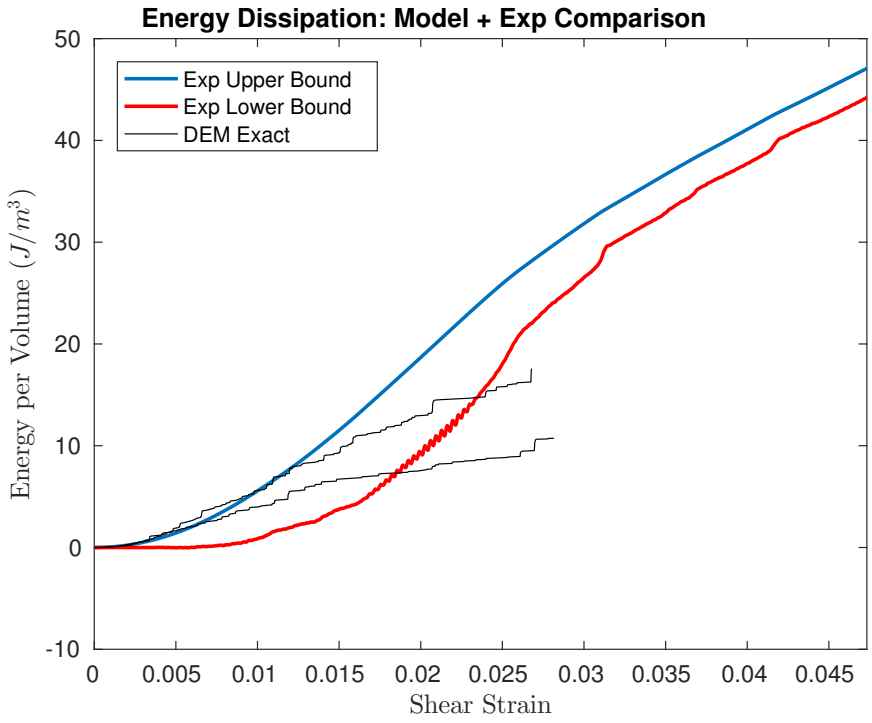
Numerical Subsample 7mm2:

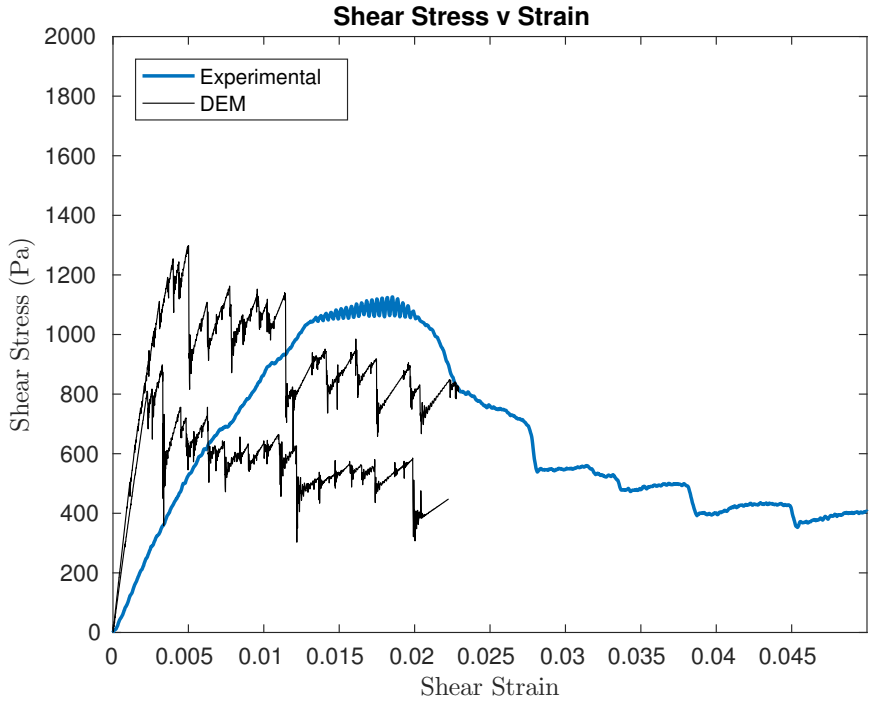


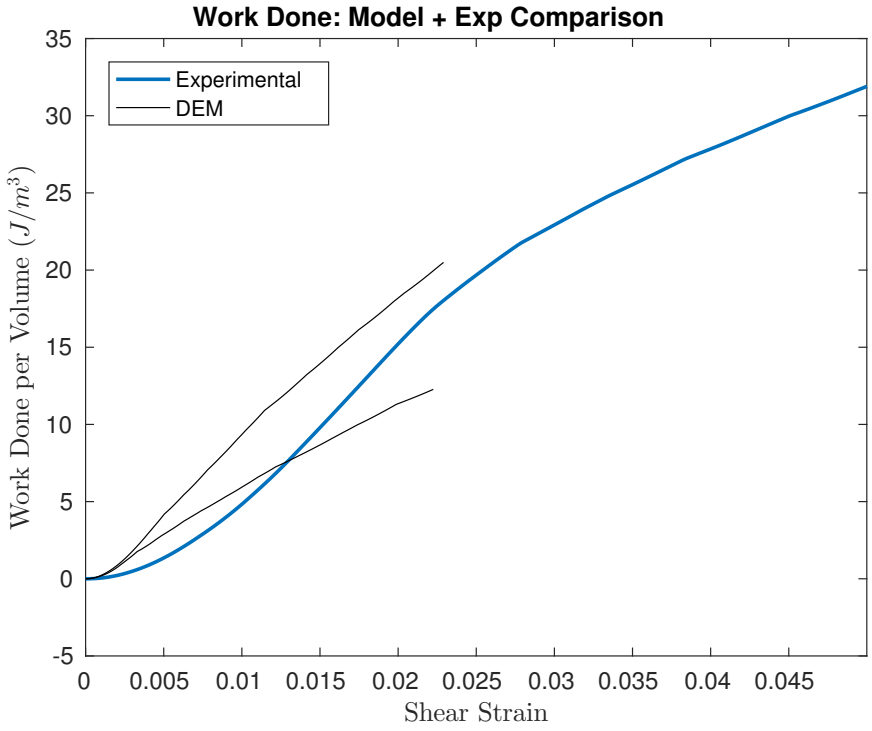

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.008
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

Test 20160803: Homogeneous

Experimental vs Numerical Comparison

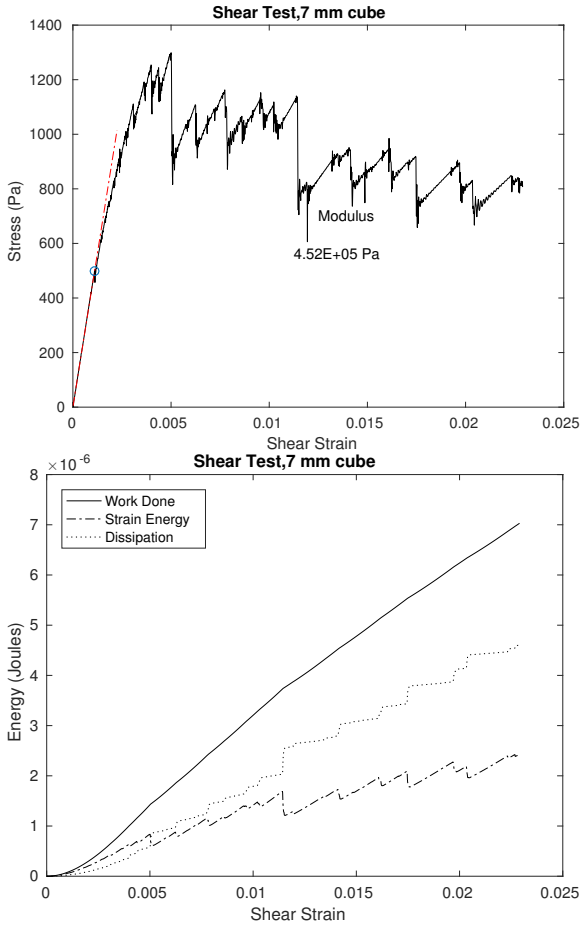


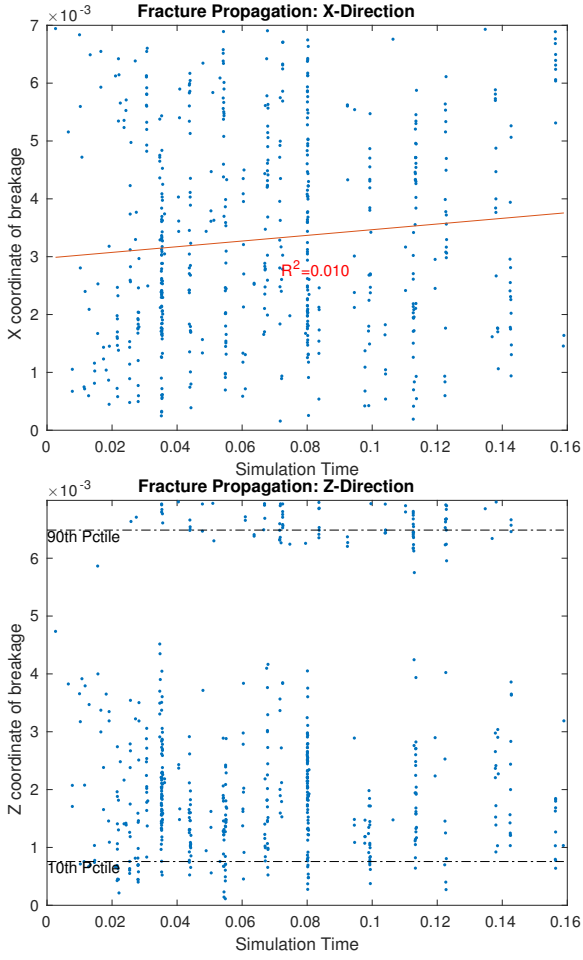

Shear Stress v Strain
Experimental
DEM
Shear Stress (Pa)
Shear Strain
2000
1800
1600
1400
1200
1000
800
600
400
200
0
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045


Work Done: Model + Exp Comparison
Experimental
DEM
Work Done per Volume (J/m^3)
Shear Strain
35
30
25
20
15
10
5
0
-5
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045

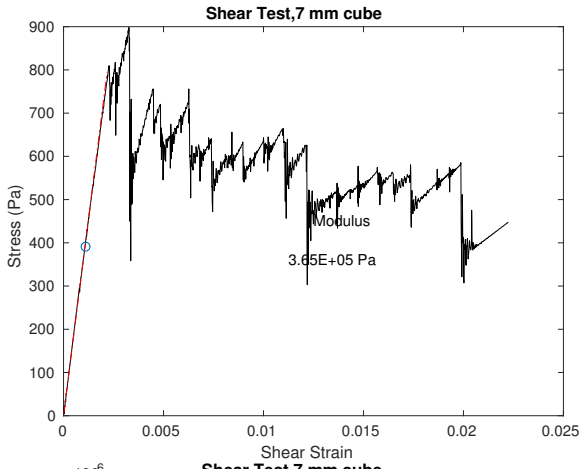
Detailed Numerical Results

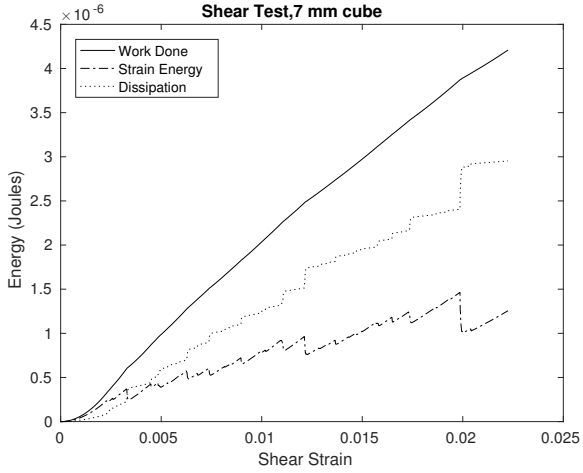
Numerical Subsample 7mm1:

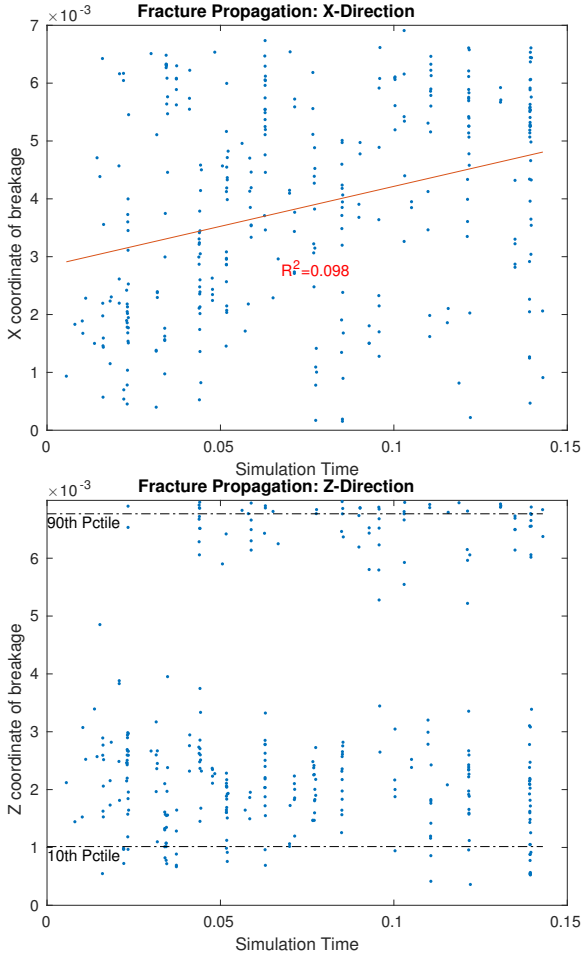



Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.010
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

Numerical Subsample 7mm2:

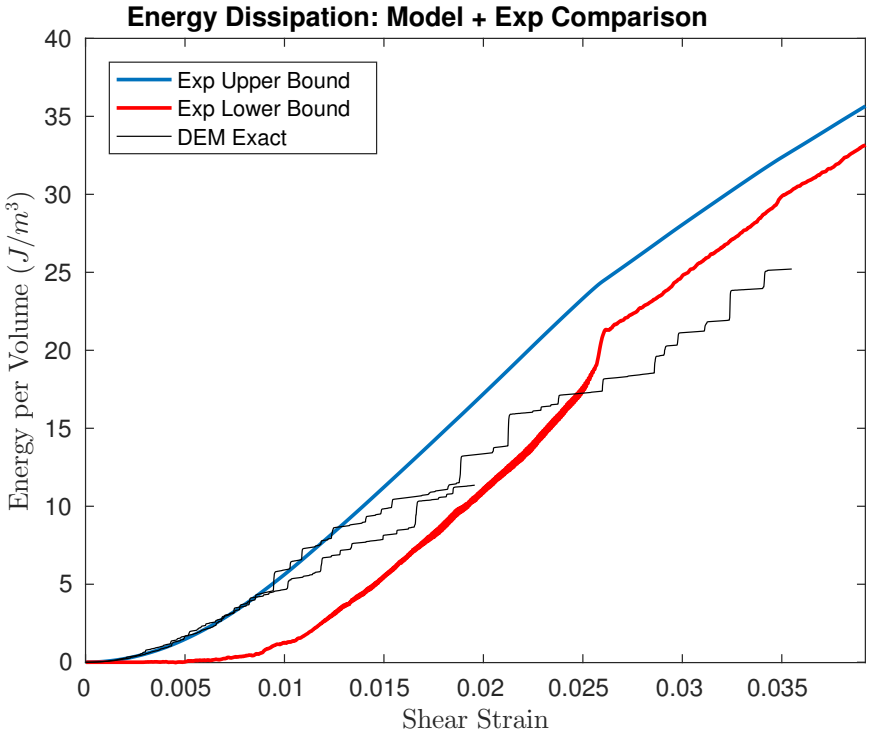


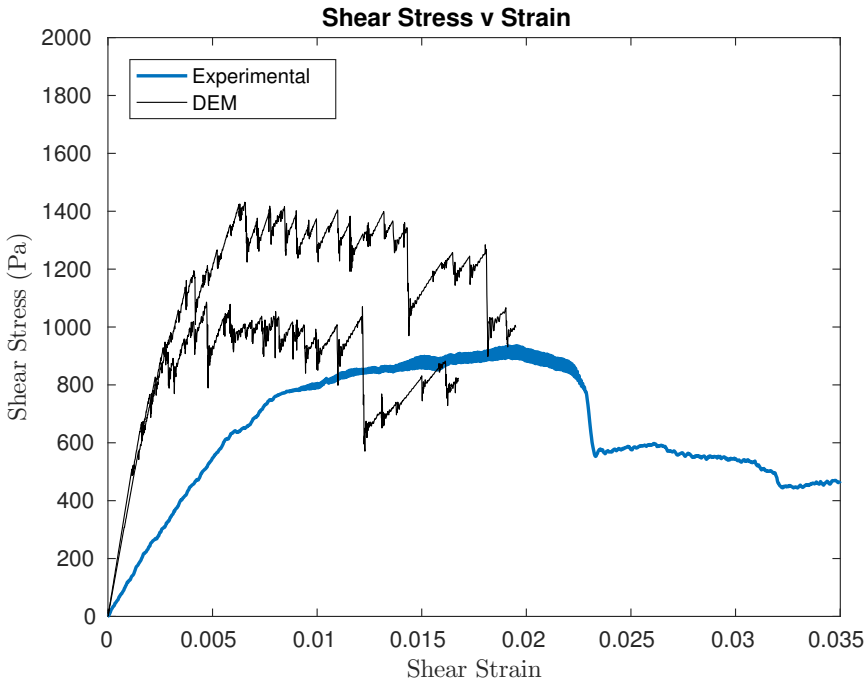


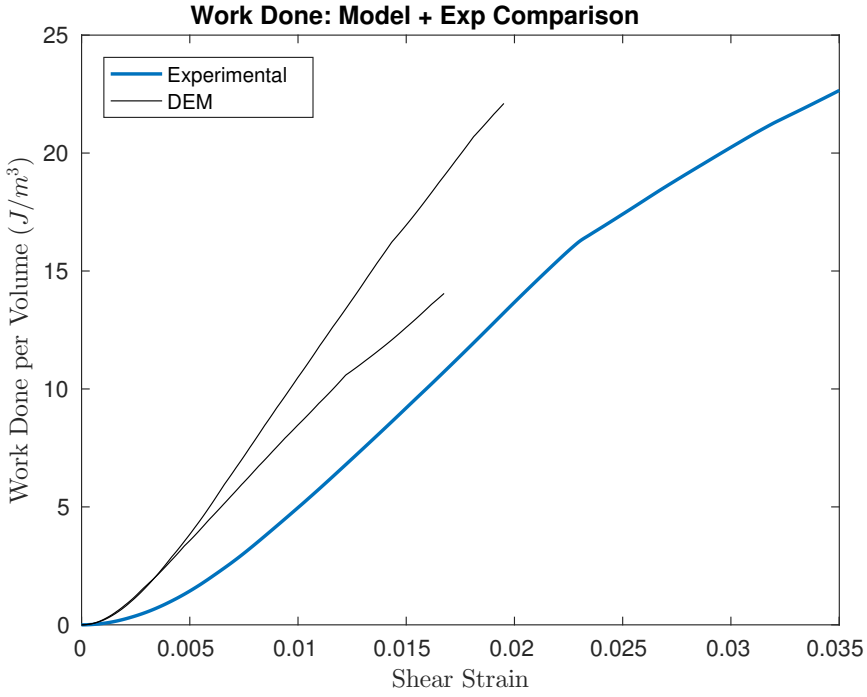

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.098
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

Test 20160804: Homogeneous

Experimental vs Numerical Comparison

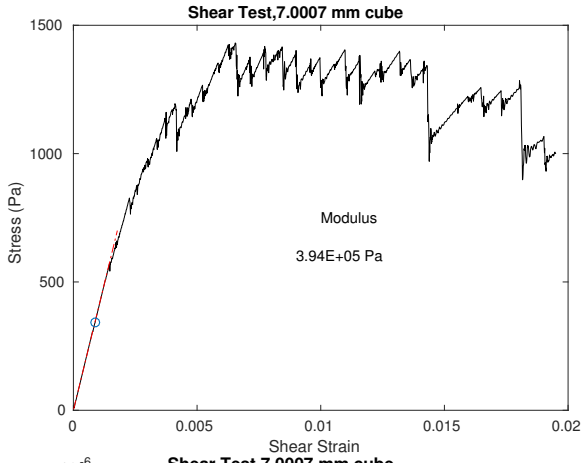


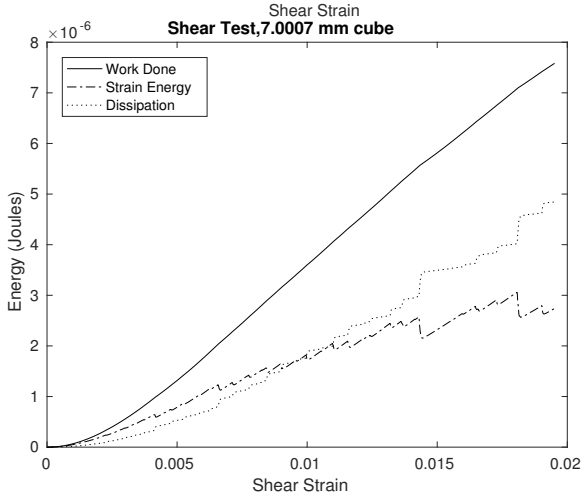

Shear Stress v Strain
Experimental
DEM
Shear Stress (Pa)
Shear Strain
0
200
400
600
800
1000
1200
1400
1600
1800
2000
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035

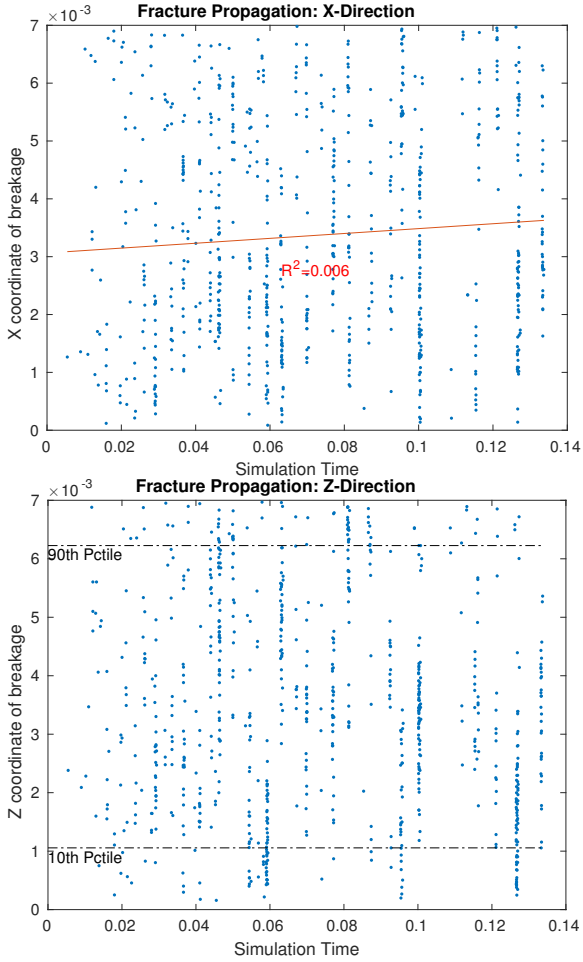

Work Done: Model + Exp Comparison
Experimental
DEM
Work Done per Volume (J/m^3)
Shear Strain
0
5
10
15
20
25
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035

Detailed Numerical Results

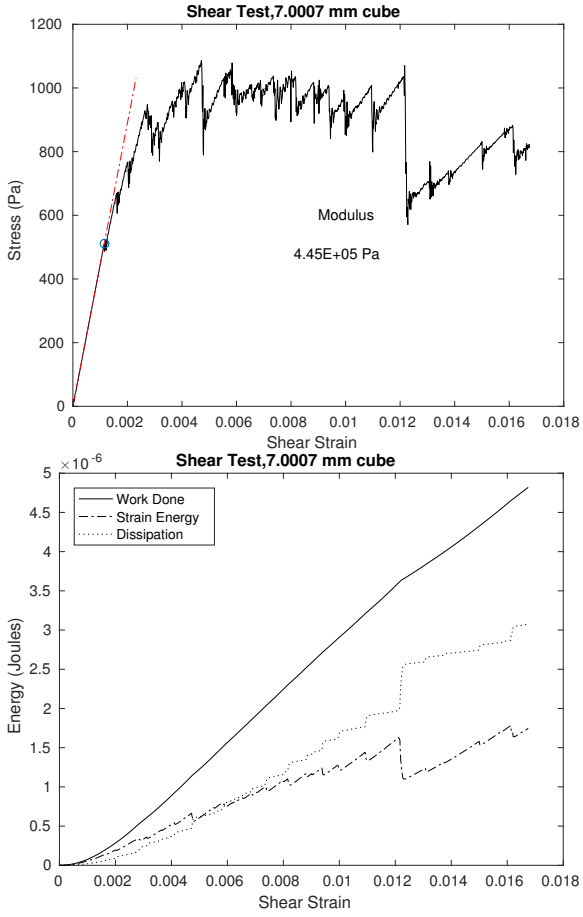
Numerical Subsample 7mm1:

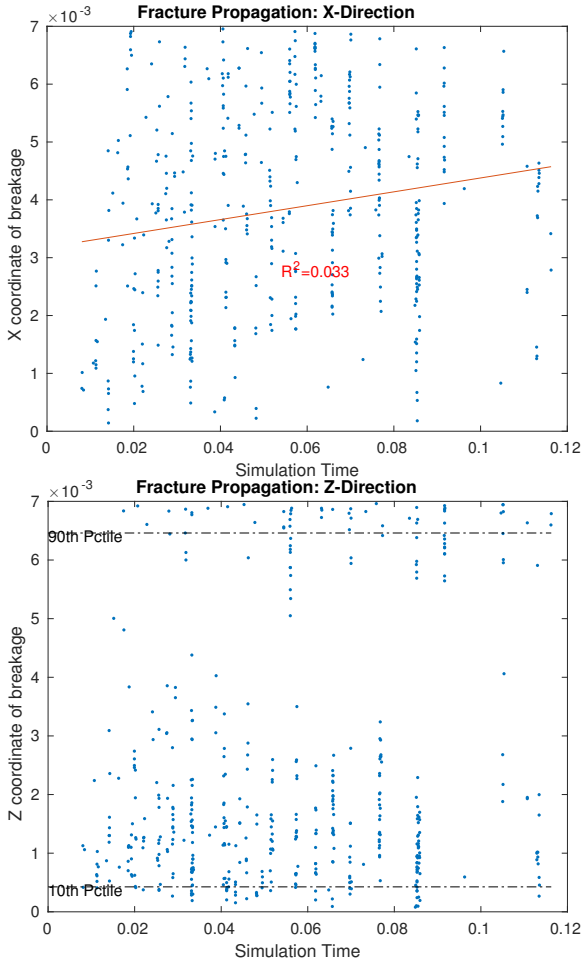





Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2 =0.006
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

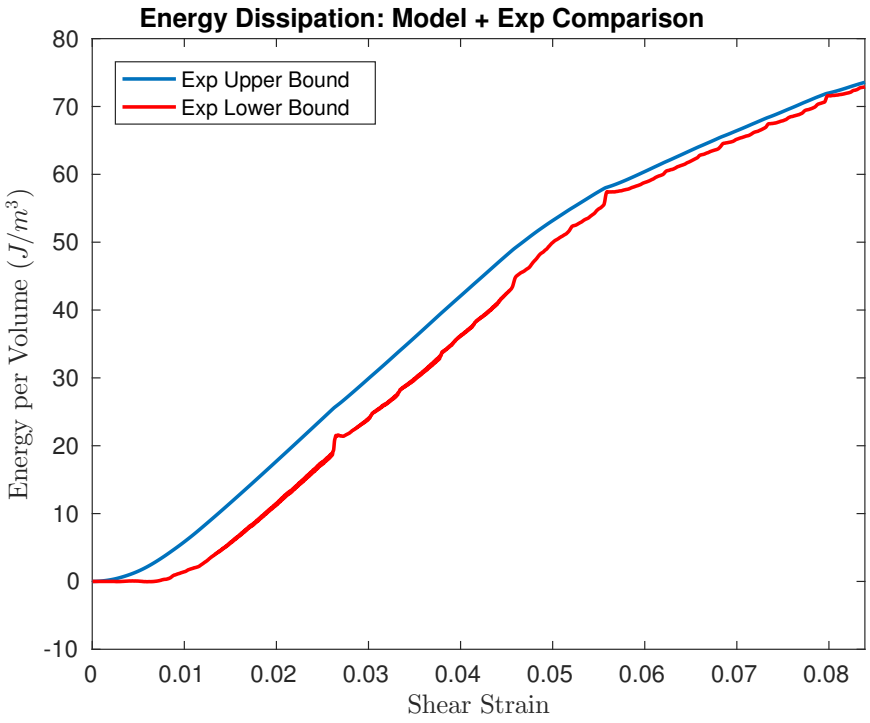
Numerical Subsample 7mm2:

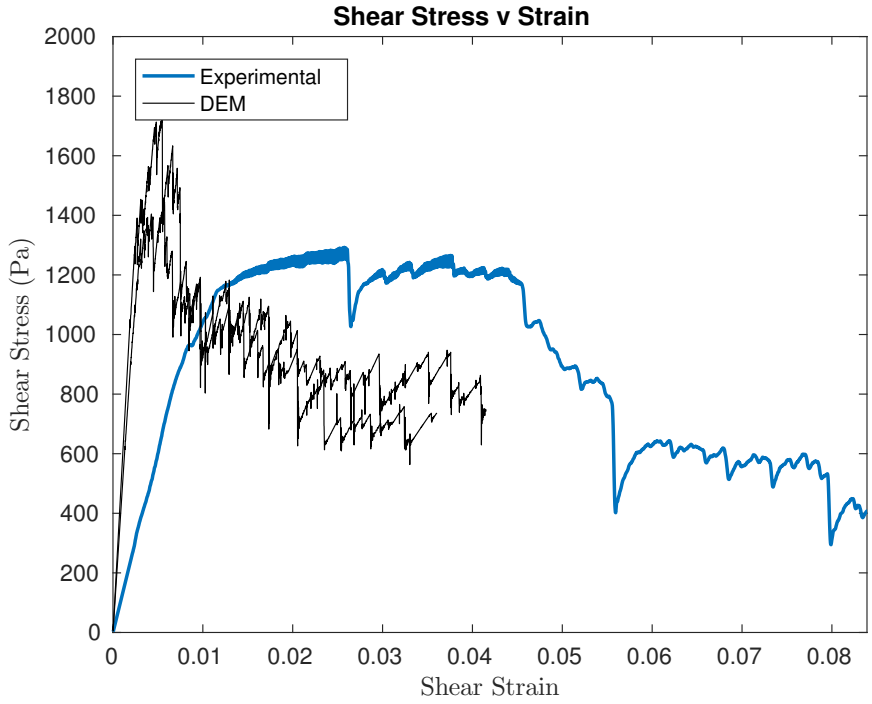


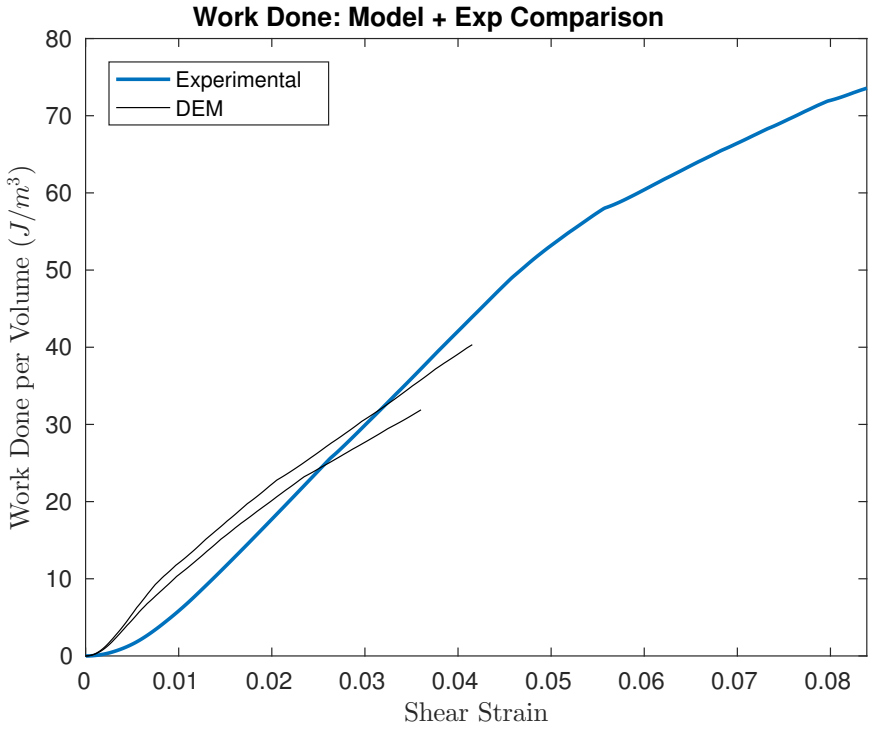

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.033
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
Simulation Time
90th Pctile
10th Pctile

Test_20160809: Homogeneous

Experimental vs Numerical Comparison

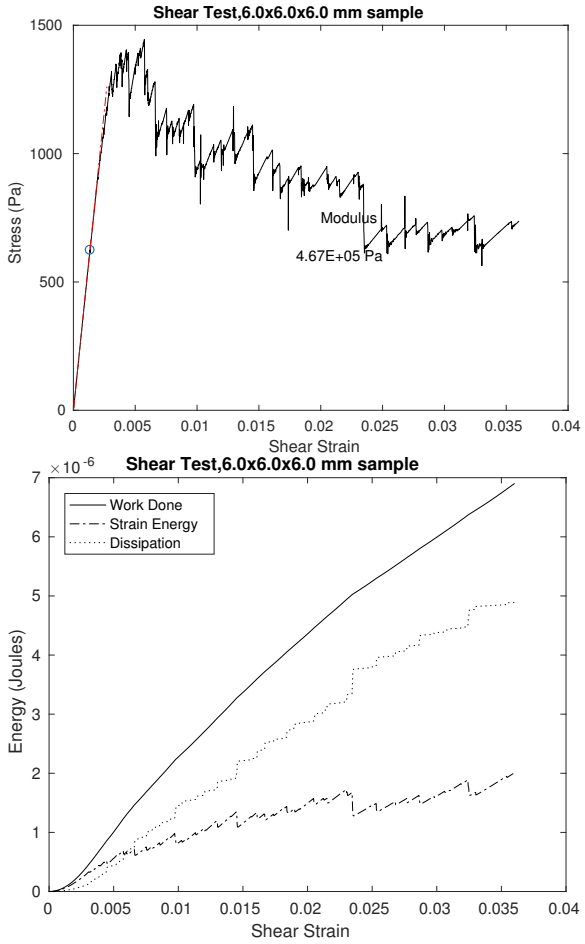


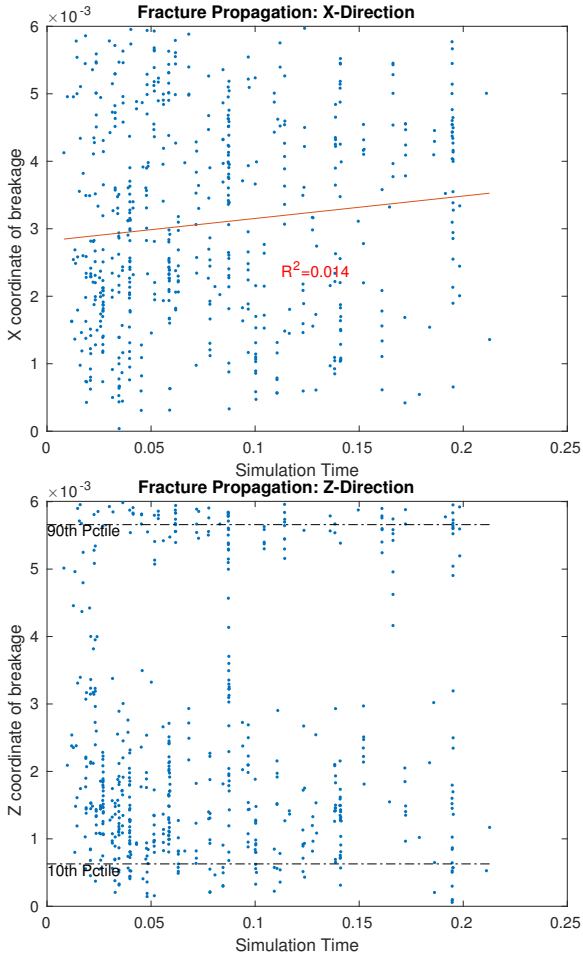

Shear Stress v Strain
Experimental
DEM
Shear Stress (Pa)
Shear Strain
0
200
400
600
800
1000
1200
1400
1600
1800
2000
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08


Work Done: Model + Exp Comparison
Experimental
DEM
Work Done per Volume (J/m^3)
Shear Strain
0
10
20
30
40
50
60
70
80
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08

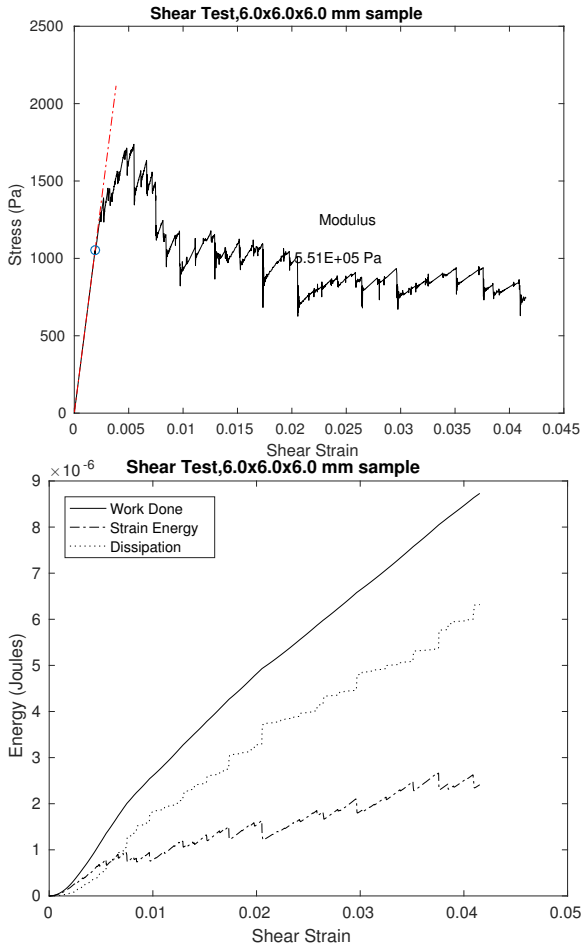
Detailed Numerical Results

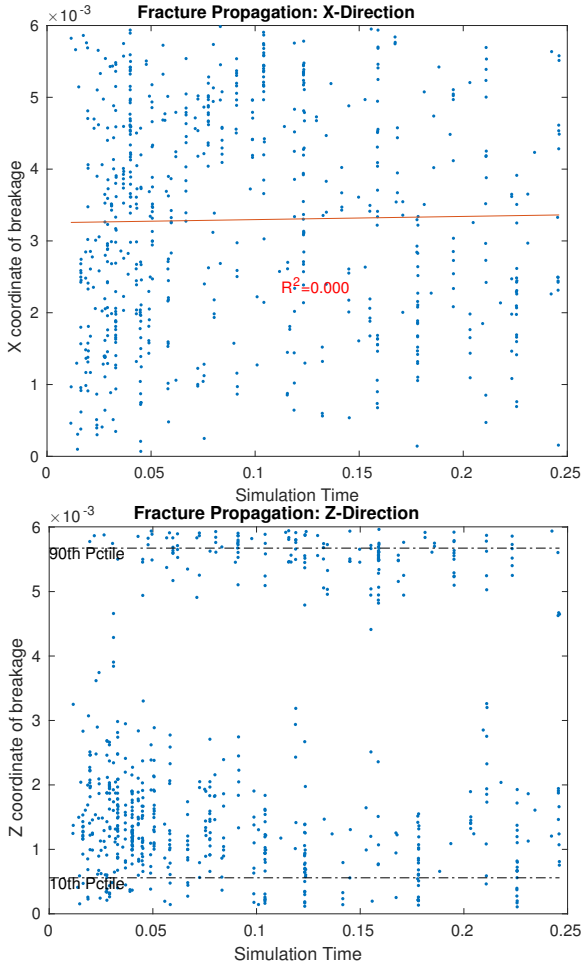
Numerical Subsample 6mm1:




Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.014
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
Simulation Time
90th Pctile
10th Pctile

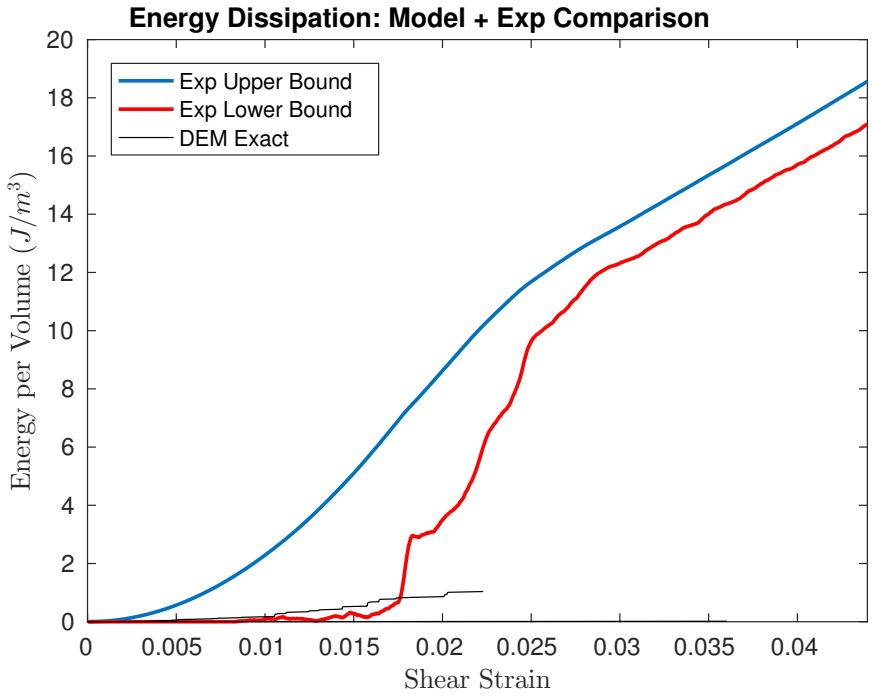
Numerical Subsample 6mm2:

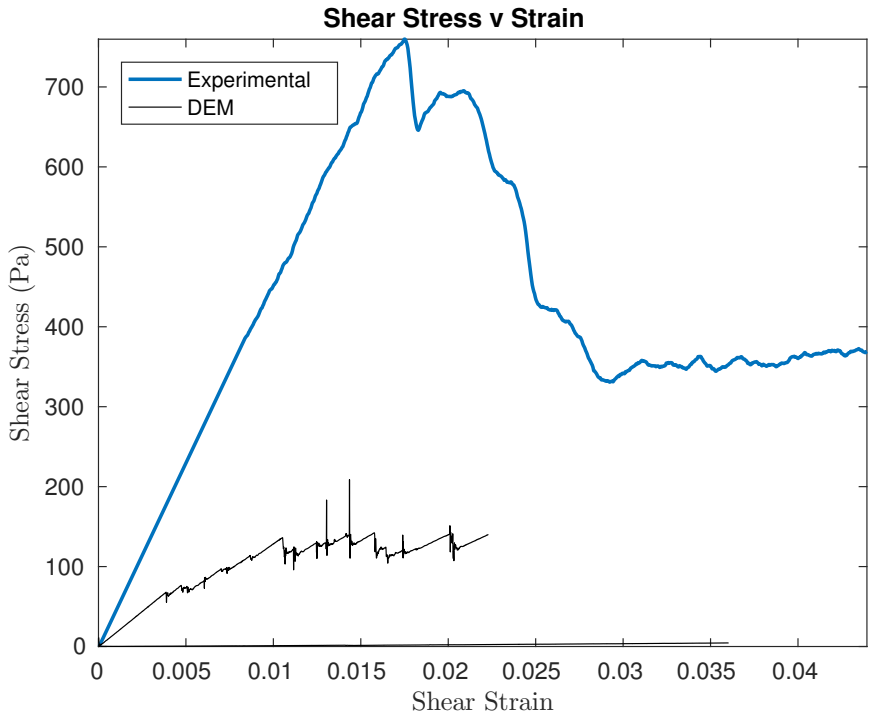


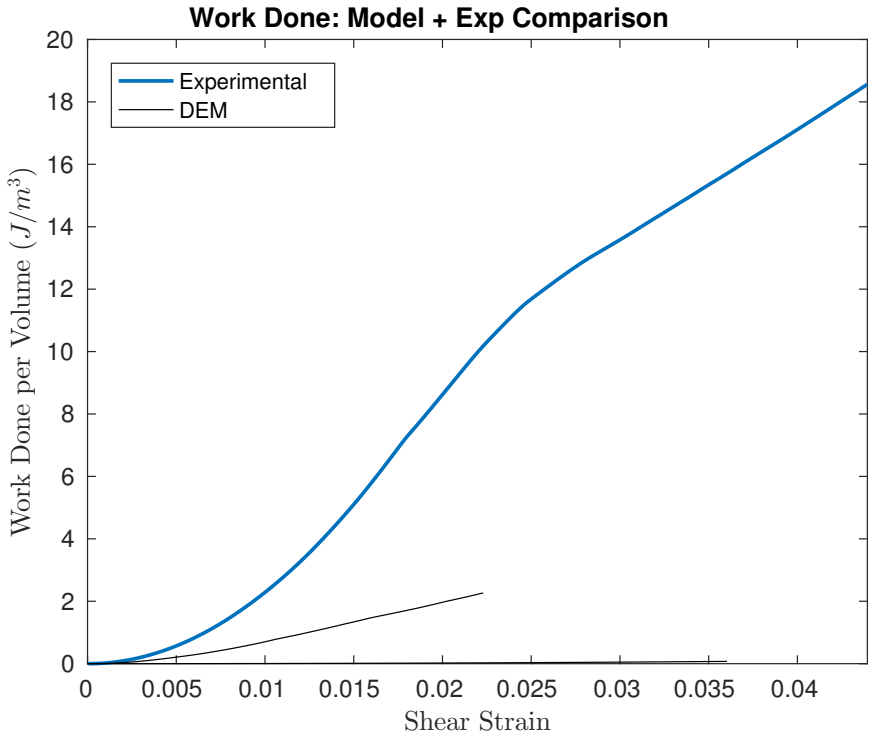

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.000
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

Test 20160916: Homogeneous

Experimental vs Numerical Comparison

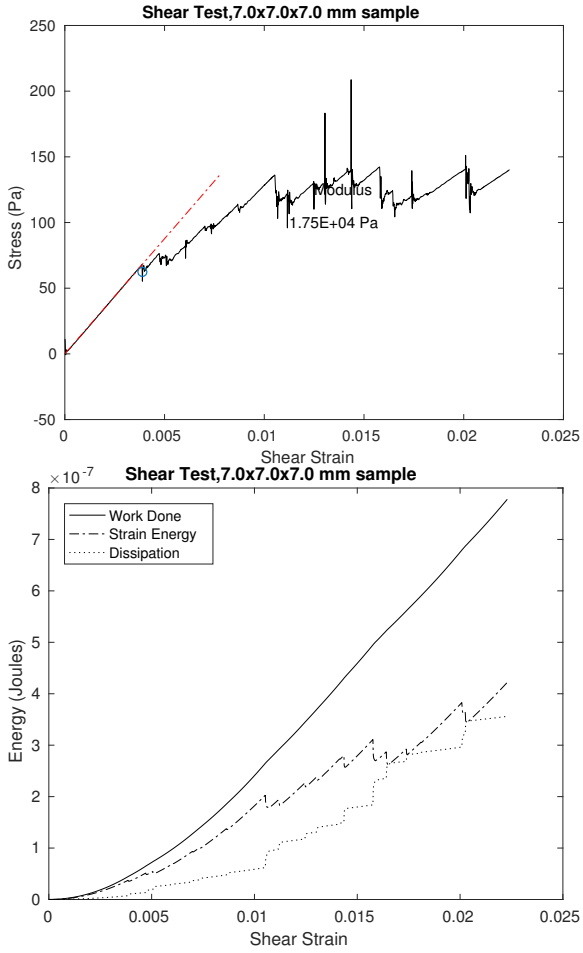


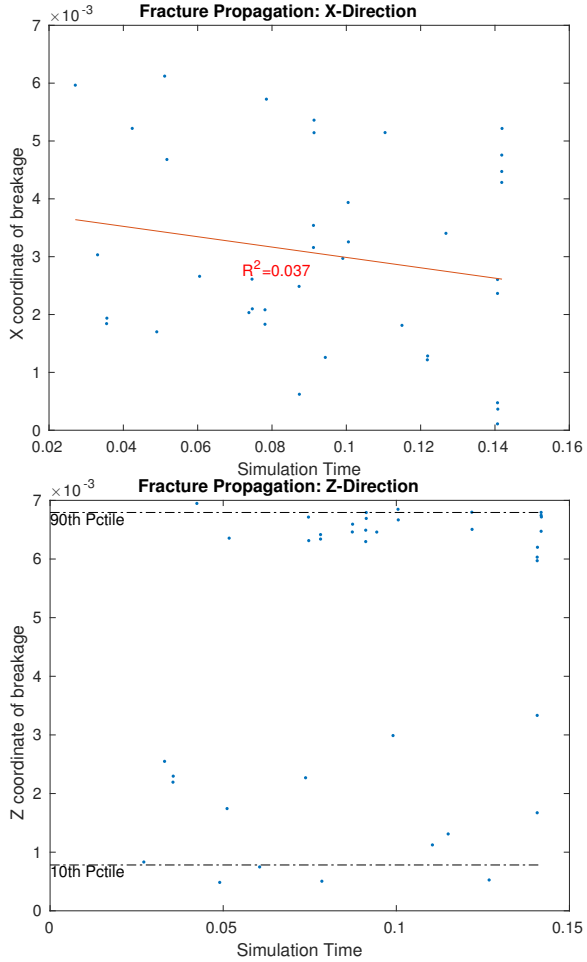

Shear Stress v Strain
Experimental
DEM
Shear Stress (Pa)
Shear Strain
0
100
200
300
400
500
600
700
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04


Work Done: Model + Exp Comparison
Experimental
DEM
Work Done per Volume (J/m^3)
Shear Strain
0
2
4
6
8
10
12
14
16
18
20
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04

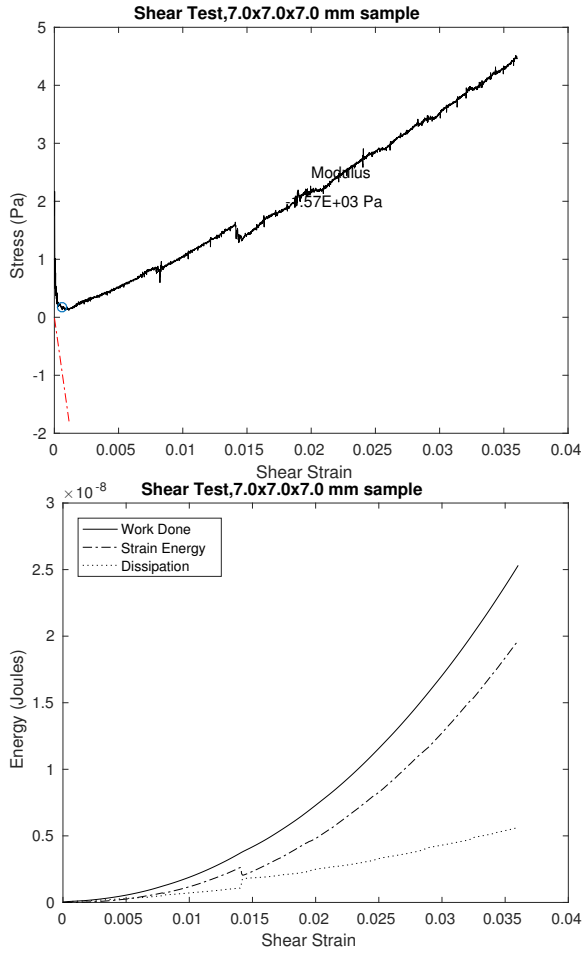
Detailed Numerical Results

Numerical Subsample 7mm1:




Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.037
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
Simulation Time
90th Pctile
10th Pctile

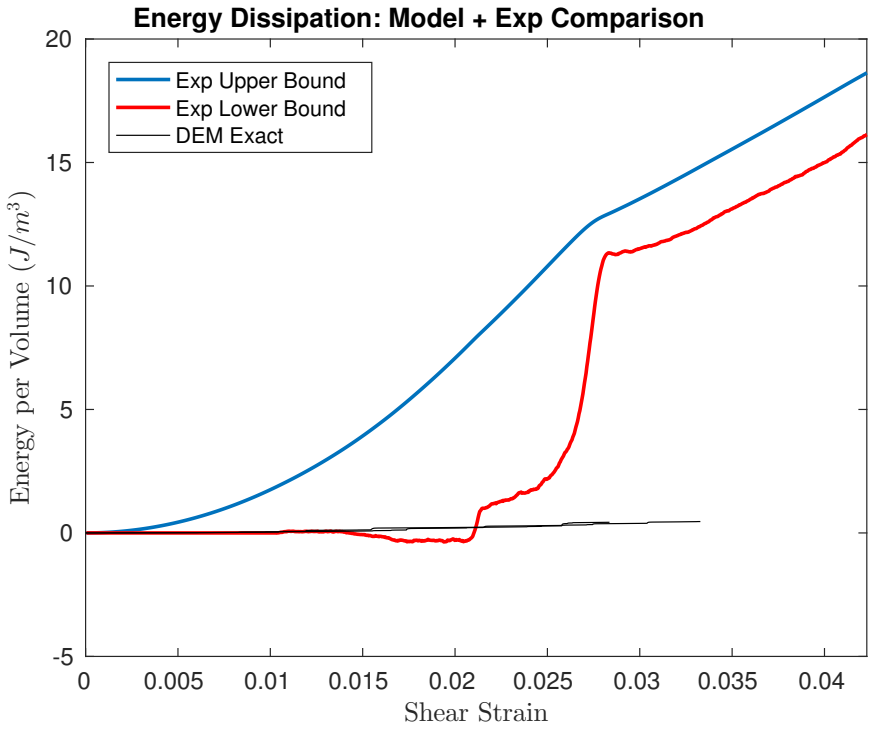
Numerical Subsample 7mm2:

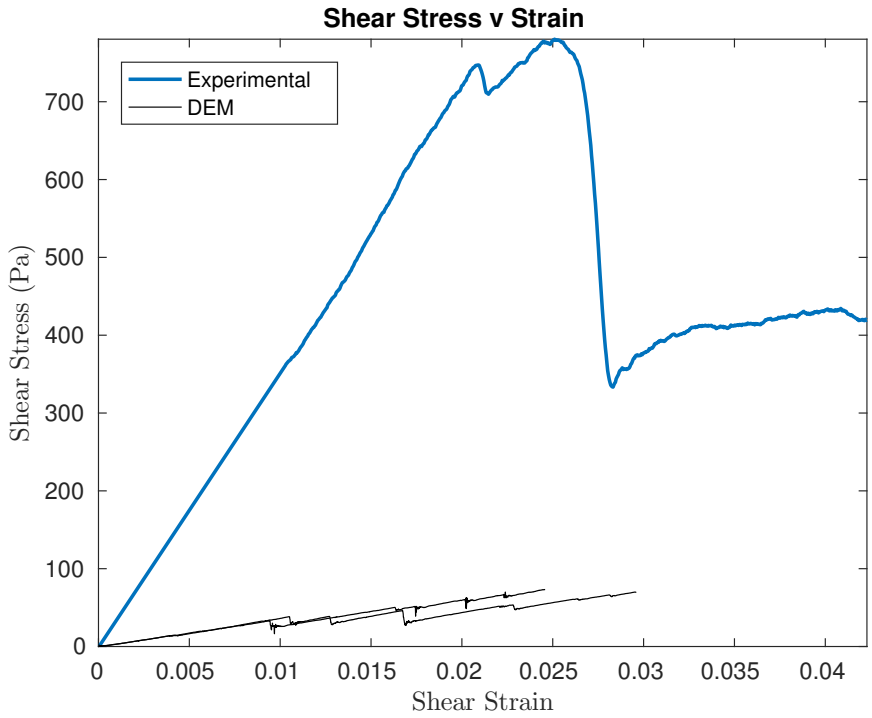


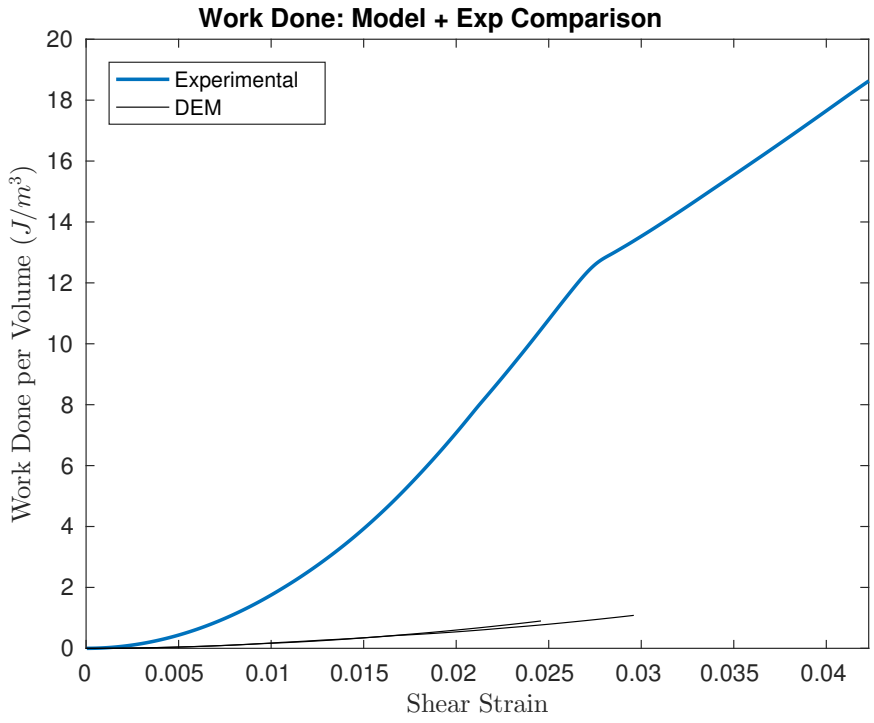
No bonds broke in this subsample, so bond breakage plots are not included.

Test 20160919: Homogeneous

Experimental vs Numerical Comparison

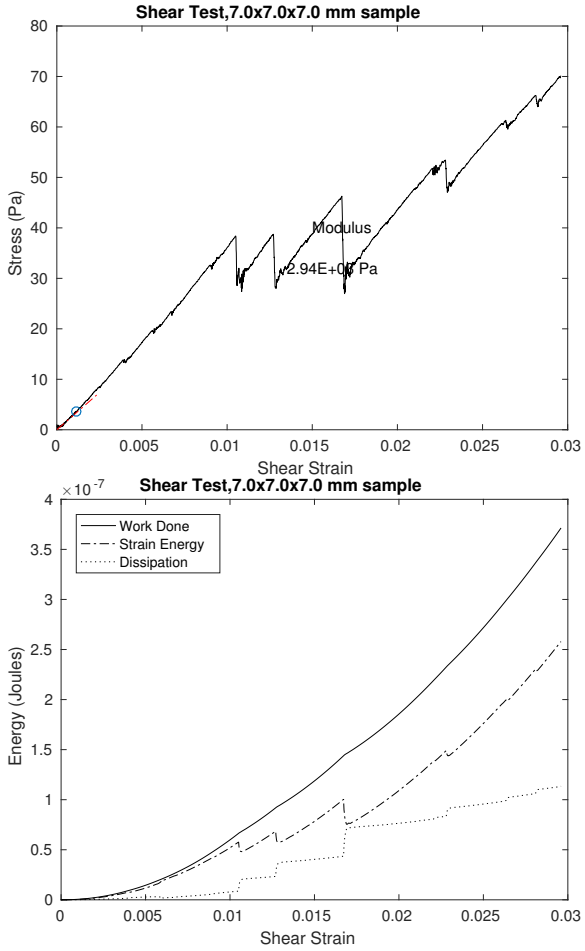


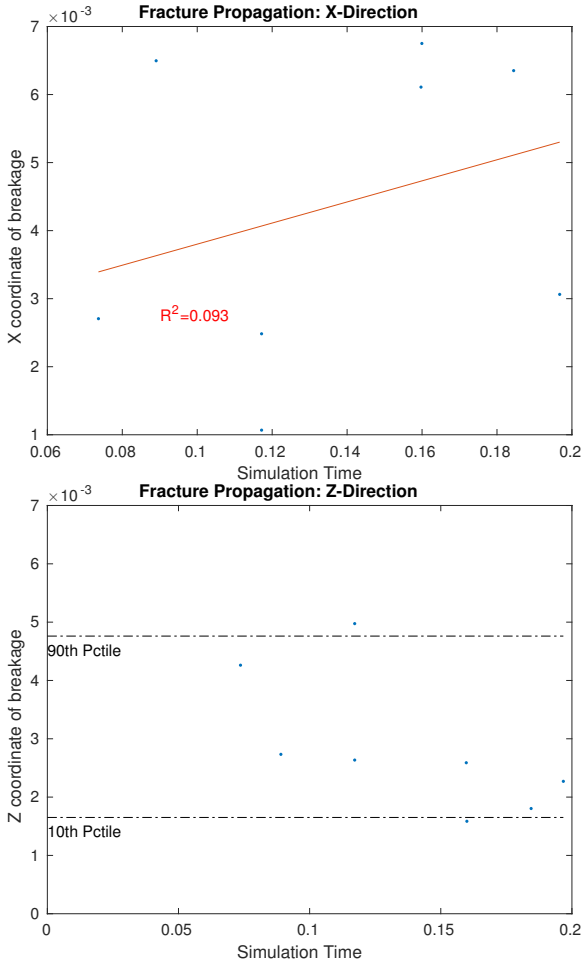

Shear Stress v Strain
Experimental
DEM
Shear Stress (Pa)
Shear Strain
0
100
200
300
400
500
600
700
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04


Work Done: Model + Exp Comparison
Experimental
DEM
Work Done per Volume (J/m^3)
Shear Strain
0
2
4
6
8
10
12
14
16
18
20
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04

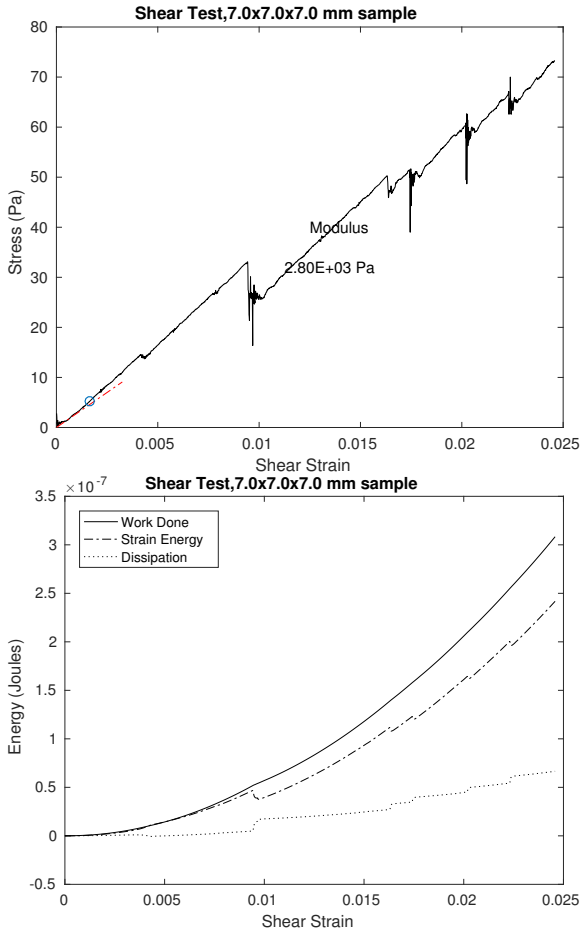
Detailed Numerical Results

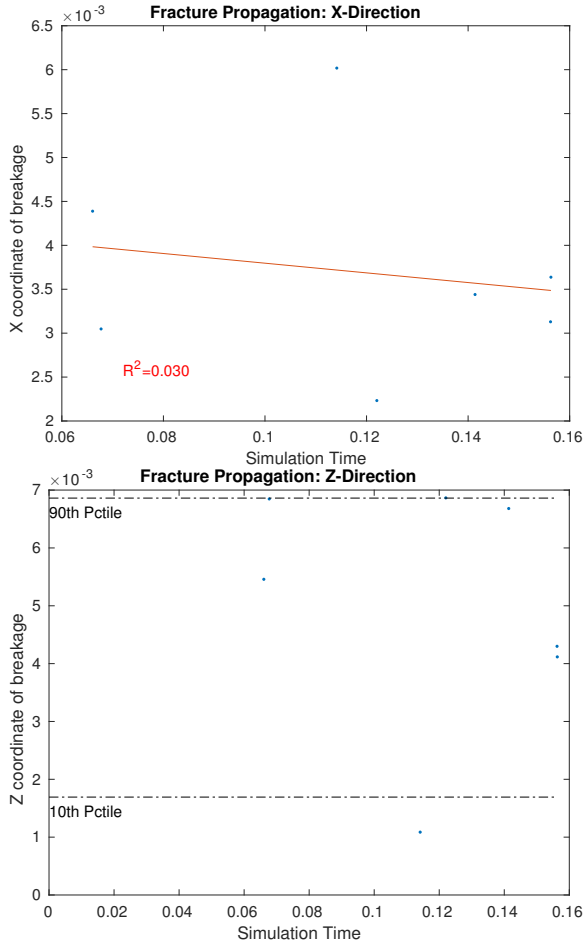
Numerical Subsample 7mm1:




Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.093
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

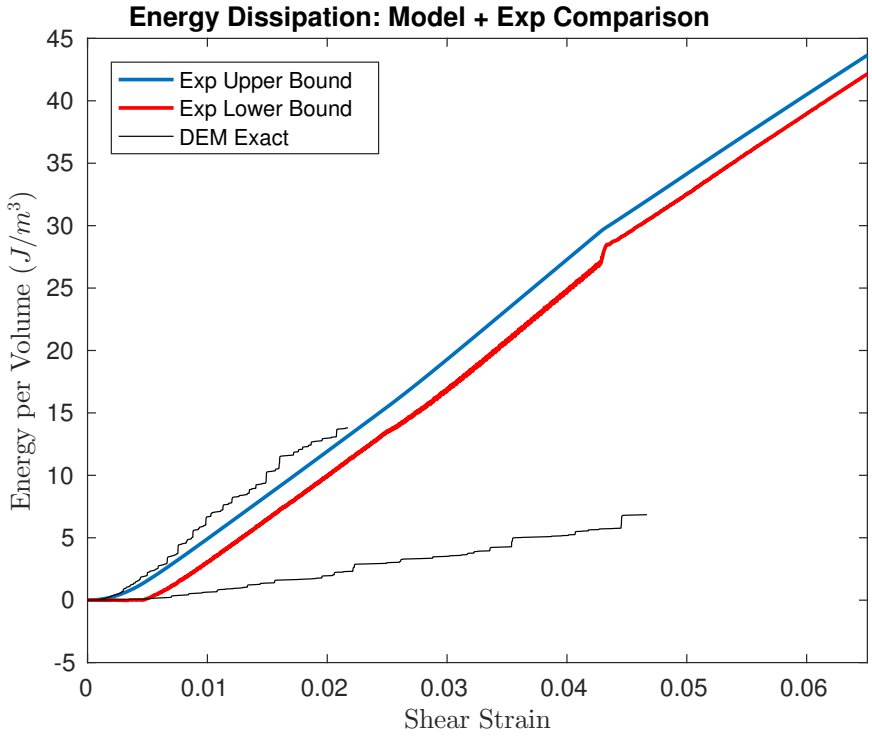
Numerical Subsample 7mm2:

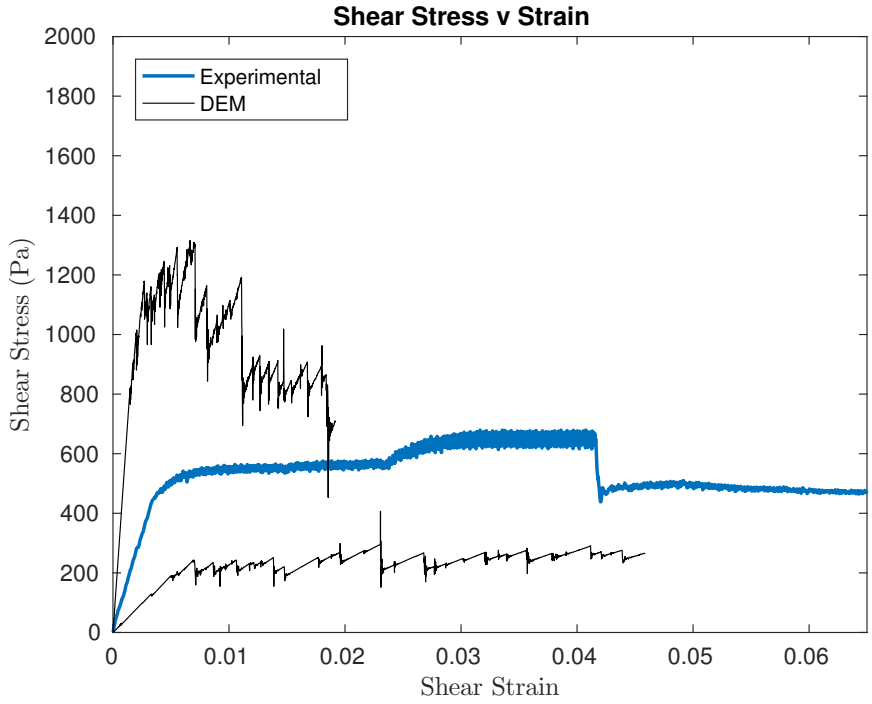


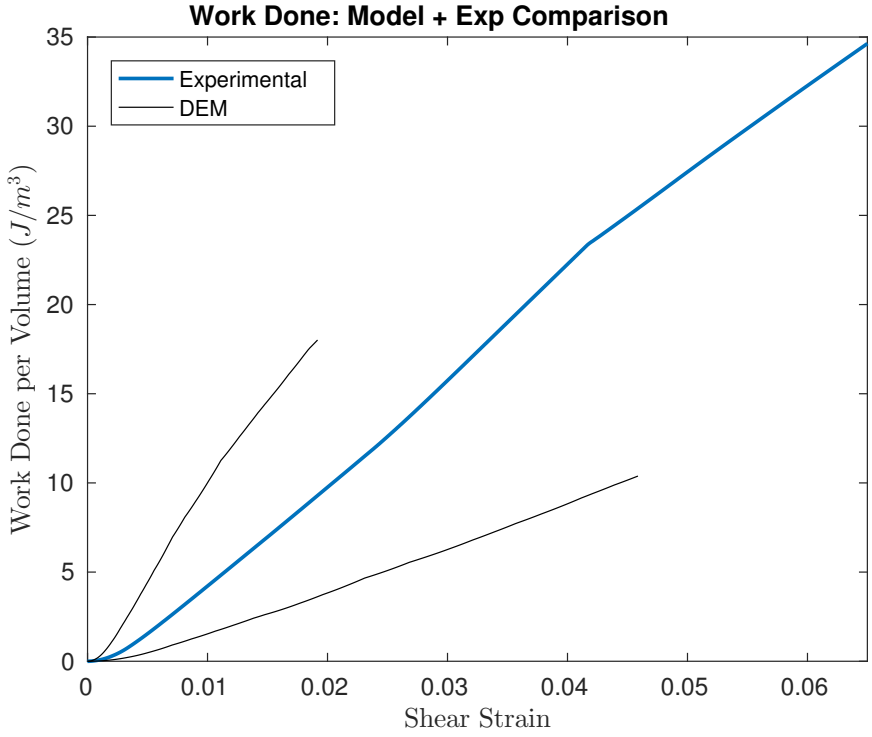

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.030
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

Test 20161120: Homogeneous

Experimental vs Numerical Comparison

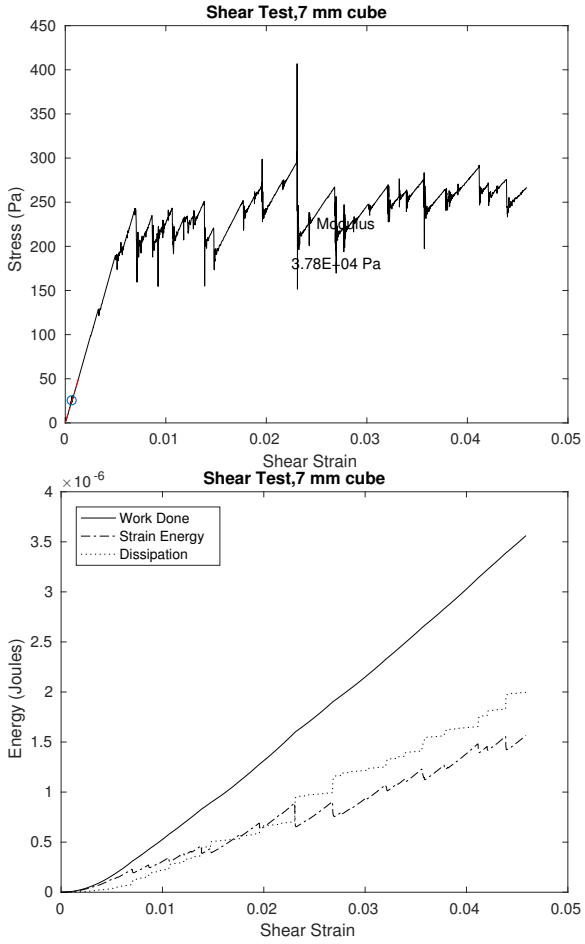


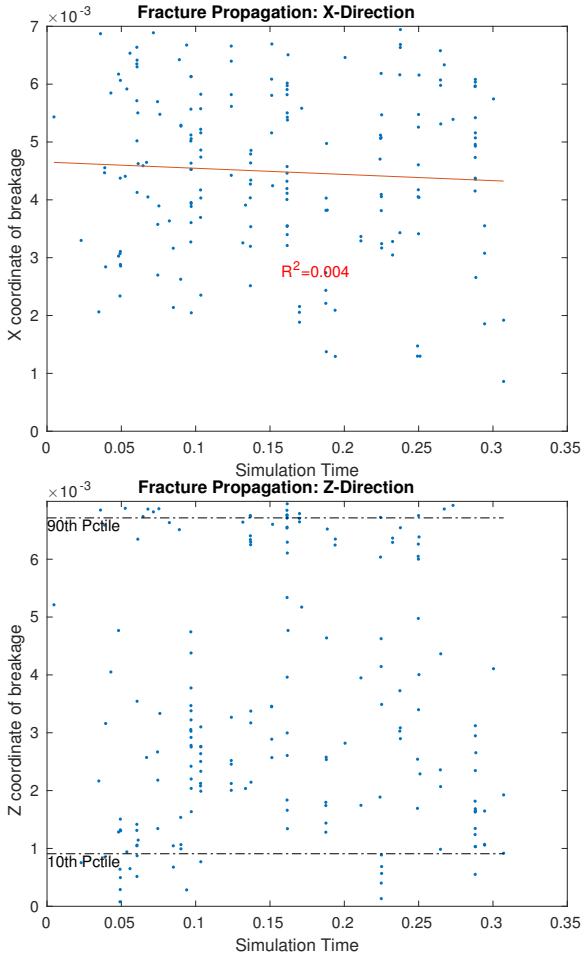

Shear Stress v Strain
Experimental
DEM
Shear Stress (Pa)
Shear Strain
0
200
400
600
800
1000
1200
1400
1600
1800
2000
0.01
0.02
0.03
0.04
0.05
0.06


Work Done: Model + Exp Comparison
Experimental
DEM
Work Done per Volume (J/m^3)
Shear Strain
0
5
10
15
20
25
30
35
0.01
0.02
0.03
0.04
0.05
0.06

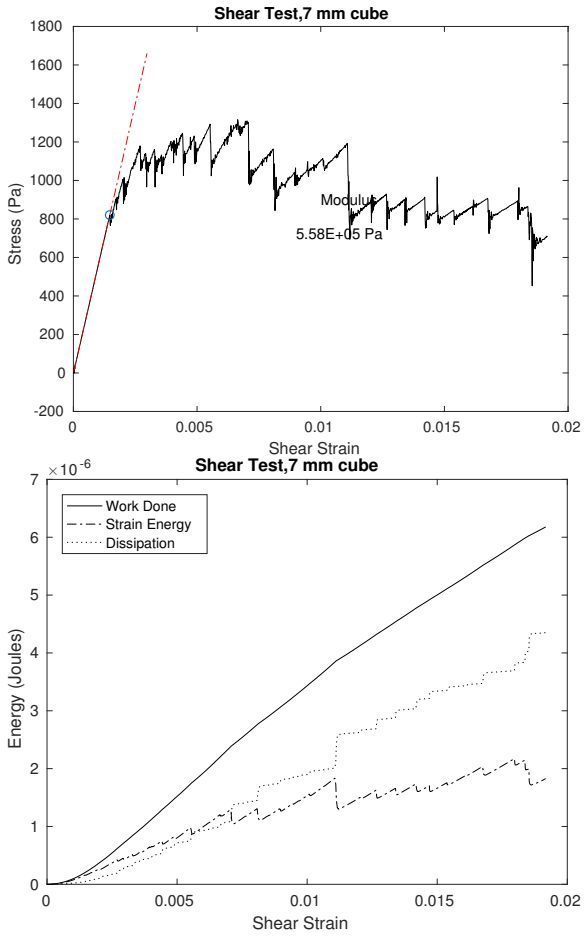
Detailed Numerical Results

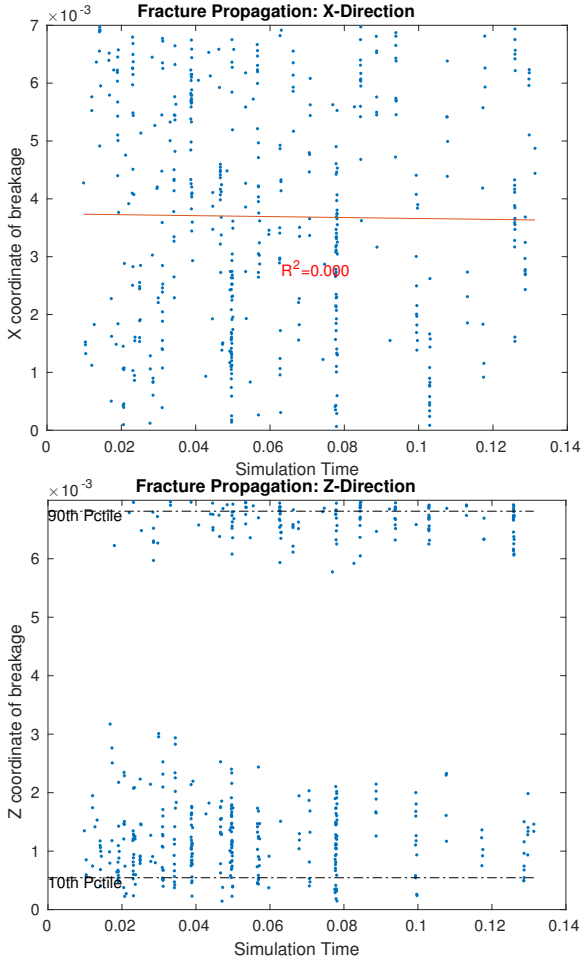
Numerical Subsample 7mm1:




Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.004
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

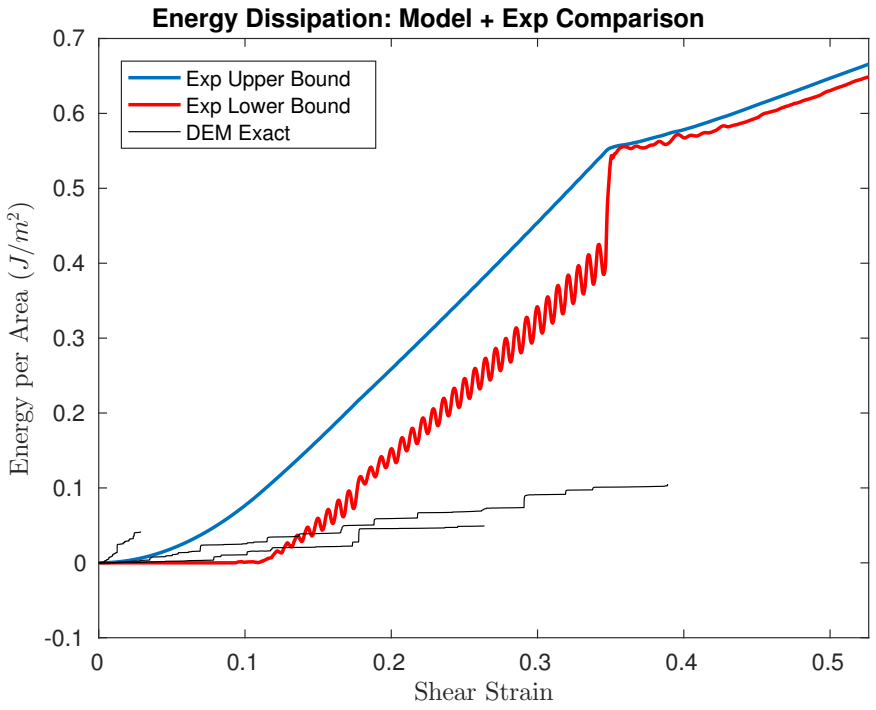
Numerical Subsample 7mm2:

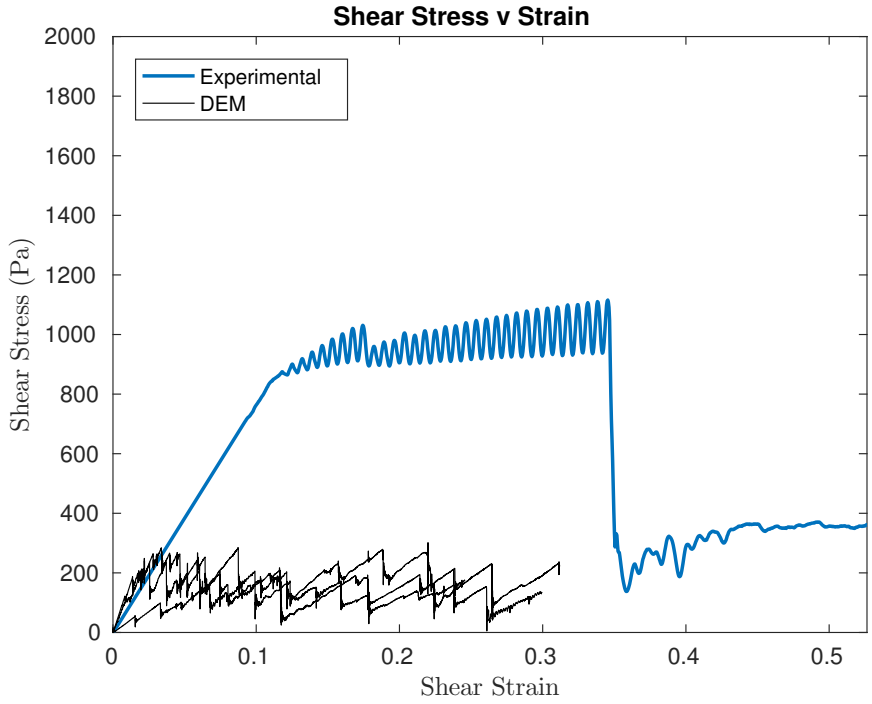


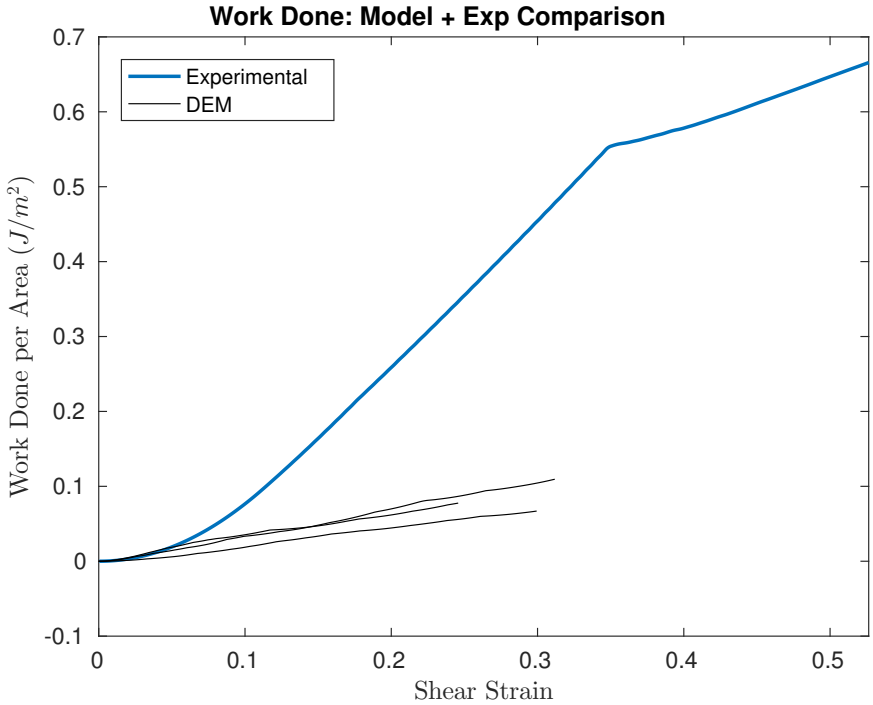

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.000
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
Simulation Time
90th Pctile
10th Pctile

Test 20170125: With Buried Surface Hoar

Experimental vs Numerical Comparison

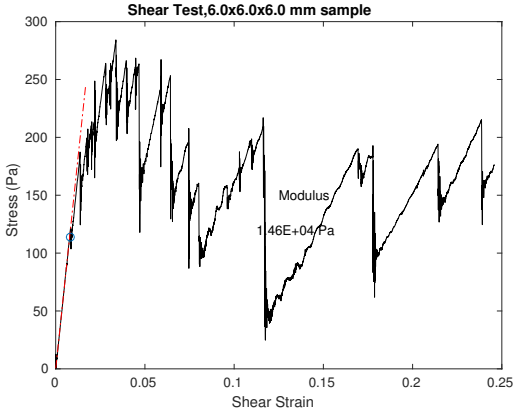


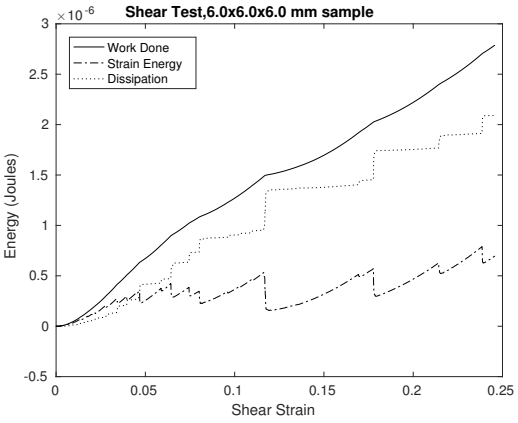

Shear Stress v Strain
Experimental
DEM
Shear Stress (Pa)
Shear Strain
2000
1800
1600
1400
1200
1000
800
600
400
200
0
0
0.1
0.2
0.3
0.4
0.5

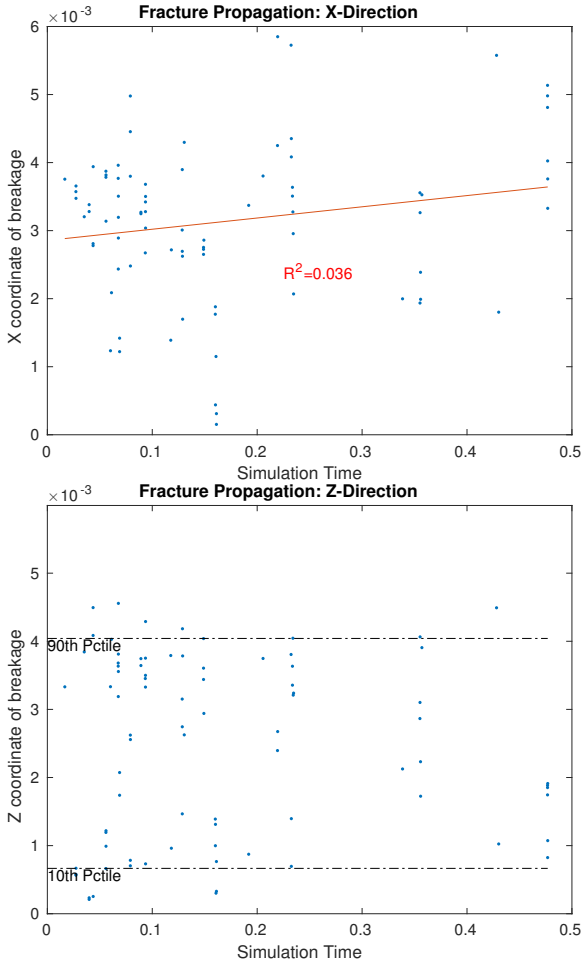

Work Done: Model + Exp Comparison
Experimental
DEM
Work Done per Area (J/m^2)
Shear Strain
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
0
0.1
0.2
0.3
0.4
0.5

Detailed Numerical Results

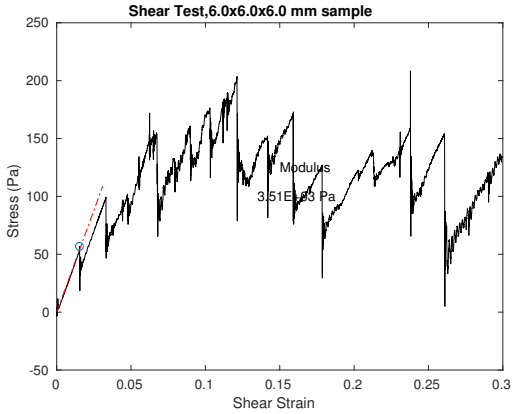
Numerical Subsample 6mm1:

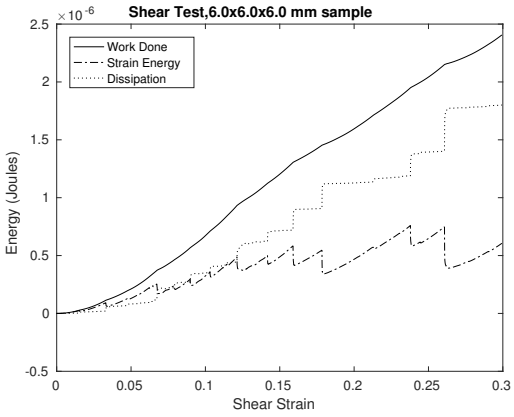


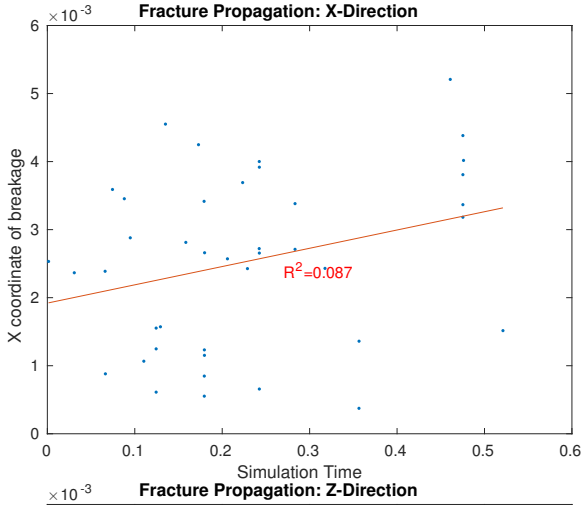


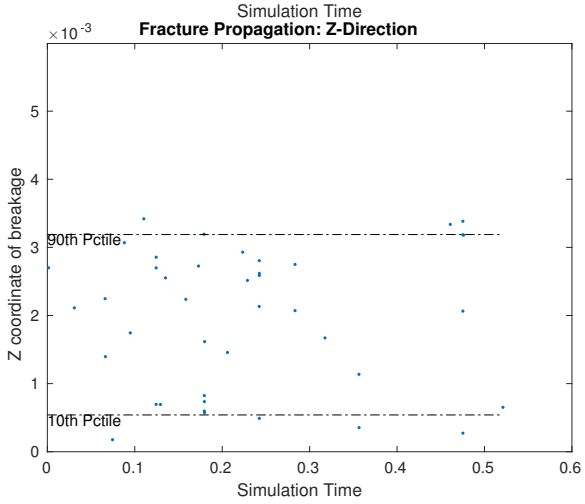

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.036
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

Numerical Subsample 6mm2:



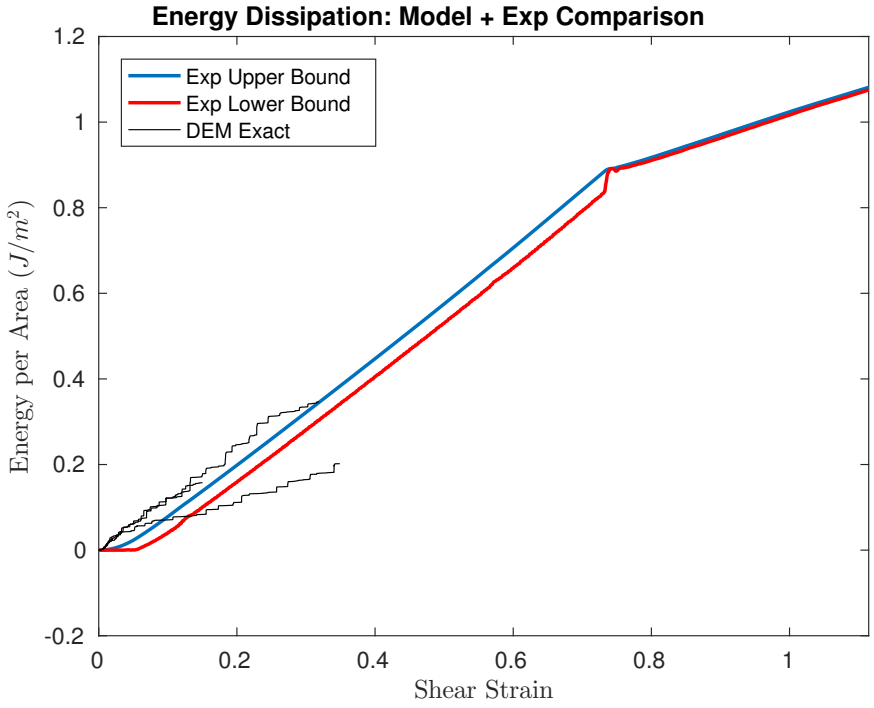


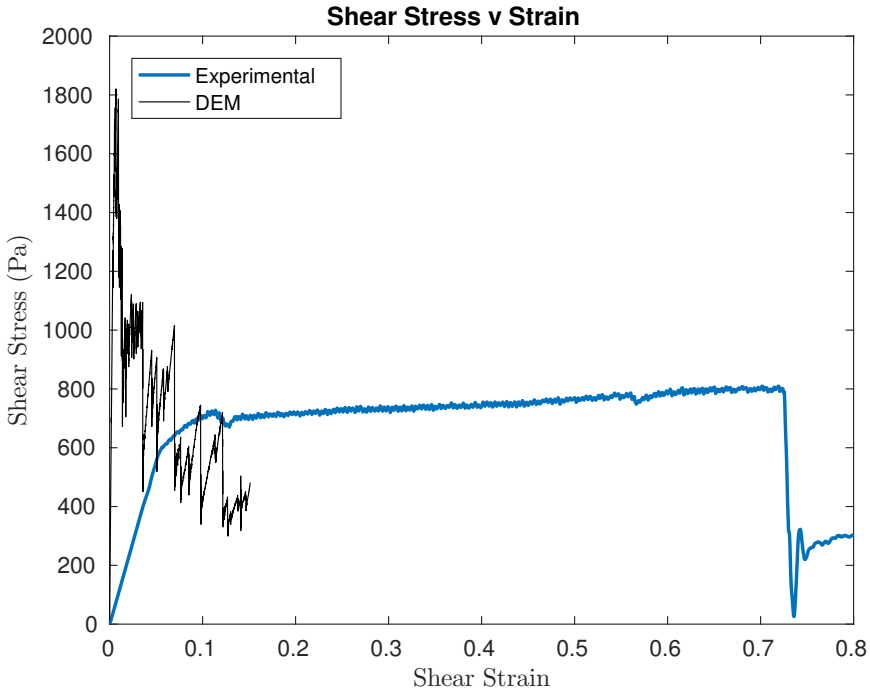

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.087

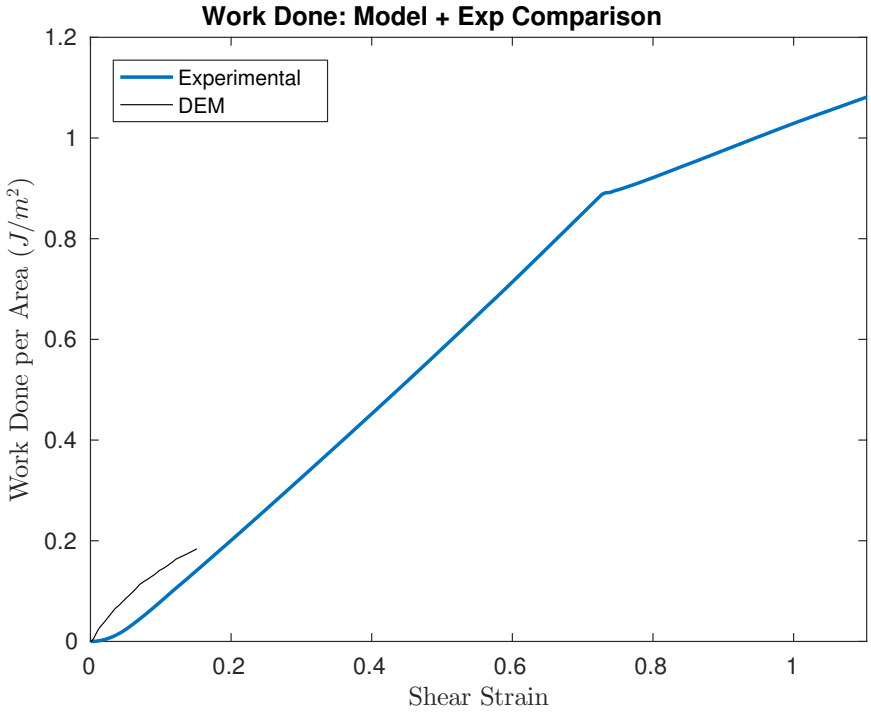

Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
Simulation Time
90th Pctile
10th Pctile

Test 20170127: With Buried Surface Hoar

Experimental vs Numerical Comparison

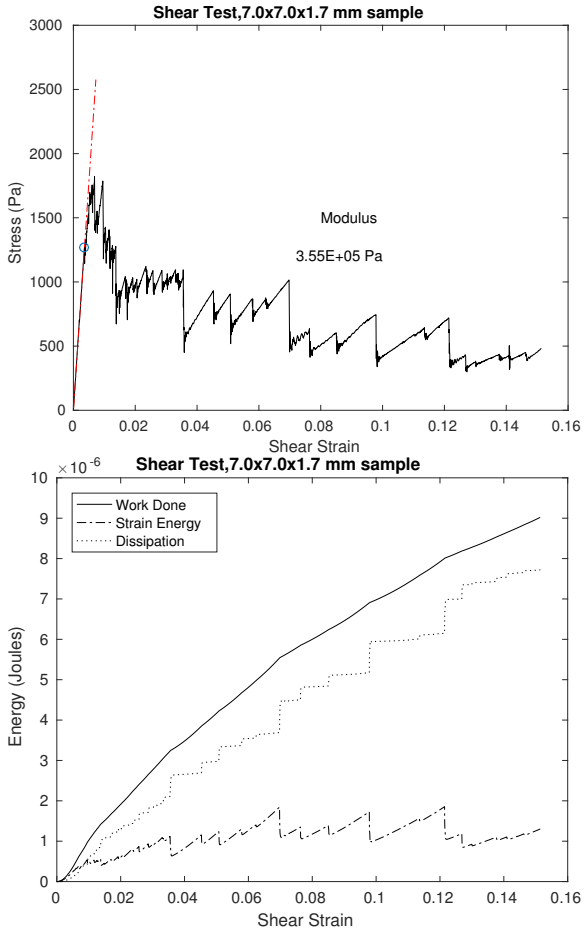


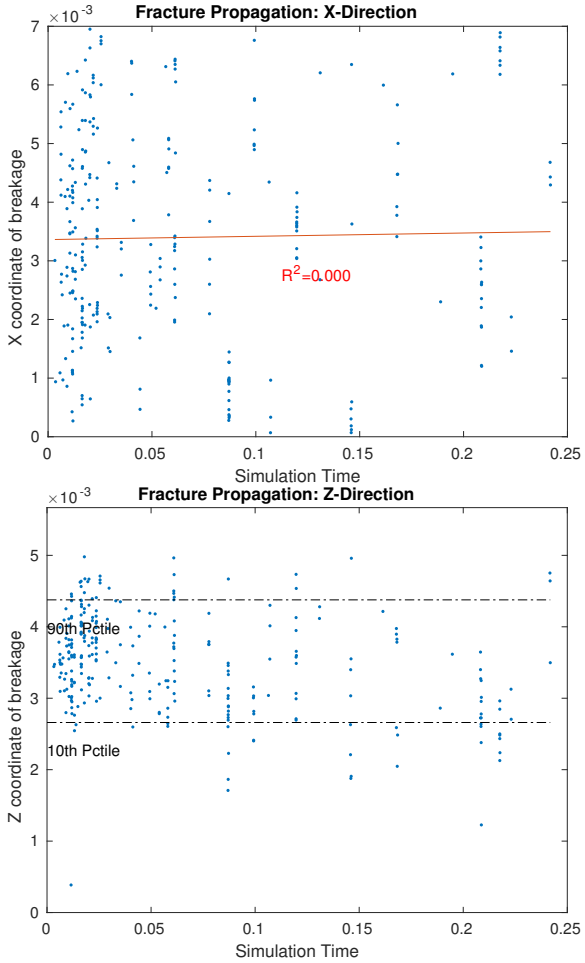

Shear Stress v Strain
Experimental
DEM
Shear Stress (Pa)
Shear Strain
0
200
400
600
800
1000
1200
1400
1600
1800
2000
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


Work Done: Model + Exp Comparison
Experimental
DEM
Work Done per Area (J/m^2)
Shear Strain
0
0.2
0.4
0.6
0.8
1
1.2
0
0.2
0.4
0.6
0.8
1

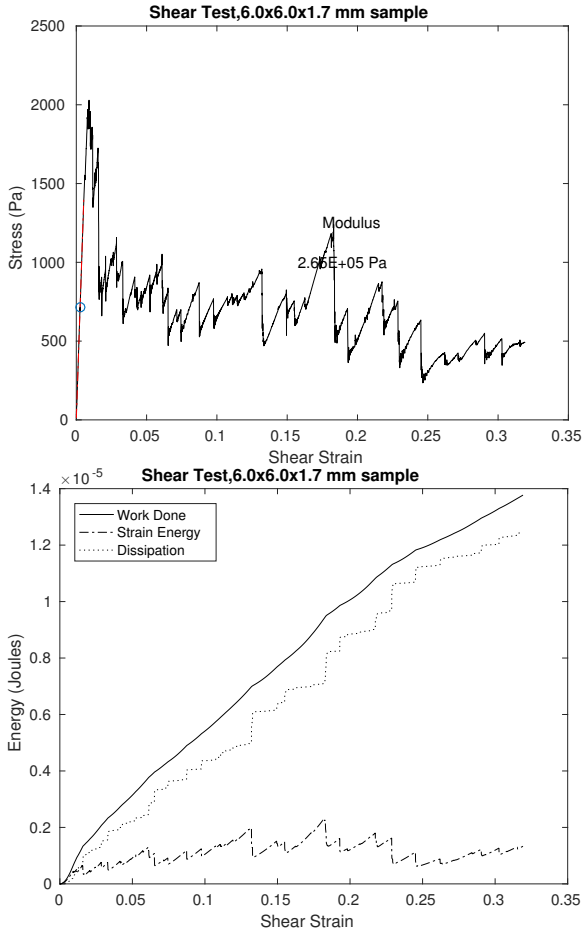
Detailed Numerical Results

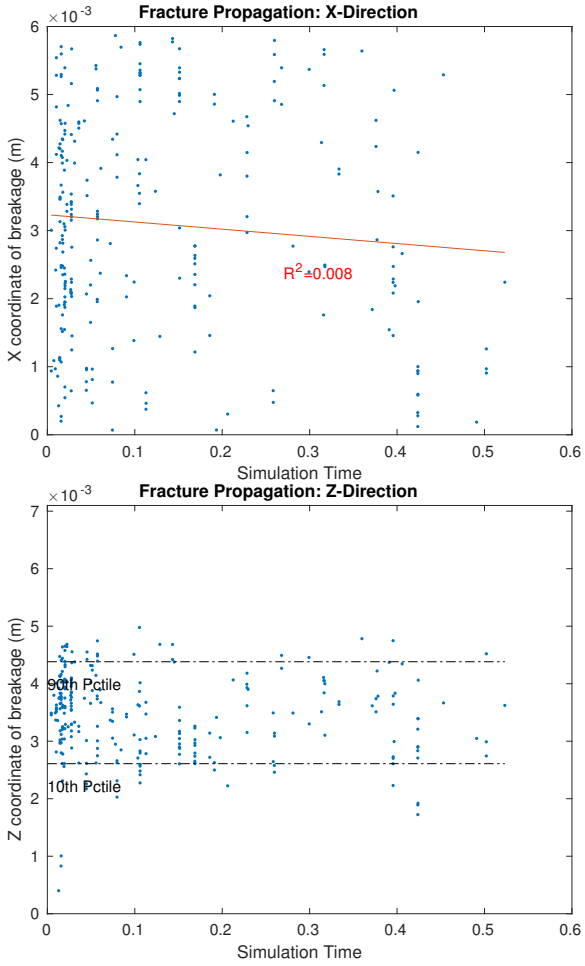
Numerical Subsample 7mm1:



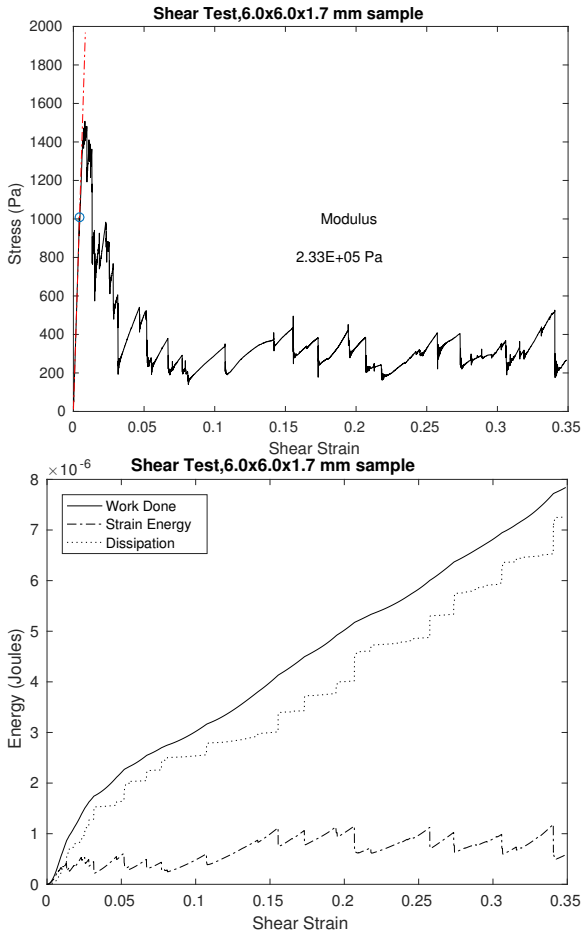

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.000
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

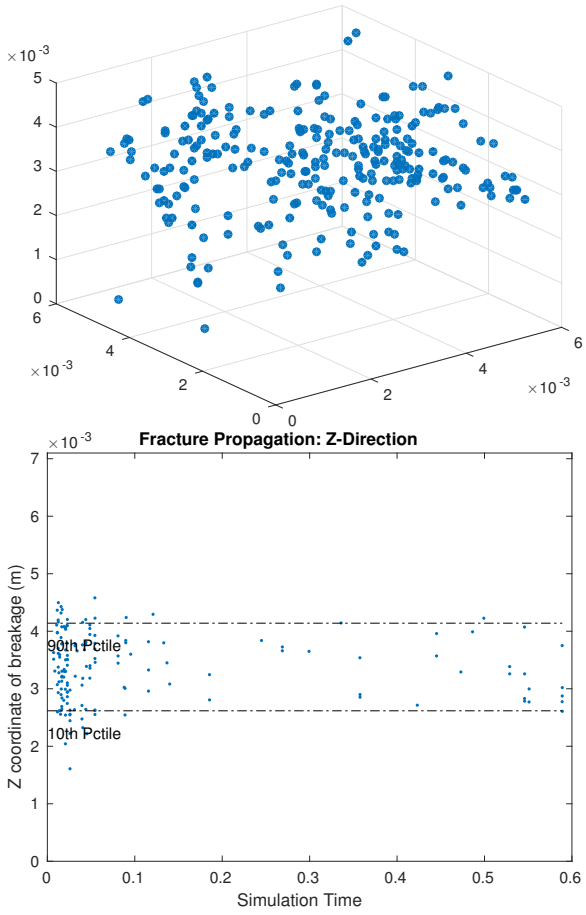
Numerical Subsample 6mm1:




Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage (m)
Simulation Time
R^2 =0.008
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage (m)
Simulation Time
90th Pctile
10th Pctile

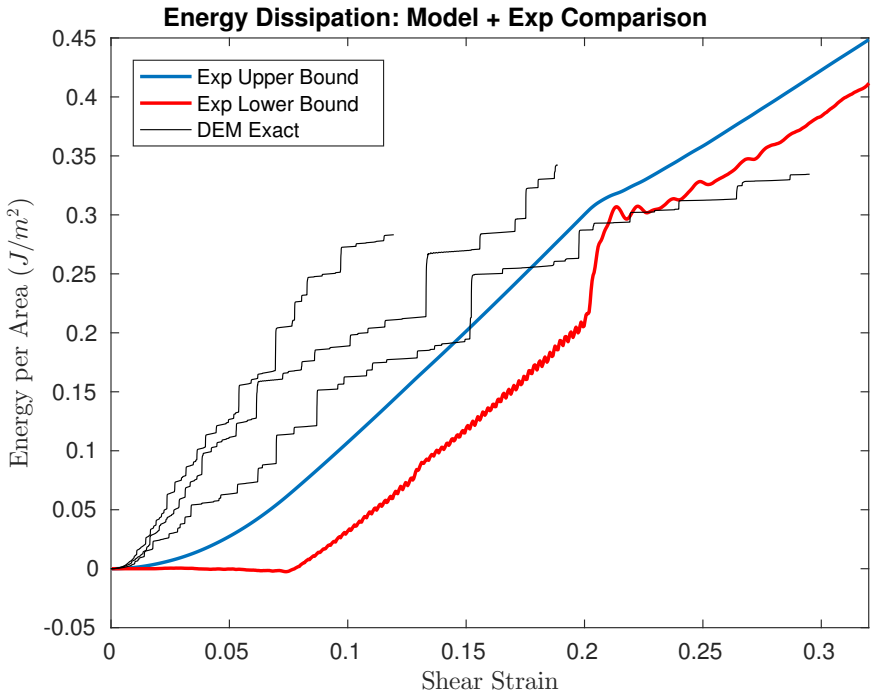
Numerical Subsample 6mm2:

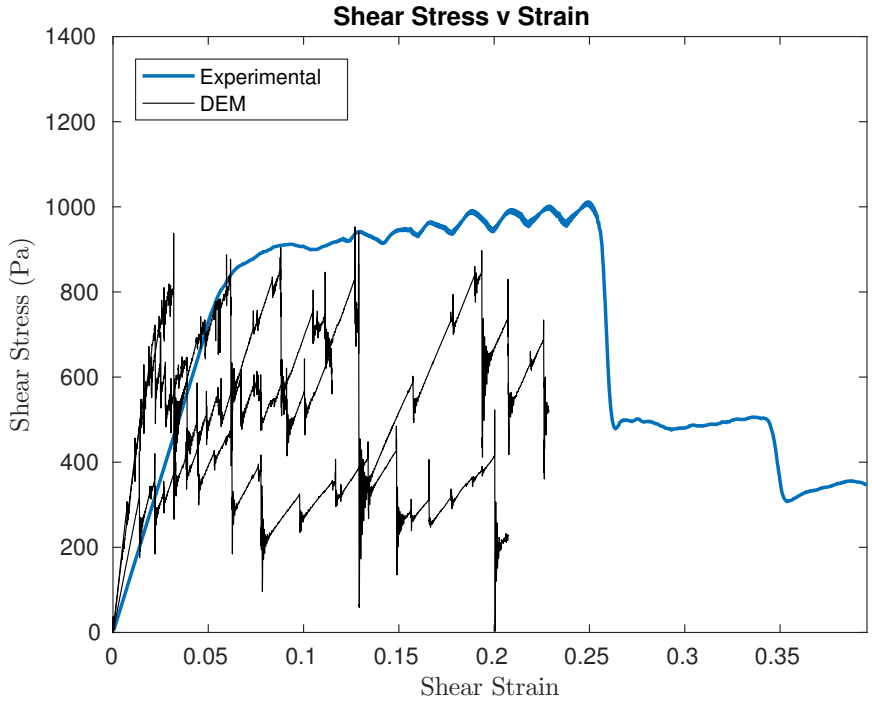


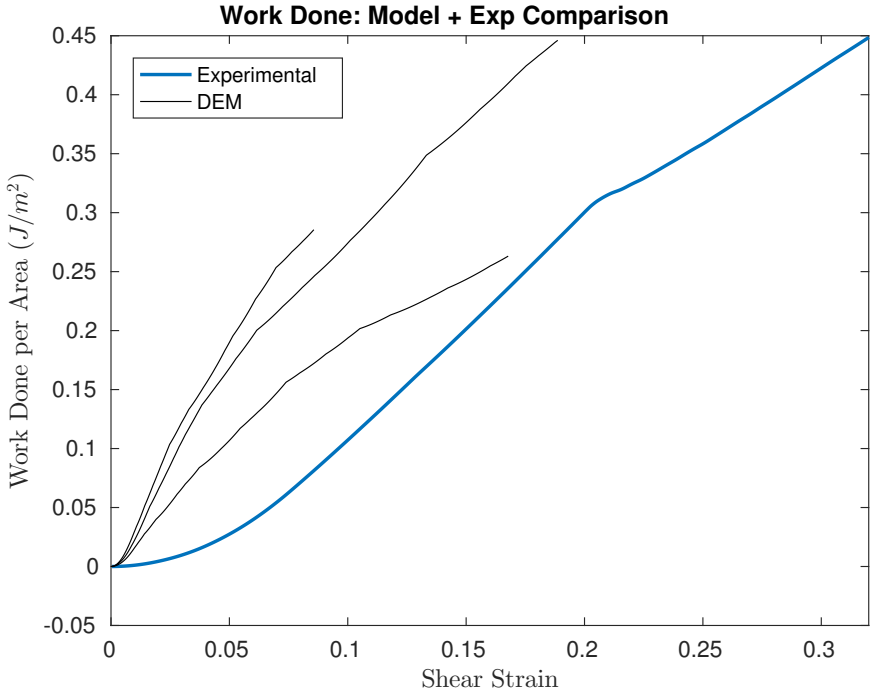

×10^-3
5
4
3
2
1
0
6
4
2
0
×10^-3
0
2
4
6
×10^-3
Fracture Propagation: Z-Direction
×10^-3
7
6
5
4
3
2
1
0
Z coordinate of breakage (m)
90th Pctile
10th Pctile
0
0.1
0.2
0.3
0.4
0.5
0.6
Simulation Time

Test 20170131: With Buried Surface Hoar

Experimental vs Numerical Comparison

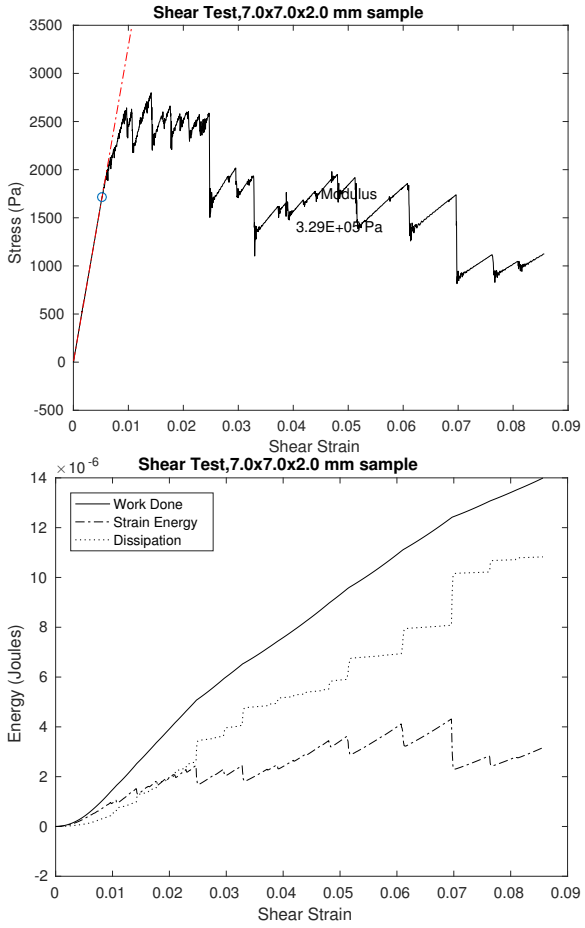


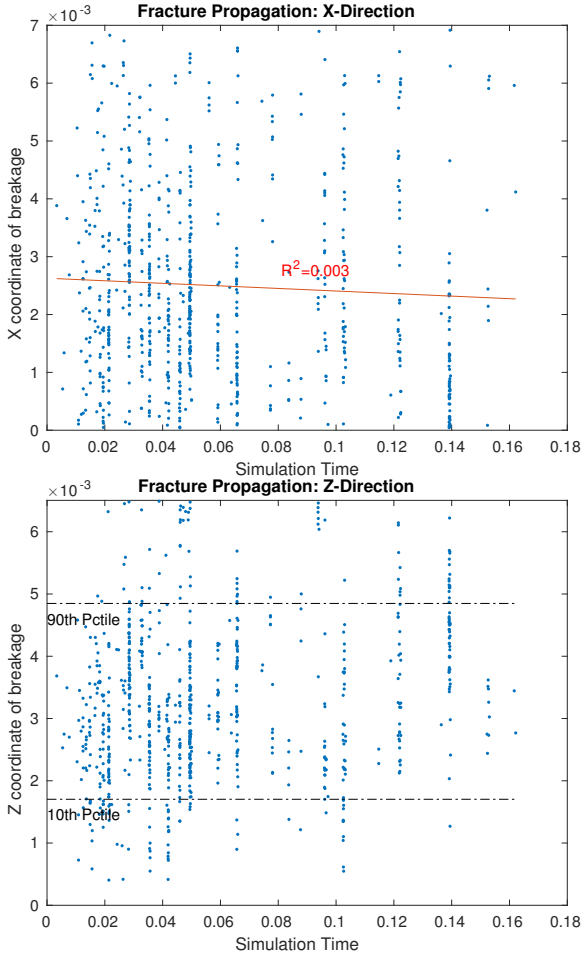

Shear Stress v Strain
Experimental
DEM
Shear Stress (Pa)
Shear Strain
0
200
400
600
800
1000
1200
1400
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35


Work Done: Model + Exp Comparison
Experimental
DEM
Work Done per Area (J/m^2)
Shear Strain
-0.05
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0
0.05
0.1
0.15
0.2
0.25
0.3

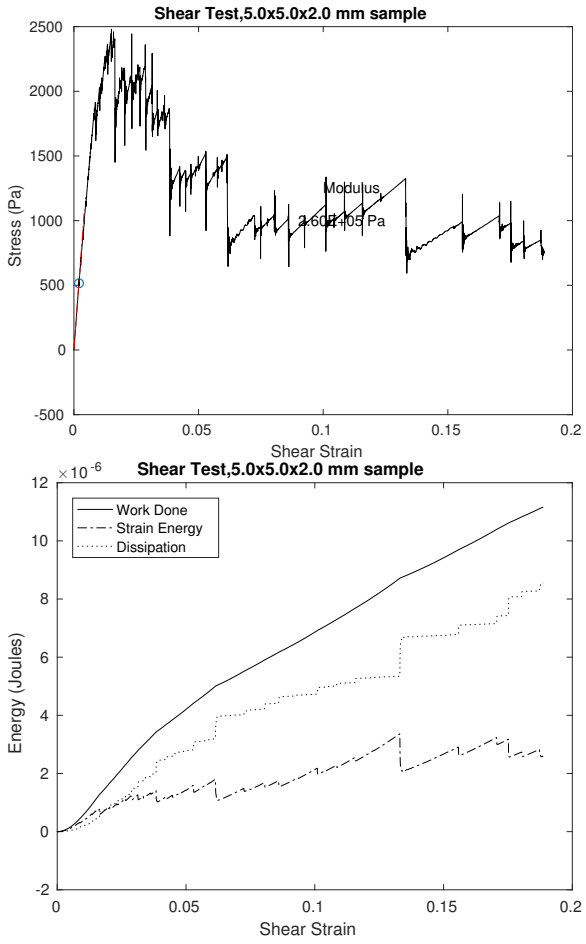
Detailed Numerical Results

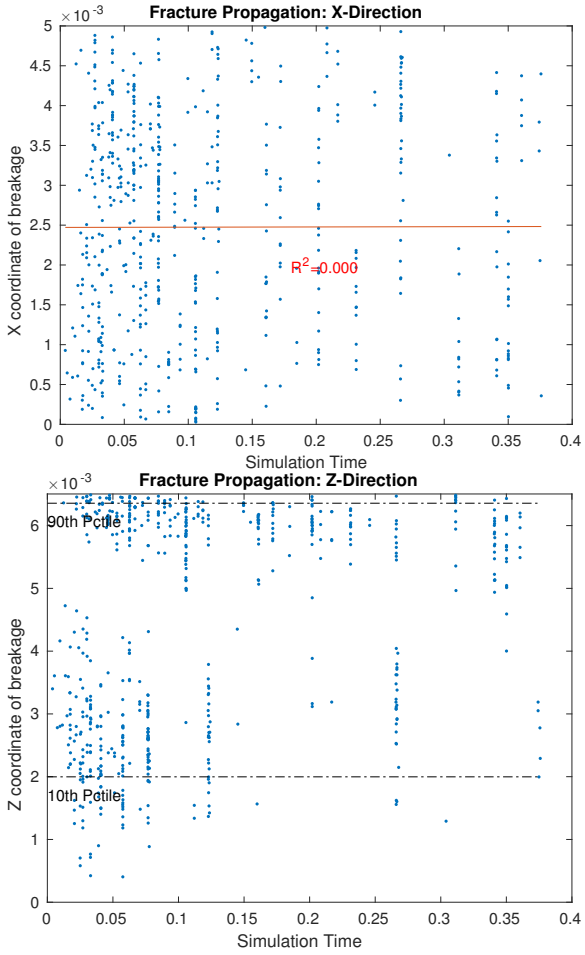
Numerical Subsample 7mm1:




Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.003
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
Simulation Time
90th Pctile
10th Pctile

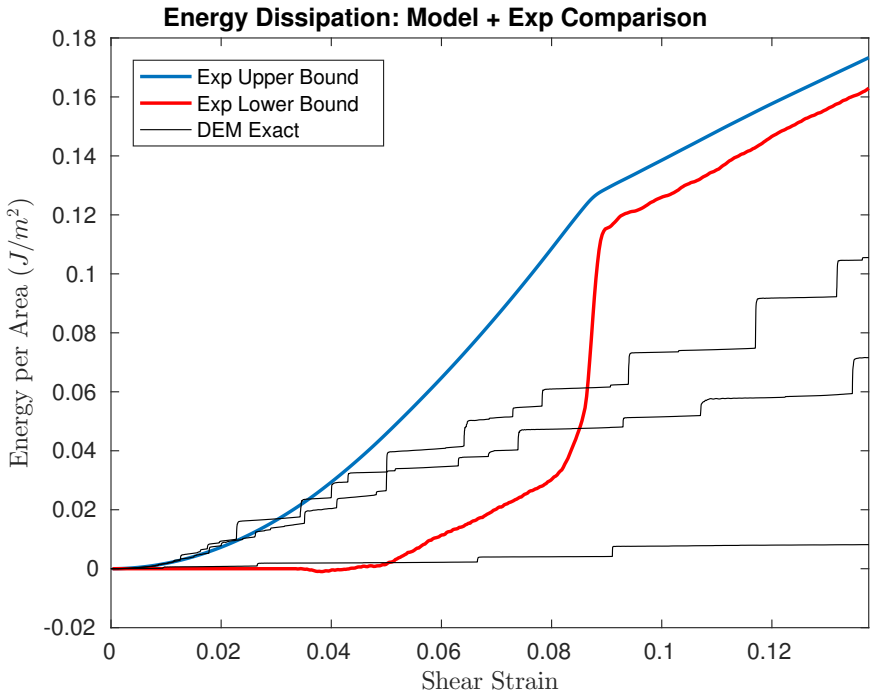
Numerical Subsample 5mm1:

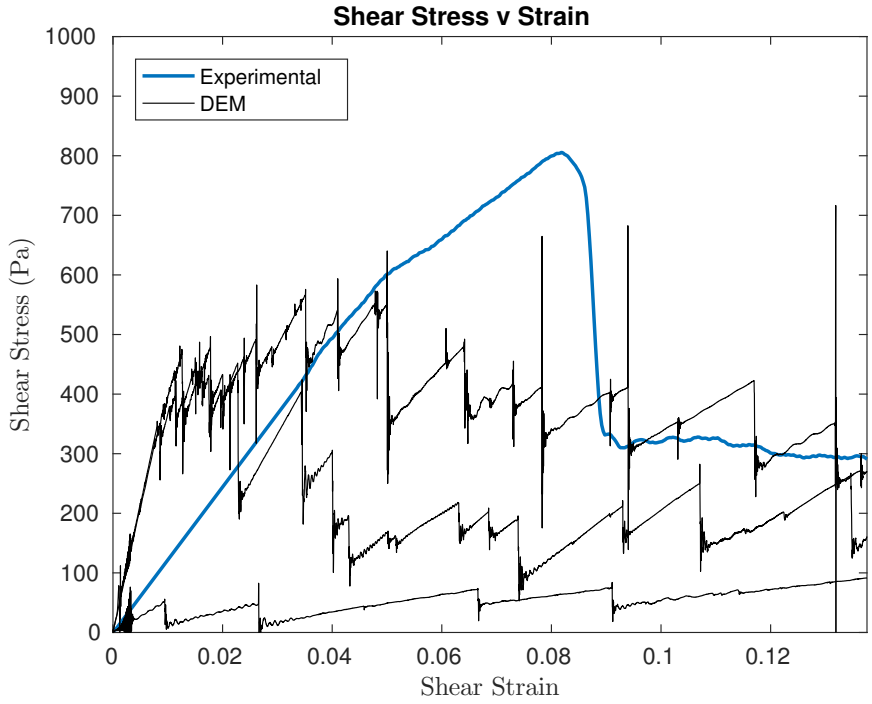


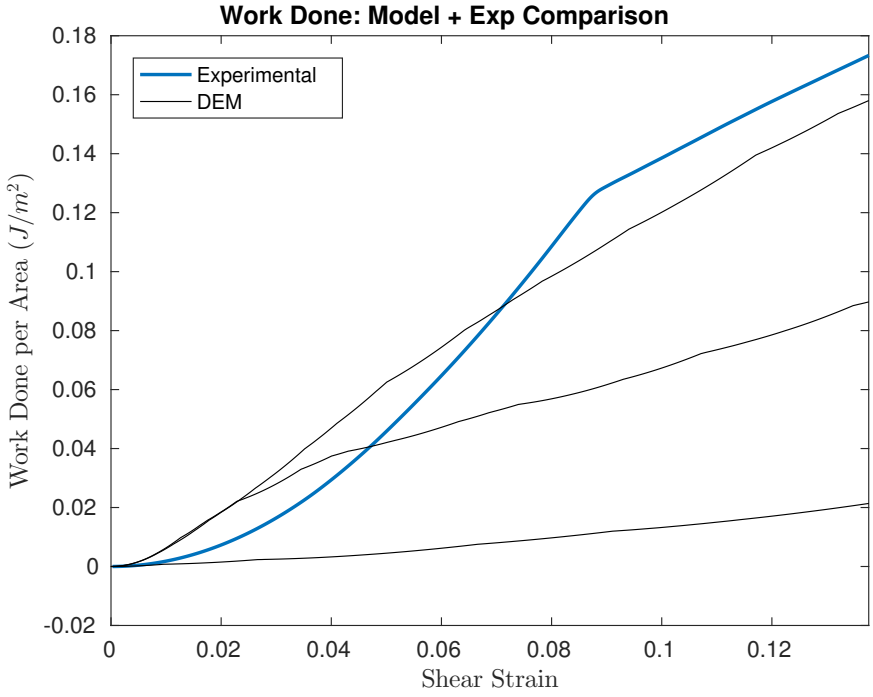

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2 = 0.000
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
Simulation Time
90th Pctile
10th Pctile

Test 20170224: With Buried Surface Hoar

Experimental vs Numerical Comparison

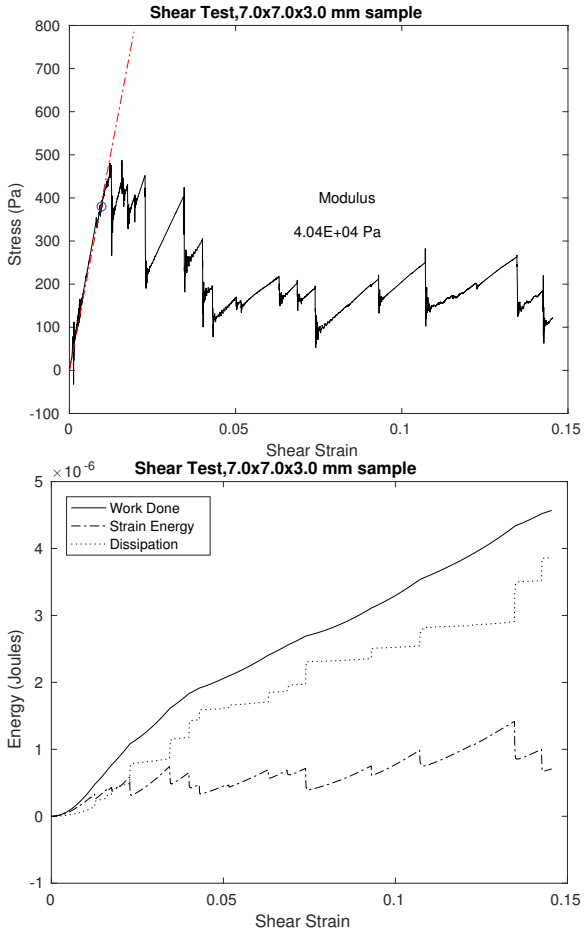


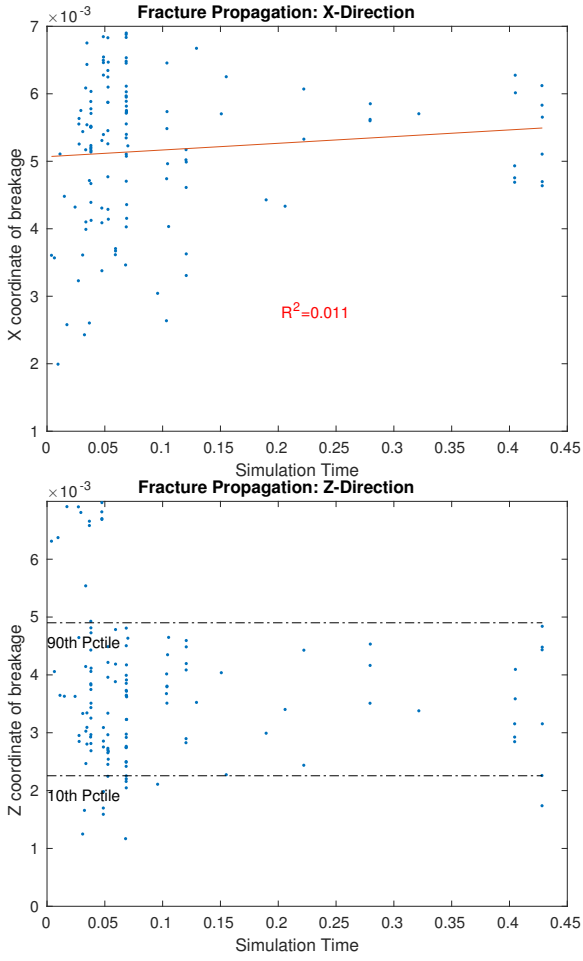

Shear Stress v Strain
Experimental
DEM
Shear Stress (Pa)
Shear Strain
0
100
200
300
400
500
600
700
800
900
1000
0.02
0.04
0.06
0.08
0.1
0.12


Work Done: Model + Exp Comparison
Experimental
DEM
Work Done per Area (J/m^2)
Shear Strain
-0.02
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18

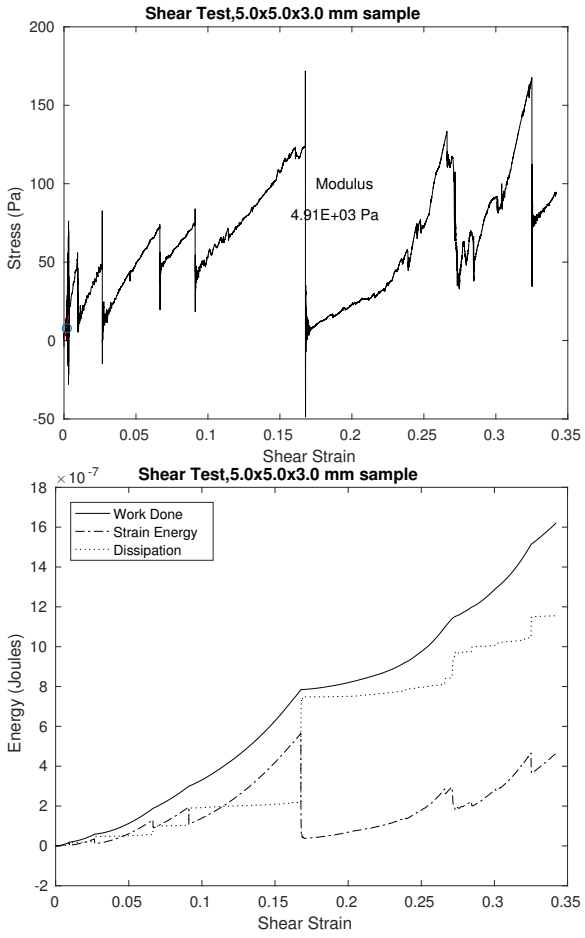
Detailed Numerical Results

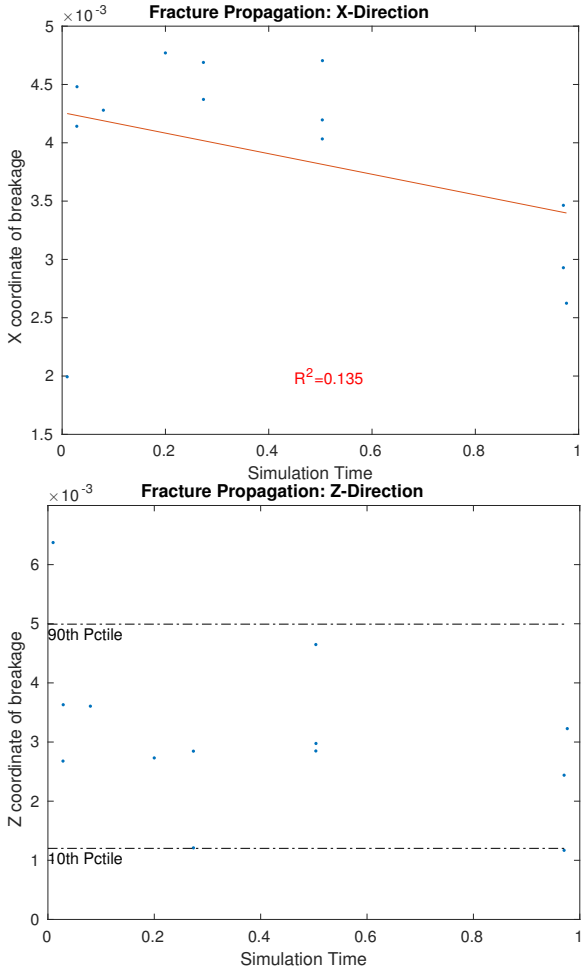
Numerical Subsample 7mm1:



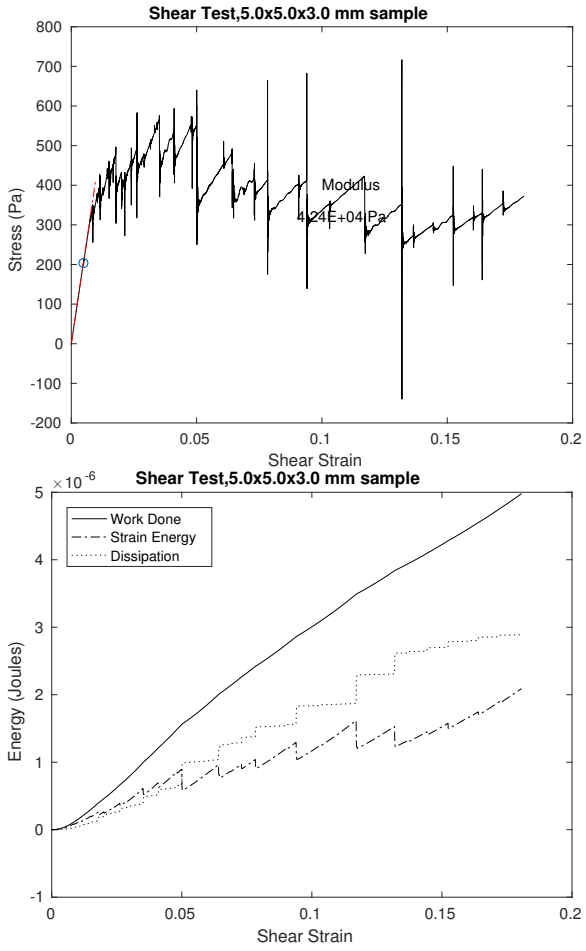

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.011
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

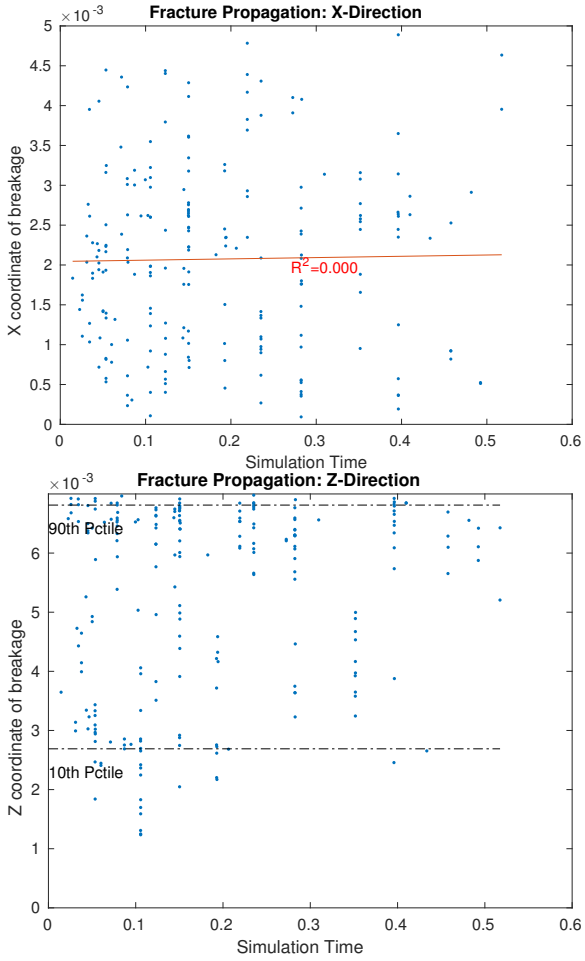
Numerical Subsample 5mm1:




Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.135
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

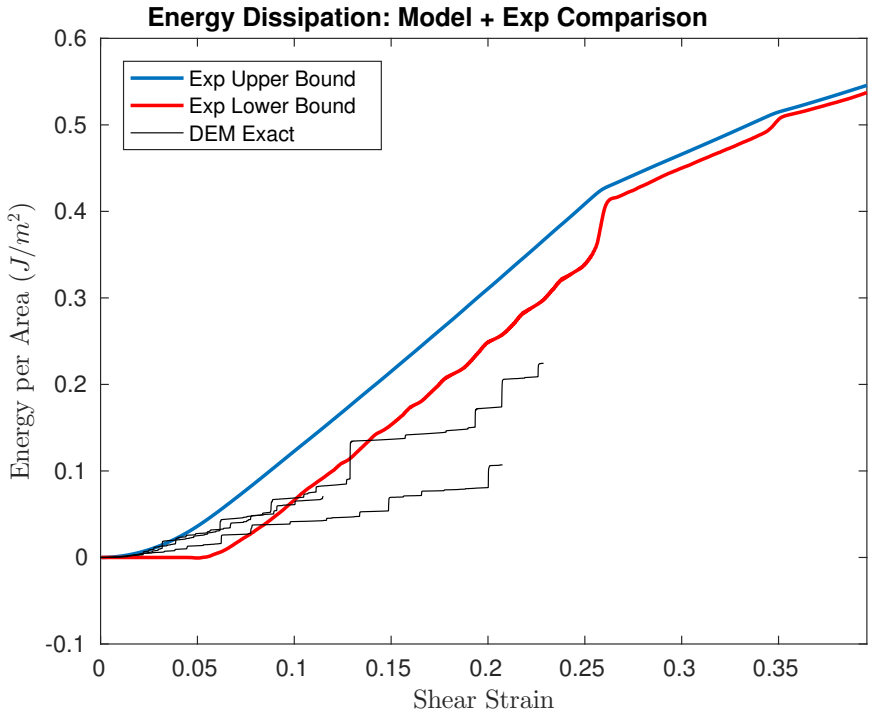
Numerical Subsample 5mm2:

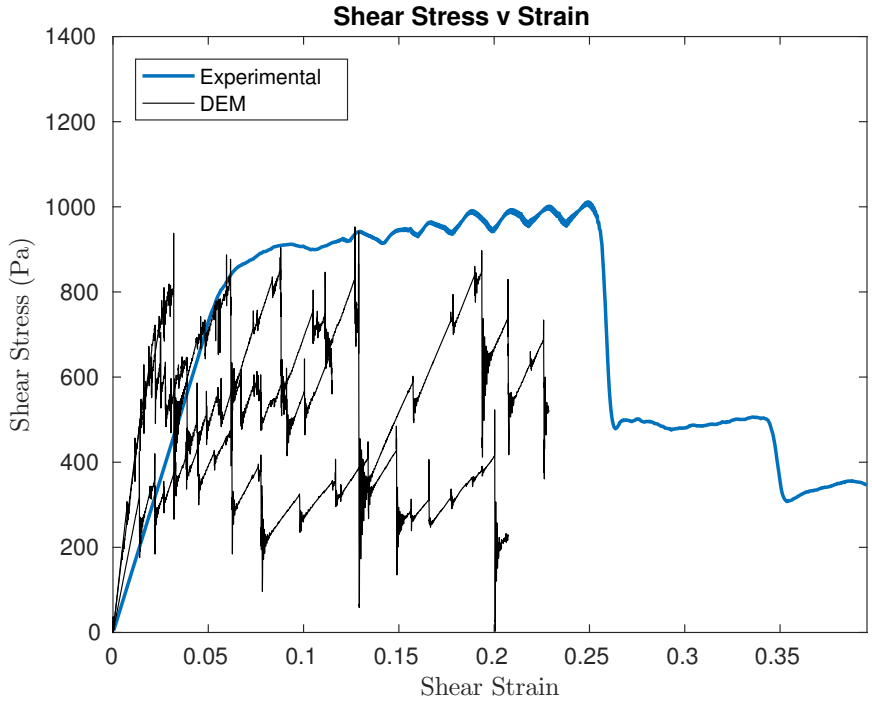


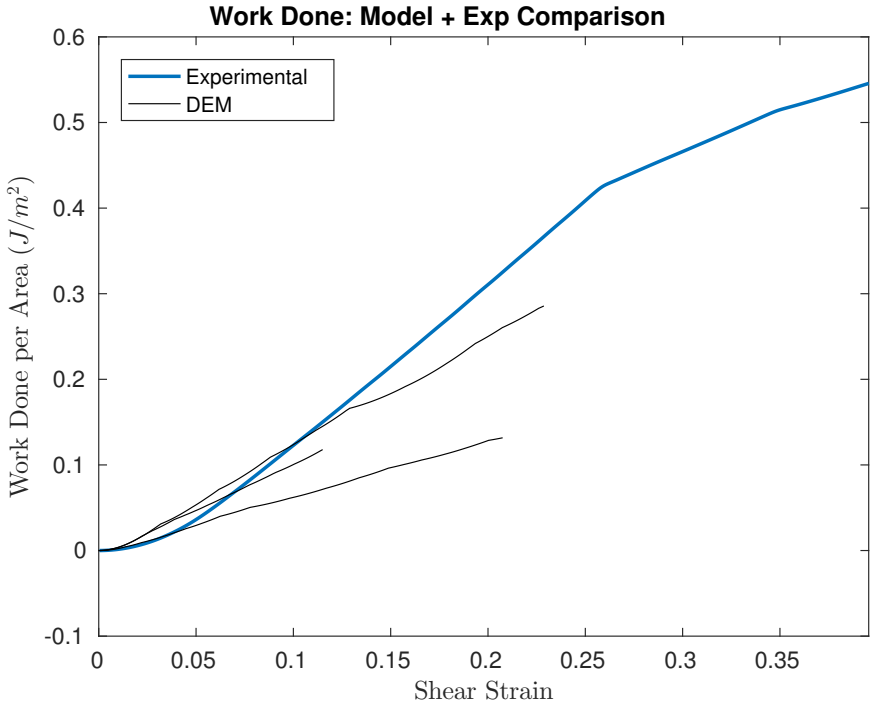

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.000
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
Simulation Time
90th Pctile
10th Pctile

Test 20170301: With Buried Surface Hoar

Experimental vs Numerical Comparison

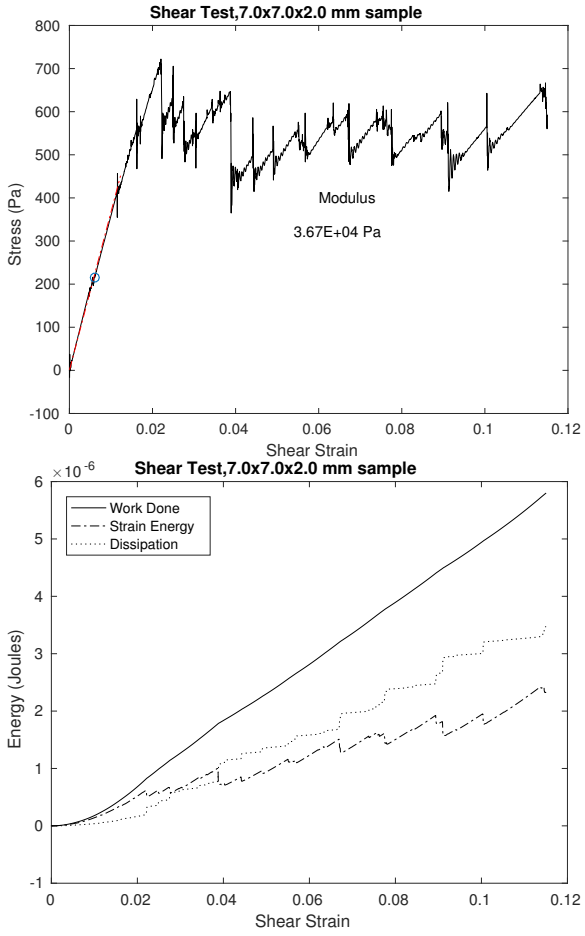


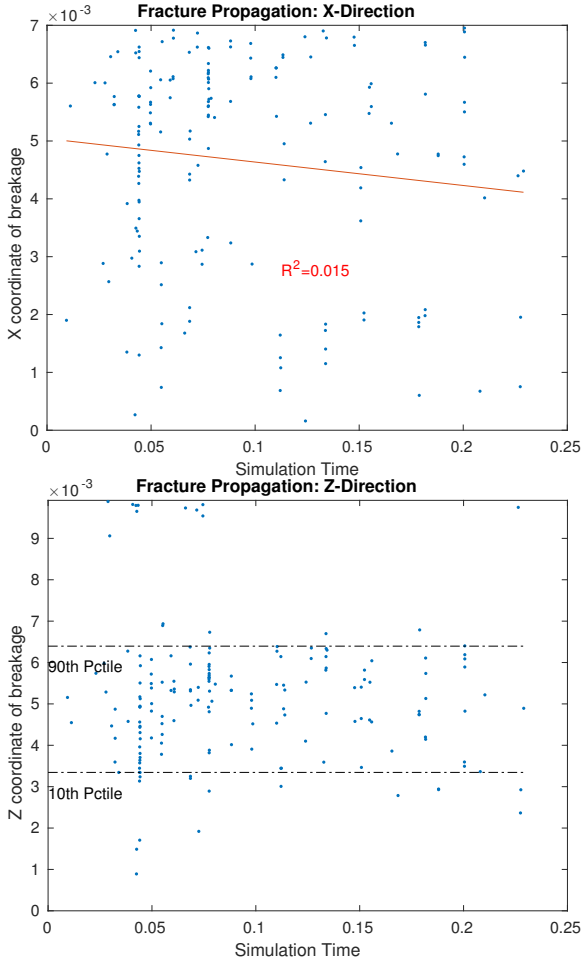

Shear Stress v Strain
Experimental
DEM
Shear Stress (Pa)
Shear Strain
0
200
400
600
800
1000
1200
1400
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35


Work Done: Model + Exp Comparison
Experimental
DEM
Work Done per Area (J/m^2)
Shear Strain
-0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35

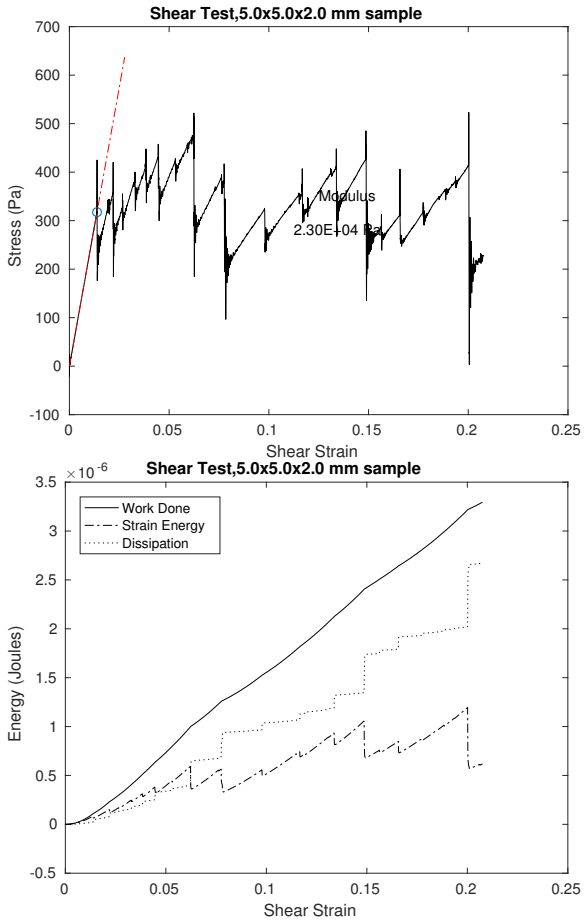
Detailed Numerical Results

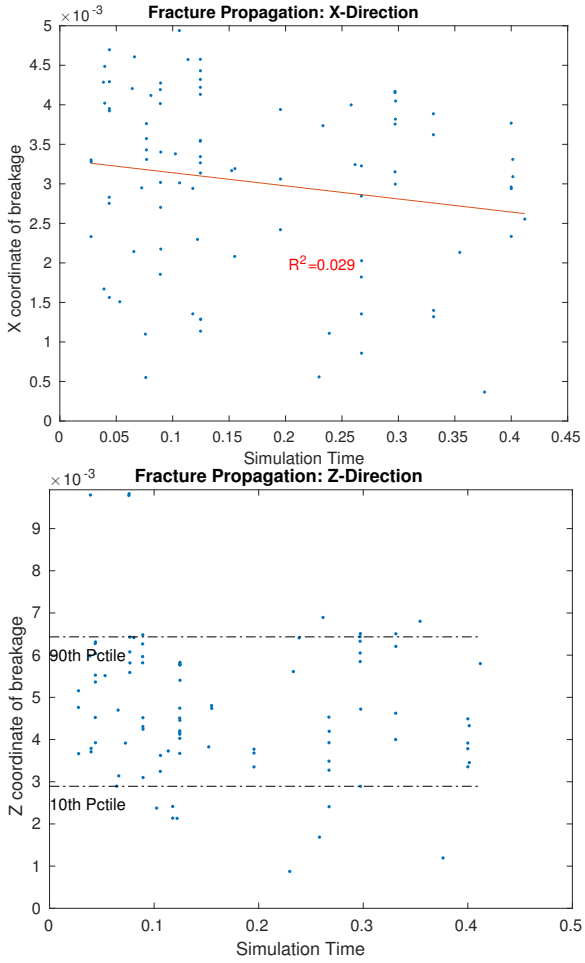
Numerical Subsample 7mm1:



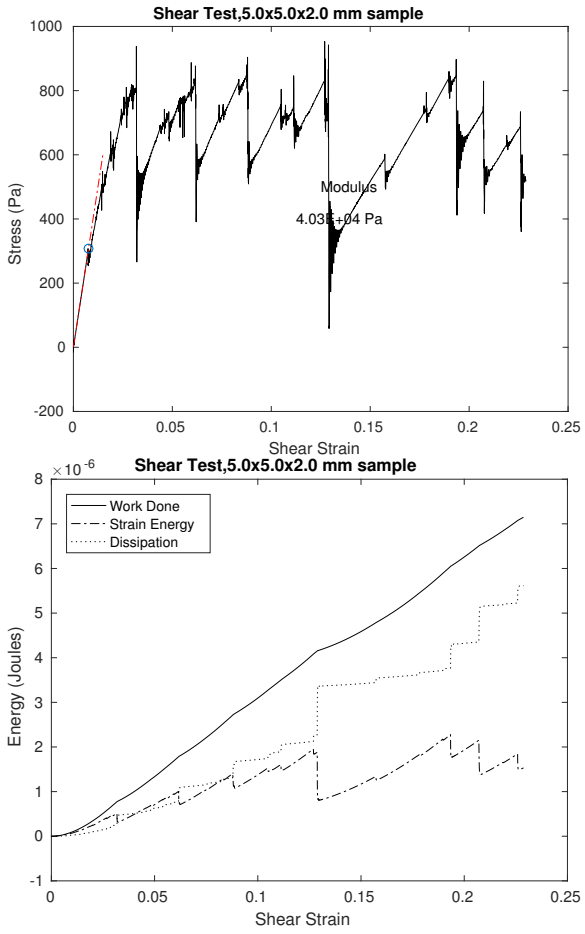

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.015
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

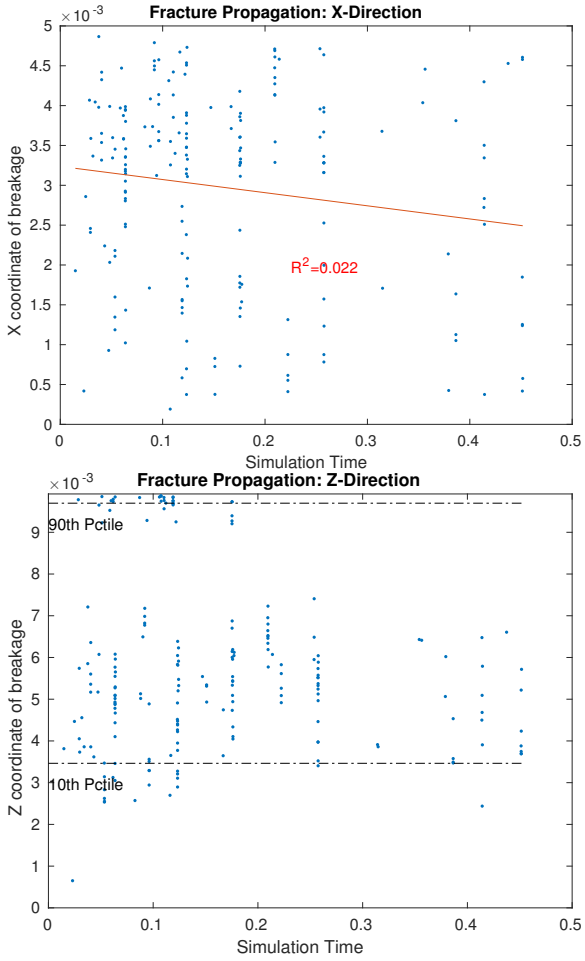
Numerical Subsample 5mm1:




Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.029
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
Simulation Time
90th Pctile
10th Pctile

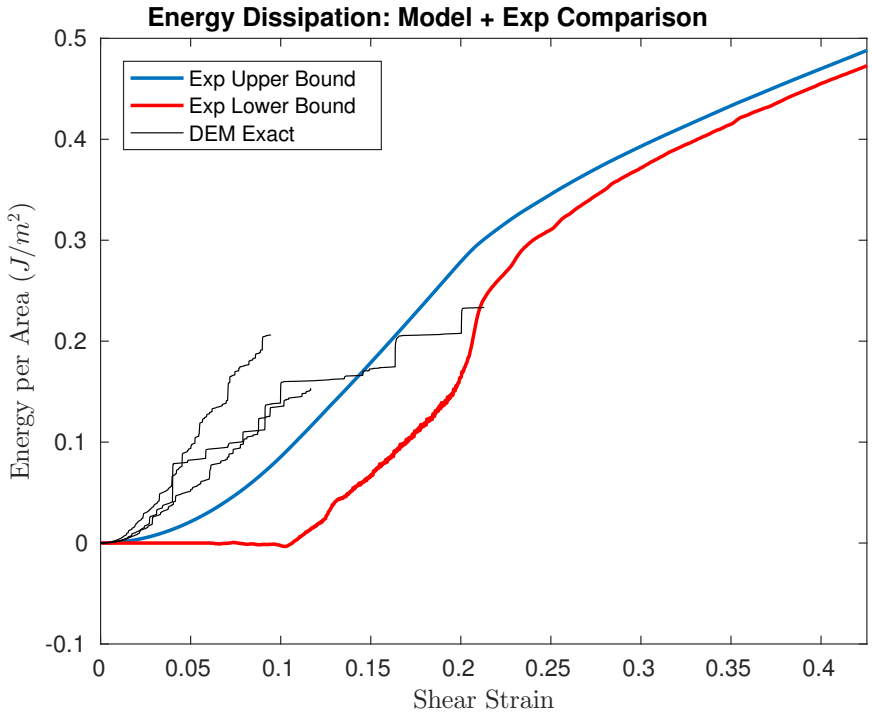
Numerical Subsample 5mm2:

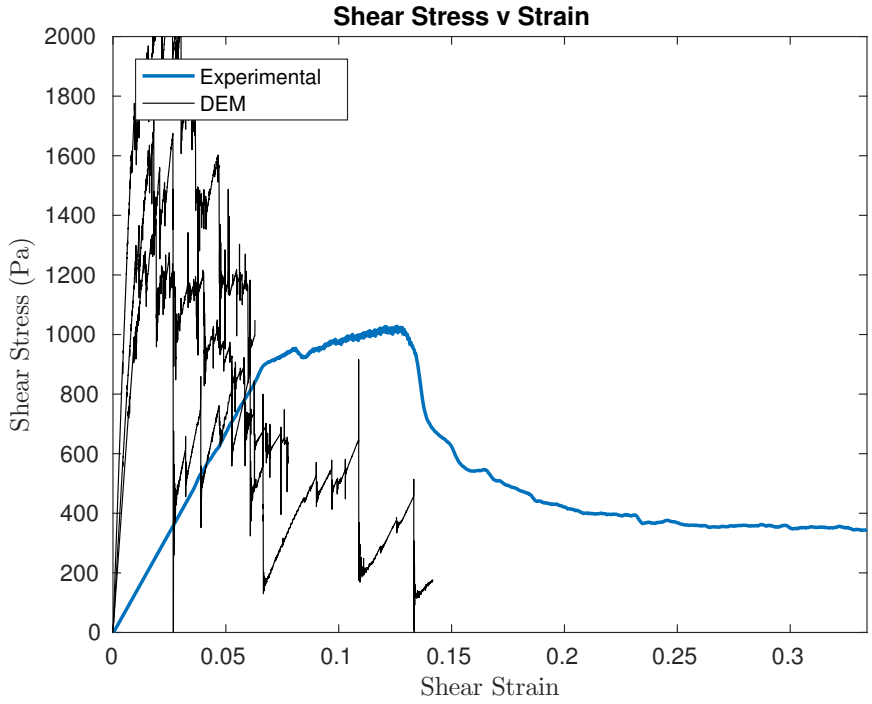


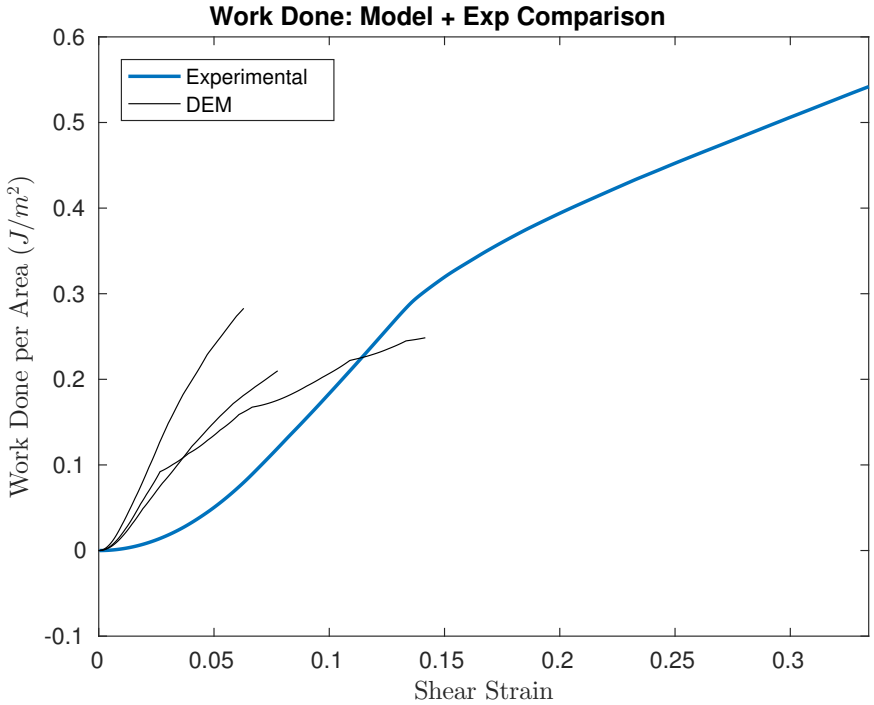

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.022
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
Simulation Time
90th Pctile
10th Pctile

Test 20170303: With Buried Surface Hoar

Experimental vs Numerical Comparison

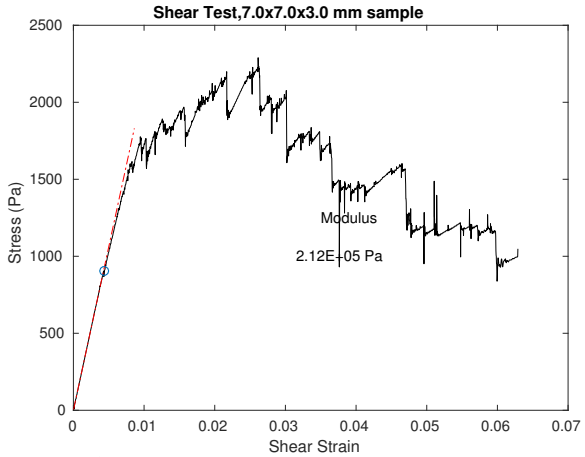


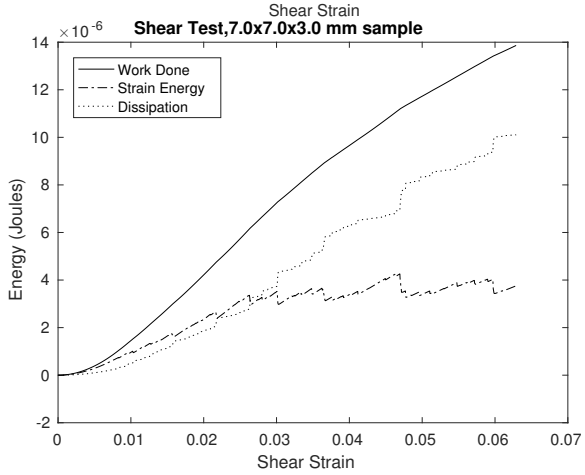

Shear Stress v Strain
Experimental
DEM
Shear Stress (Pa)
Shear Strain
2000
1800
1600
1400
1200
1000
800
600
400
200
0
0
0.05
0.1
0.15
0.2
0.25
0.3

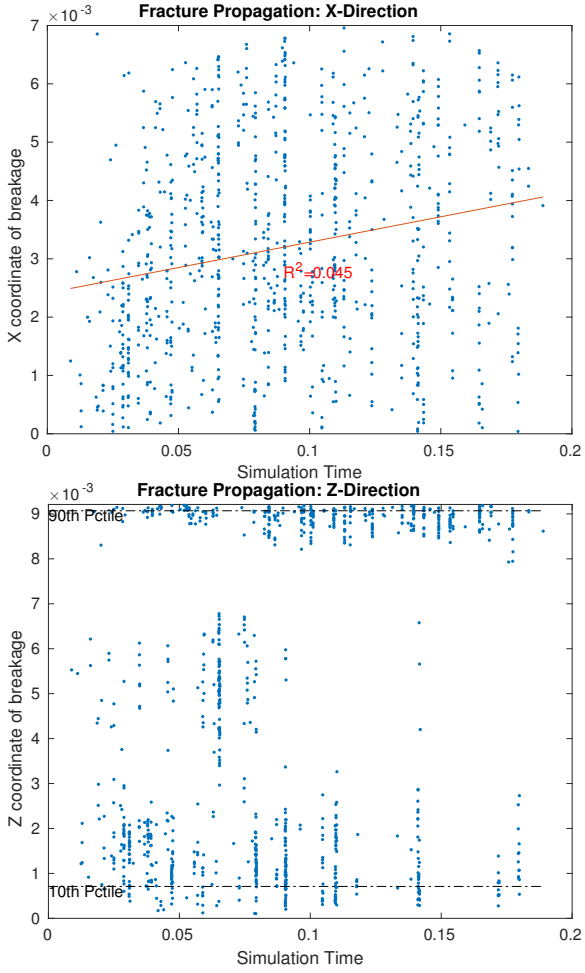

Work Done: Model + Exp Comparison
Experimental
DEM
Work Done per Area (J/m^2)
Shear Strain
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
0
0.05
0.1
0.15
0.2
0.25
0.3

Detailed Numerical Results

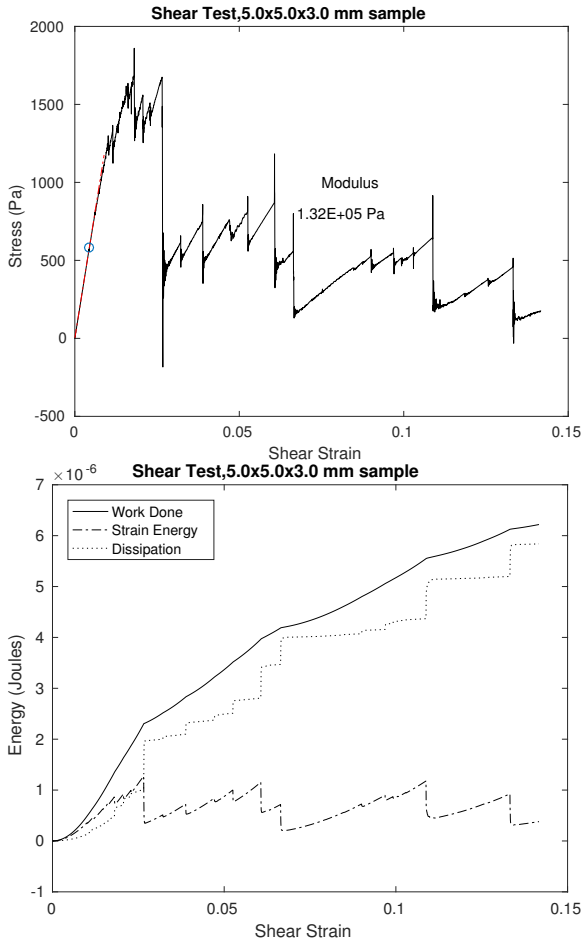
Numerical Subsample 7mm1:

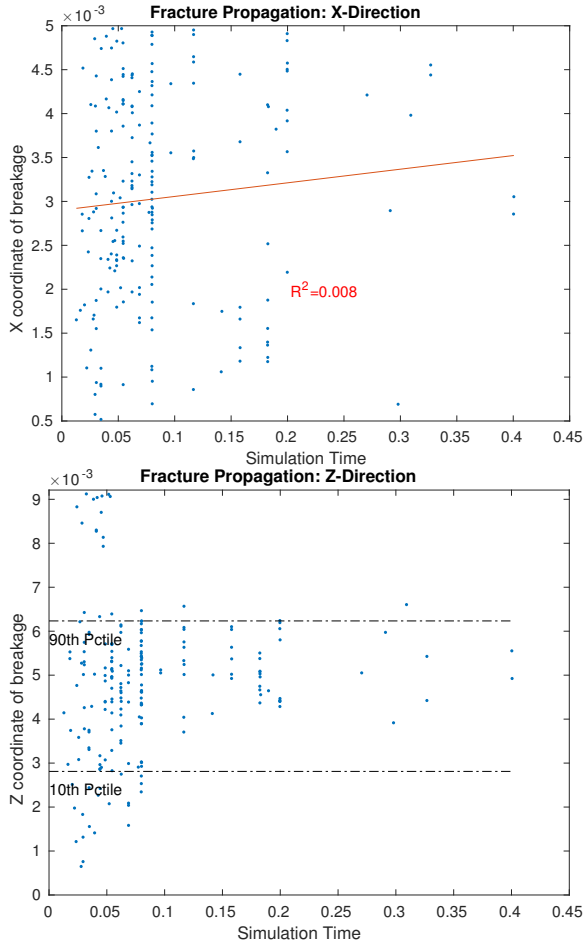




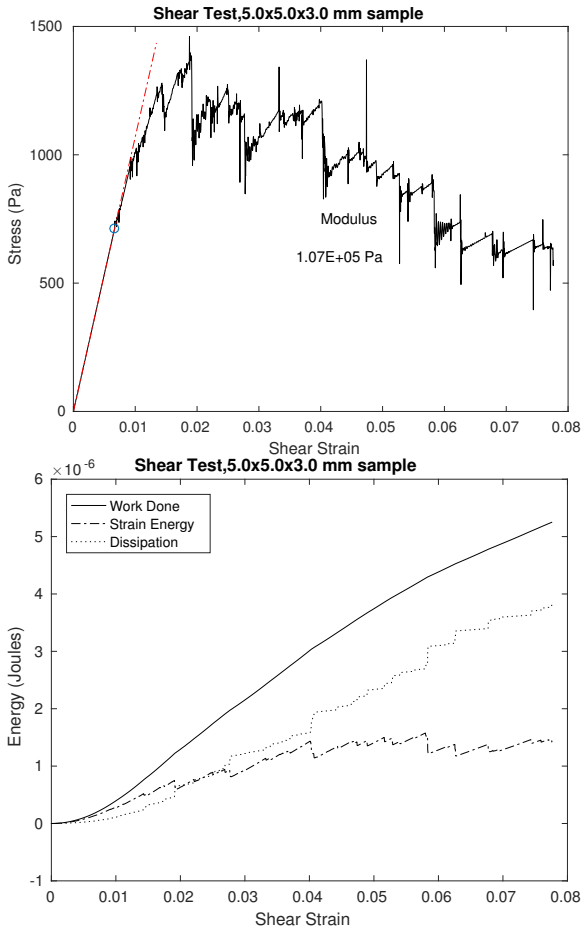

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2 = 0.045
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
Simulation Time
90th Pctile
10th Pctile

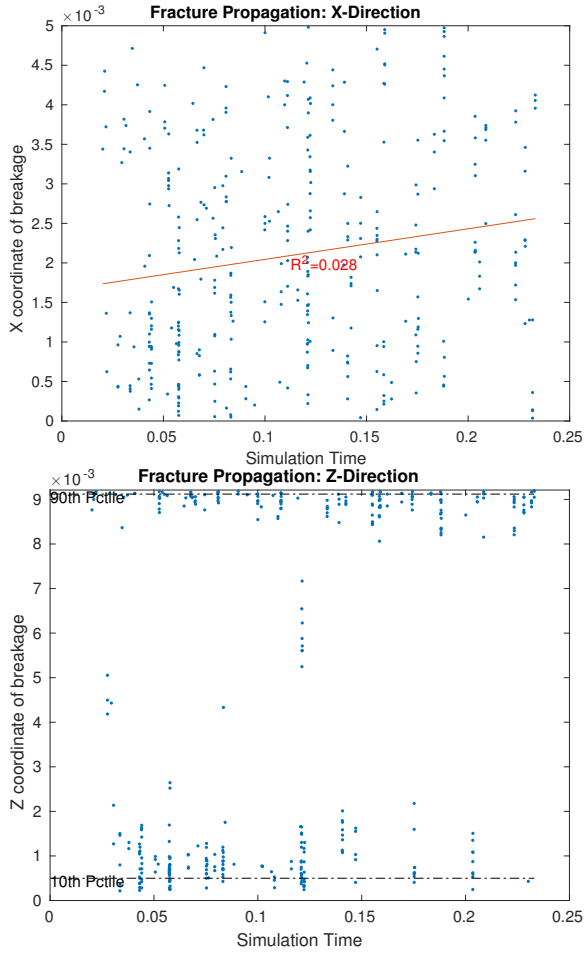
Numerical Subsample 5mm1:




Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.008
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

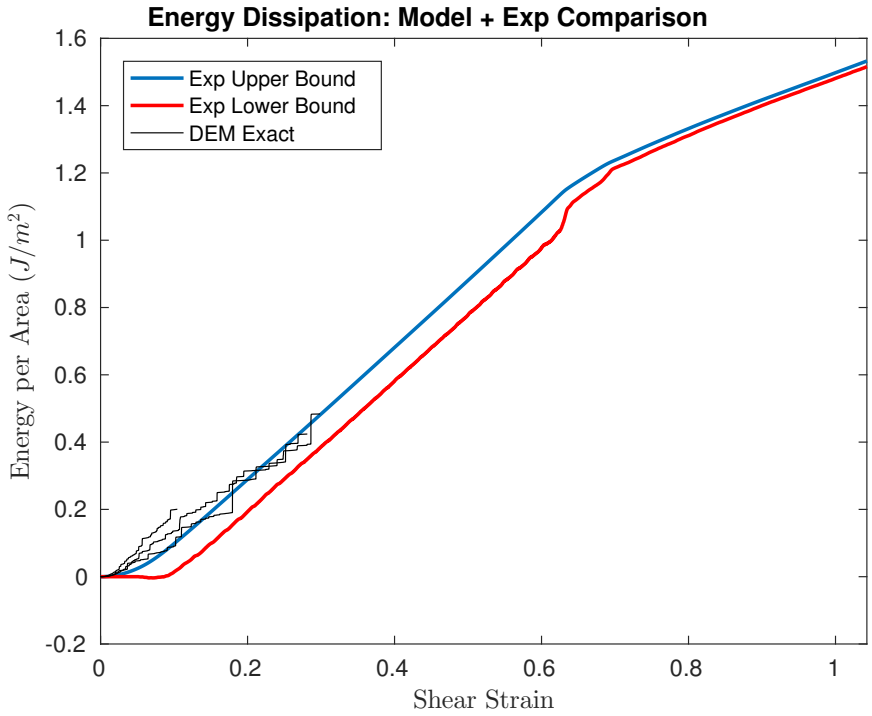
Numerical Subsample 5mm2:

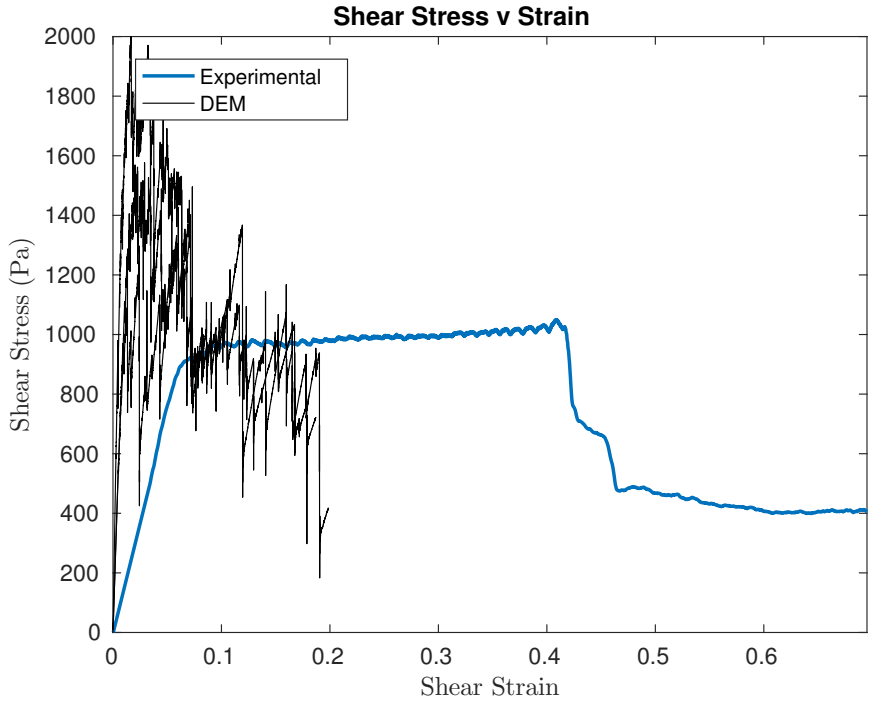


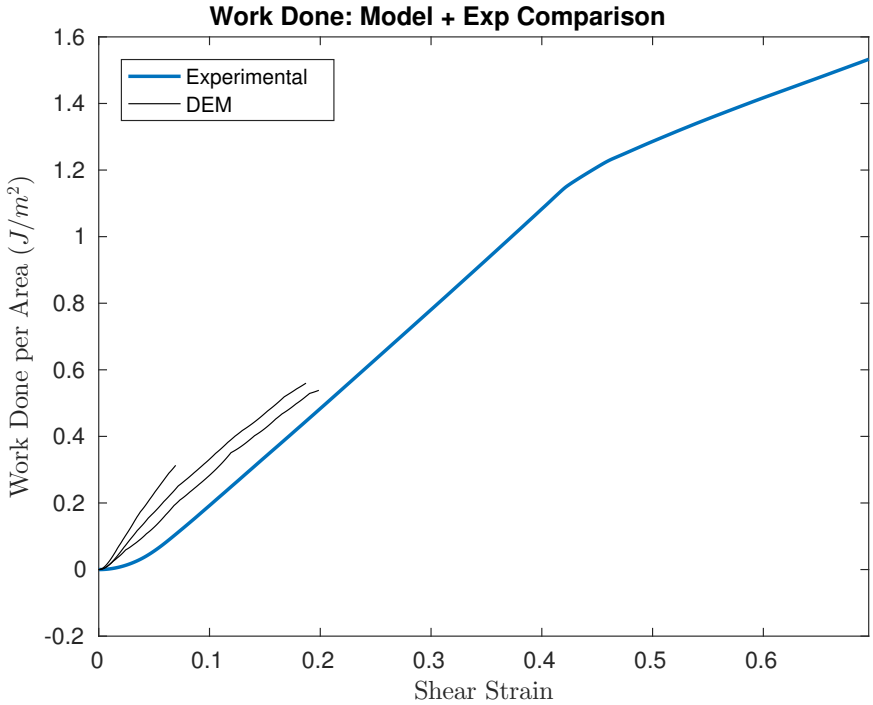

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.028
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
Simulation Time
90th Pctile
10th Pctile

Test 20170306: With Buried Surface Hoar

Experimental vs Numerical Comparison

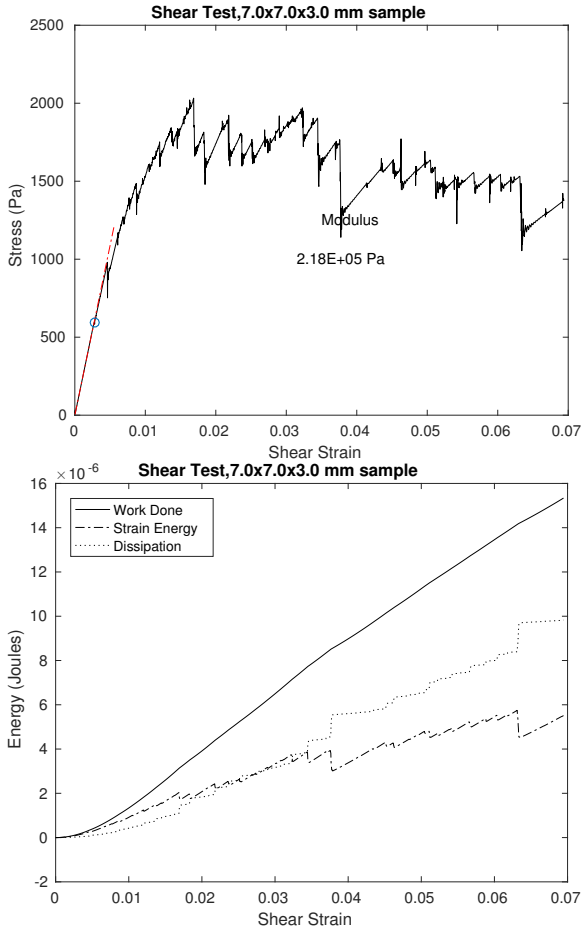


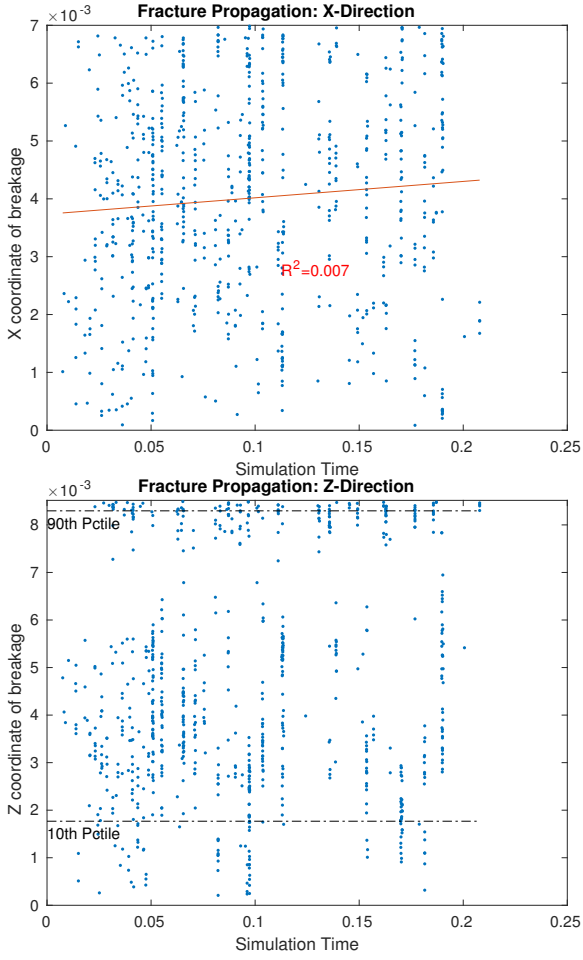

Shear Stress v Strain
Experimental
DEM
Shear Stress (Pa)
Shear Strain
2000
1800
1600
1400
1200
1000
800
600
400
200
0
0
0.1
0.2
0.3
0.4
0.5
0.6


Work Done: Model + Exp Comparison
Experimental
DEM
Work Done per Area (J/m^2)
Shear Strain
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.1
0.2
0.3
0.4
0.5
0.6

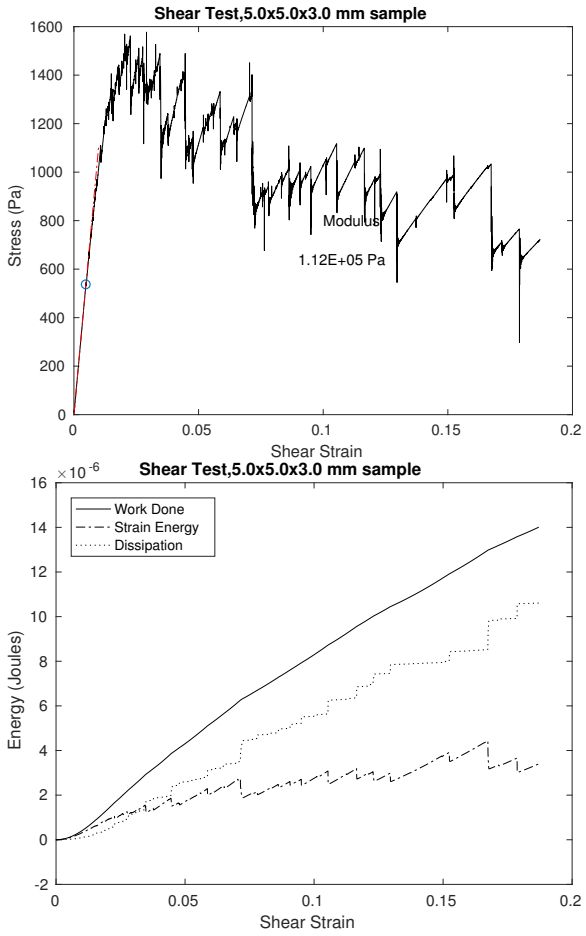
Detailed Numerical Results

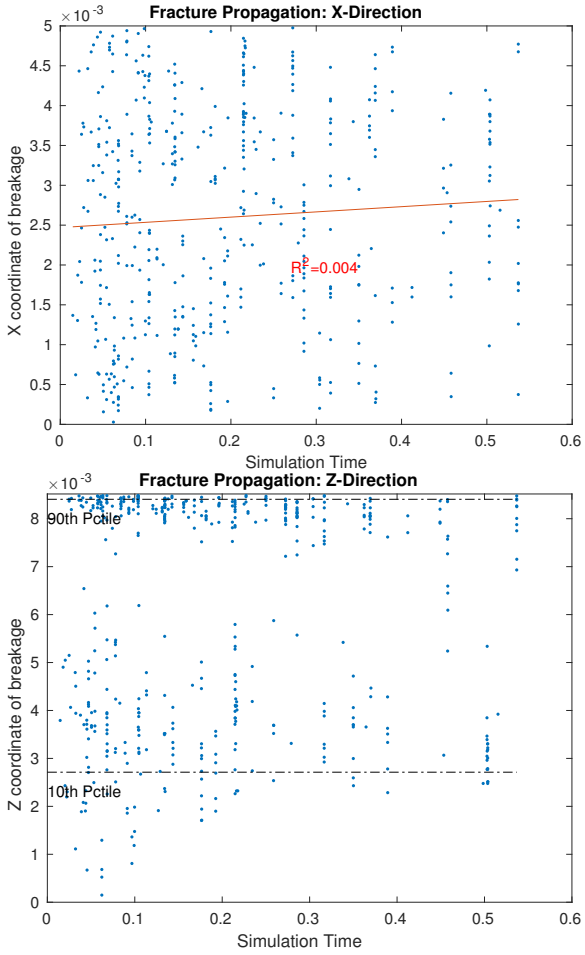
Numerical Subsample 7mm1:



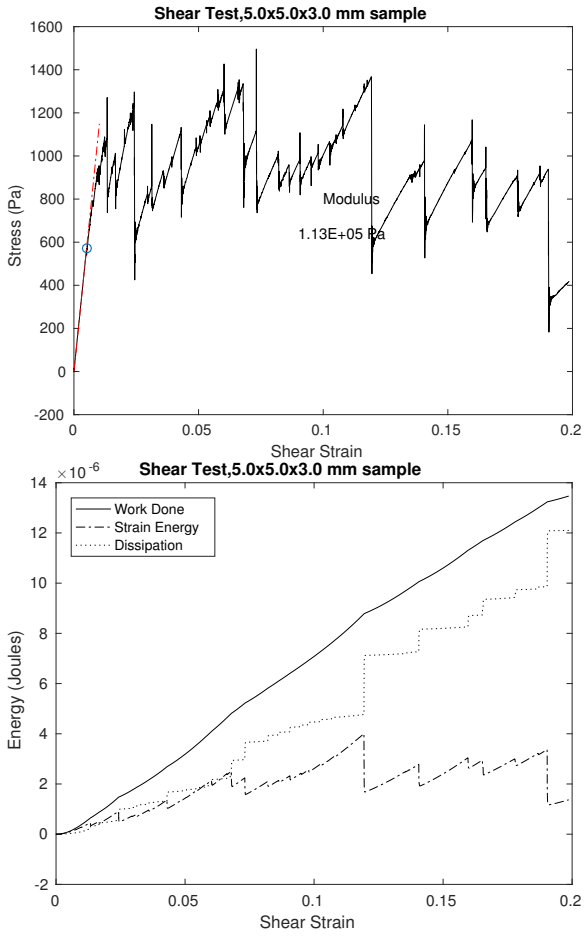

Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.007
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

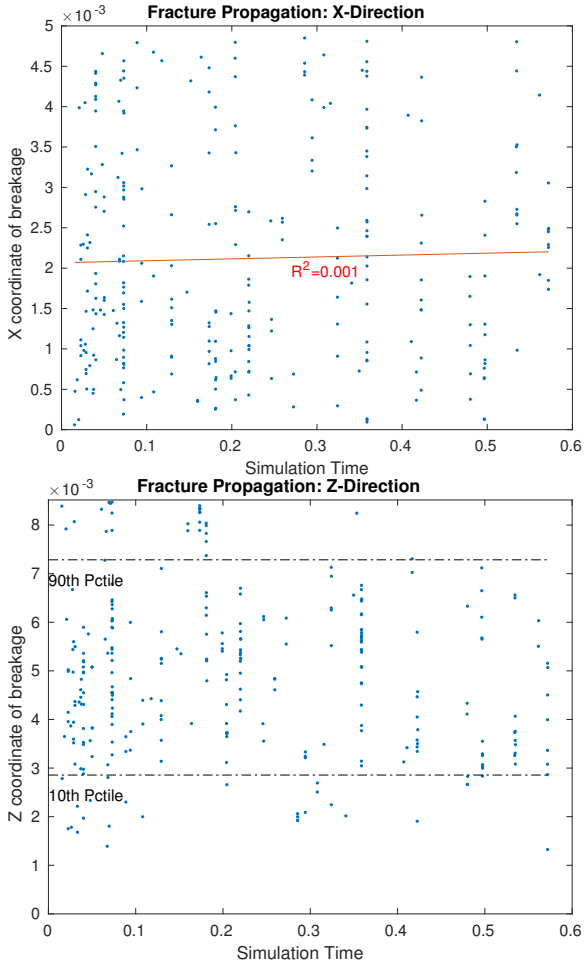
Numerical Subsample 5mm1:




Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2 =0.004
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
90th Pctile
10th Pctile
Simulation Time

Numerical Subsample 5mm2:




Fracture Propagation: X-Direction
×10^-3
X coordinate of breakage
Simulation Time
R^2=0.001
Fracture Propagation: Z-Direction
×10^-3
Z coordinate of breakage
Simulation Time
90th Pctile
10th Pctile