Achieving Damage Tolerance via Inhomogeneity and Nonlocality through Ultra-Lattice Materials

Wei-Hong Li

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

Master of Science in Mechanical Engineering

University of Washington

2019

Committee:

Marco Salviato

Junlan Wang

Lucas Meza

Program Authorized to Offer Degree:
College of Engineering

University of Washington

Abstract

Wei-Hong Li

Chair of the Supervisory Committee:
Dr. Marco Salviato
Aeronautics and Astronautics

Cellular materials find several industrial applications as engineered materials due to their outstanding specific mechanical properties, such as specific stiffness under bending and compressive strength. These features enable the attainment of lightweight structures with several potential benefits in e.g. aerospace, automotive and wind energy production.

However, cellular materials and structures are typically quasi-brittle and can suffer from severe failures. As a consequence, increasing the damage tolerance without penalizing the strength and stiffness significantly has been the subject of intensive studies in recent years. The formulation of a general approach to increase the damage tolerance of cellular materials lies at the heart of the present work.

To achieve this goal, two concepts are explored in this work: (1) non-locality realized by combining lattices featuring multiple length scales and (2) inhomogeneity which is achieved by combining strentch and bending dominated lattices. This new type of cellular material is called here *Ultra-Cellular Material* (UCM). It combines two or more representative periodic structures characterized by their topologies along with multiple characteristic length scales. UCMs are designed to improve the fracturing behavior compared to conventional cellular materials while maintaining a good balance between stiffness, strength, and toughness.

Towards this goal, a study on 2D regular lattices is conducted first to fully understand the mechanical behavior of different types of lattices and to set a performance benchmark for UCMs. Secondly, since the study focused on the case of a lattice made of quasi-brittle materials, the traction separation law was implemented to determine material properties

and to mitigate the sensitivity to element sizes.

In addition, finite element models are developed to perform the comprehensive analysis of its mechanical response from linear elastic behavior to ultimate failure. The comparison of damage tolerance between conventional and UCMs are estimated in terms of their force-displacement curve performance, toughness, size effects due to boundary layers, fracture energy and crack sensitivity.

To evaluate improvement in damage tolerance, the unloading tests are conducted to obtain the toughness of the lattice; lattice panels are test at different size to examine the free-edge effects on mechanical properties and the type II size effect were applied to evaluate the fracture energy. Since the provided size of lattice panel has not attained the region of linear elastic fracture mechanics (LEFM) yet, the Bažant size effect law is adjusted and implemented to approximate the fracture energy accordingly.

The analysis results show that the UCM gained an increase in specific toughness by 23 times with no measurable penalty in specific peak load and only 0.3% drop in specific stiffness compared to the conventional regular triangular lattice. Moreover, the UCM exhibited less sensitivity to the free edges effect and to the central crack compared to its component lattices (triangular and hexagonal lattice).

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I owe a debt of gratitude to my advisor, Marco Salviato for his invaluable help with my thesis program. He provided a challenge for me to take on a difficult task. His belief in me was a great encouragement for me to complete this project.

Secondly, I am deeply grateful to all members of the MAMS group, and especially to Sean E. Phenisee, for his mentorship and help in writing and presentation. His words and suggestion made me become an independent thinker, and his patience and kindness helped me to coordinate my project especially in solving problems I met.

Furthermore, a special thanks goes to my home-stay parent Ken Ross, who helped me with my English writing skills and gave me support to keep working on.

Last but not least, many thanks go to my parents Liu and Li for watching me from Taiwan and gave me any kind of support I needed. Their support motivated me to pursue this thesis degree.

NOMENCLATURE

a crack length

a_0 original crack length

A cross-section area

A_s cell-wall cross-section area

A_P projected area of RVE

$\hat{a}$ corrected coefficient of the relative density

$\hat{A}$ coefcient of the relative density

b out-of-plane thickness

B coefficient of Bažant type II size effect

b_s number of struts

$\hat{b}$, $\hat{B}$ exponent and coefficient of the lattice stiffness

c_f :addition crack length

C compliant stiffness

$\hat{c}$, $\hat{C}$ exponent and coefficient of the lattice strength

D size of specimen or width of entire lattice structure

D_0 transitional size of specimen

$\hat{d}$, $\hat{D}$ exponent and coefficient of the lattice fracture toughness

E, E' lattice Young's Modulus

E_s base material Young Modulus

E_m measured lattice Young's Modulus

E_p perfect lattice Young's Modulus

F load

g dimension less energy release rate

$\mathcal{G}$ energy release rate

G base material elastic shear modulus

G_f lattice fracture energy

G_{fs} base material fracture energy

h characteristic length of element

H coefficient of fracture energy depending on cell topology

I inertia of beam strut

j number of friction-less joint

K_I stress intensity

K_{IC} lattice fracture toughness

K_{13}, K_{13} cross-section shear stiffness

k the section dependent shear factor

$k(\alpha)$ dimensionless stress intensity

ℓ length of lattice strut

L size of unit cell

m number of mechanism

M moment acting on a lattice strut

M mass

n number of element

N number of cells

n_f number of broken cell-walls

p porosity of cellular material

P, P_s, P_{sum} global load, local load and sum of reaction force

$\bar{P}$ specific force

r distance ahead of the crack tip

s number of states of self-stress

u total extension or total displacement

u_e small displacement

u_p displacement at peak load

u_f displacement at fully failure

V_s solid wall volume

V^* topological volume of unit cell

W weight of lattice

Z node connectivity

α fraction of crack length to specimen size

δ total displacement

δ_s deflection of a beam

ϵ engineering strain or nominal strain

ϵ_t true strain

ν Poisson's ratio

ρ_s base material density

ρ^* equivalent density of lattice

$\bar{\rho}$ lattice relative density

σ global stress

σ_f lattice fracture stress

σ_f base material fracture stress

σ_s local stress

σ_u ultimate stress

σ_{u_s} ultimate stress of cell-wall material

σ_N nominal stress of lattice specimen

σ_Y lattice yield stress

σ_{Y_s} base material yield stress

Chapter 1

INTRODUCTION

The interest in cellular materials is growing continuously thanks to their weight efficiency [23], outstanding stiffness to weight ratio under bending [27], fracture strength to density ratio when loaded shear [27] and good energy absorption [23, 27]. The growing number of publication shown in Figure 1.1 indicates that more and more research engages in study relating to cellular solids. These foam solid exhibit low density and unique mechanical

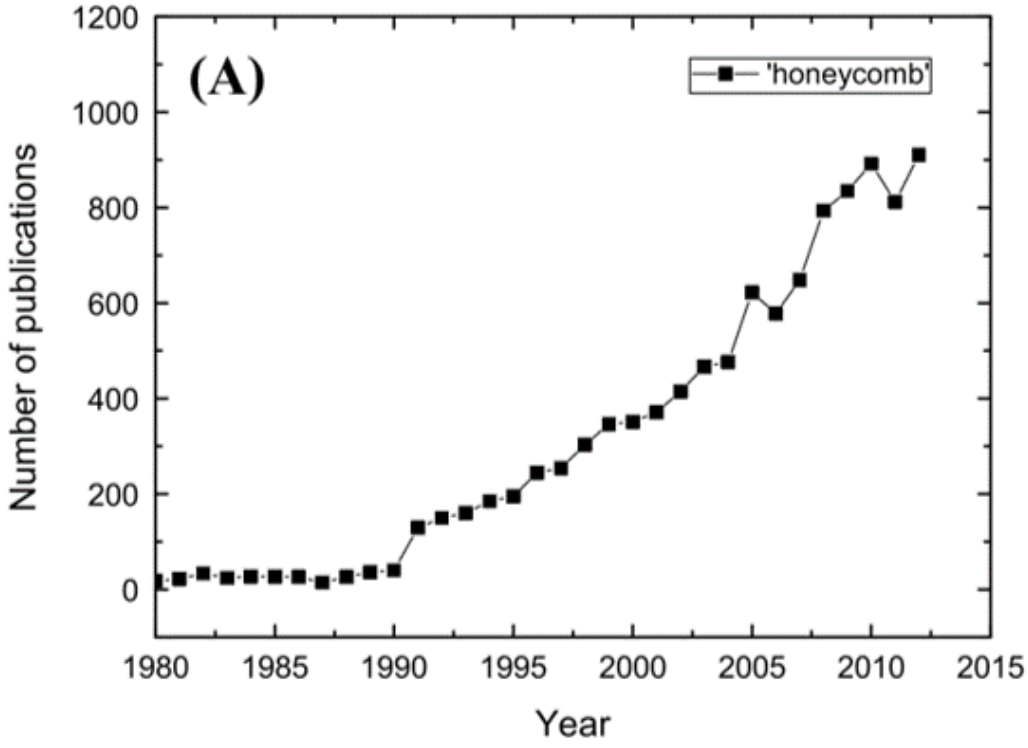


Figure 1.1: Number of annual publications on 'honeycomb' collected from Science Citation Index Expanded. Image courtesy of [1].

properties. This can be seen in Figure 1.2 which shows that the bulk materials such as metal

(near the top right) have high stiffness/strength and high density while cellular materials (near the bottom left) have low stiffness/strength but lower density. The foam material also has better specific stiffness/strength compared to non-technical ceramics. Additionally, the application of lower density material yields extended dimensions and greater surface area. This gives an opportunity for creativity in engineering which the exclusive use of bulk material cannot offer. As such, these lattice materials have been finding many practical applications in a great variety of engineering field [11, 23, 28, 29].

It is common to see the honeycomb composite cored sandwich skin covers used in the aircraft industry for wings and fuselages [30, 31] where weight efficiency is a concern. On the other hand, owing to the large surface area of honeycombs, the use of metallic and polymeric foam catalytic converters is increasing in the automotive field [23, 32] where carriers of catalysts are critical. Another example of the application of lattice materials is in the field of tissue engineering where these materials are used to construct scaffolds for tissue growth. There advantage here is their lightness and ability to provide space for cell growth [33–35]. It is important to note that cellular solids are ubiquitous in nature, such as in cork [36], balsa-wood cores [37], bee's nests [38], trabecular bone [39,40], sea sponges [41], cuttlefish shells [23] etc. Most of these natural cellular materials have been used by people throughout history [23]. Wood, for instance, is still widely used as structural material [23].

The foregoing applications of cellular materials have fostered, over time, a significant research effort devoted to enhancing the mechanical behavior of these materials. Schaedler et al. demonstrated possibly the lightest but also the strongest metallic lattice ever produced [42] (see in Figure 1.3(a)); Meza and colleagues demonstrated the creation of ultralight hollow ceramic structural metamaterials that can recover after significant compression (Figure 1.3(b)). This implied that it is possible to transform a strong and dense brittle ceramic into a ultralight, energy-absorbing, and recoverable metamaterial. Meanwhile their contribution extended the range of properties (strength and stiffness) available for engineering lattices [4]; Similarly, Bauer and co-workers fabricated and evaluated the hierarchical nano-lattices of glassy carbon (Figure 1.3(c)) which provide strength close to the theoretical value and an impressive strength to density ratio higher than that of conventional microlattices [5]; and Plesha et al. [6] manufactured a 3D lattice with negative Poisson's ratio that

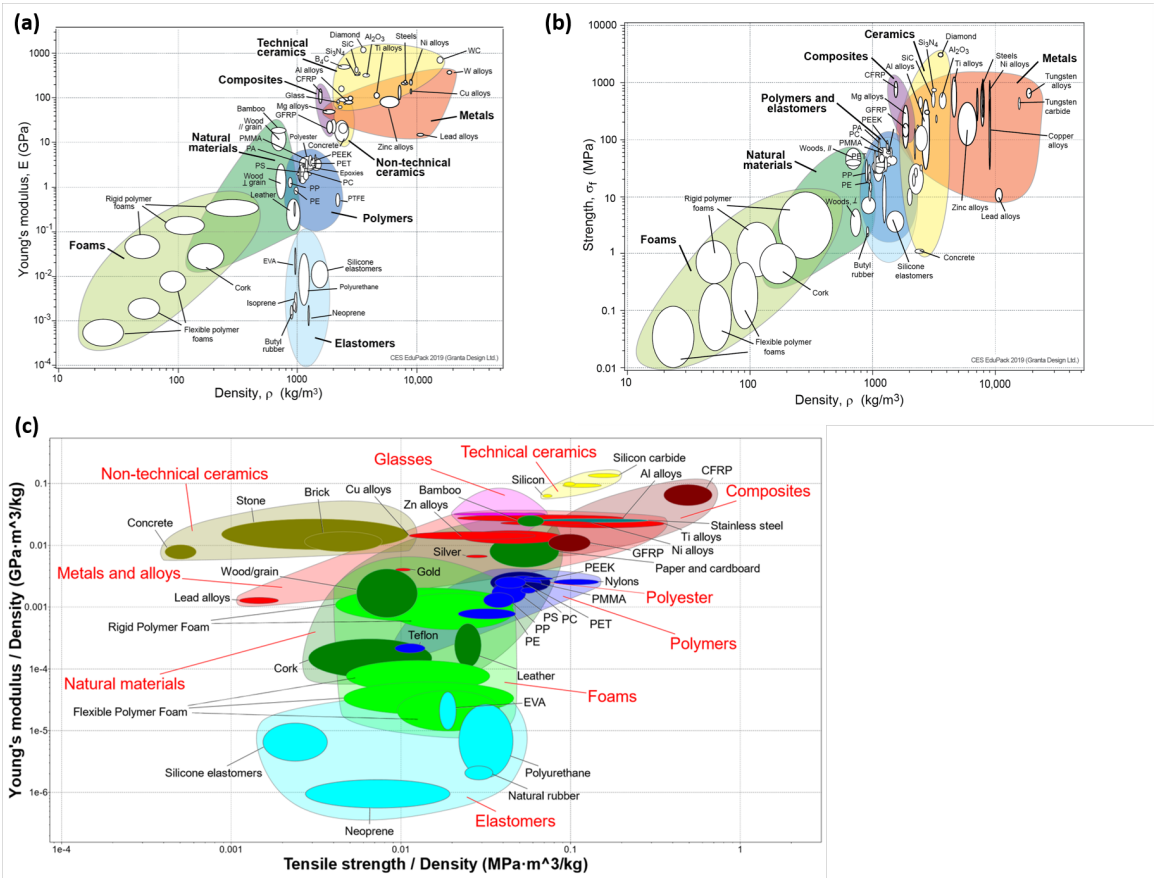


Figure 1.2: Bubble charts showing distribution of mechanical properties range from bulk materials to natural cellular materials: (a) The stiffness-density chart; (b) The strength-density chart; (c) The specific stiffness-specific strength chart. PE, polyethylene; CFRP, carbon fibre reinforced polymer; GFRP, glass fibre reinforced polymer; PEEK, poly ether ether ketone; PA, polyamide; PC, polycarbonate; PET, polyethylene terephthalate. Chart (a) and (b) are taken from [2] while chart (c) is created using CES EduPack 2017, Granta Design Ltd

can increase energy absorption (Figure 1.3(d)). Recently, Pham and his colleagues [7] mimicked materials' grains via lattice and showed damage resistance to interference in lattice structures as shown in Figure 1.4.

In spite of the impressive characteristic of energy absorption in these cellular materials, there has remained a serious concern for the toughness of cellular solids. In addition, the lattices of some types of materials exhibit brittleness in certain applications such as the

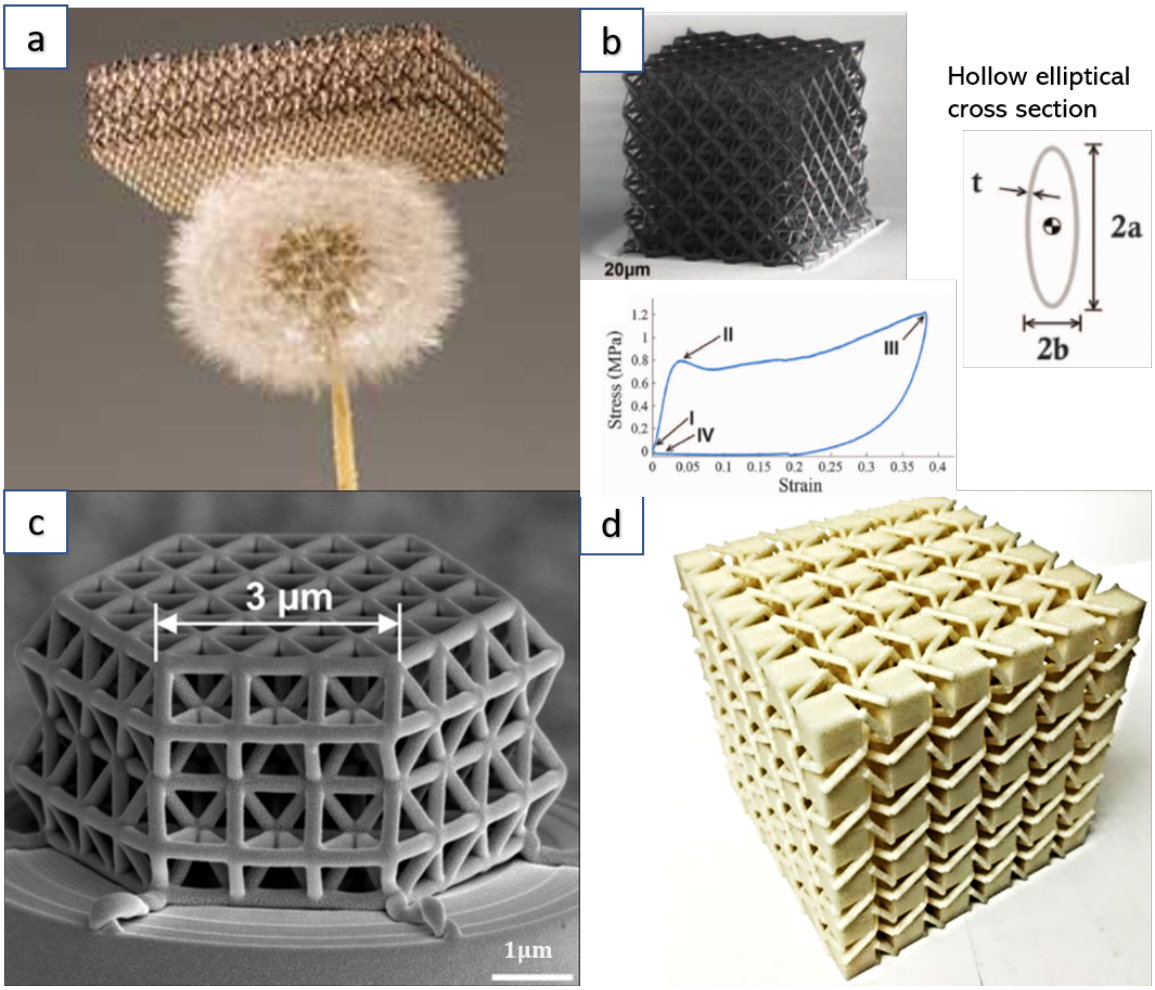


Figure 1.3: (a) Researchers at HRL laboratory created a metallic microlattice structure of interconnected hollow tubes, which can stand on a dandelion. Image courtesy of HRL Laboratories [3]; (b) A SEM image of alumina octet-truss nanolattice which provides a ductile-like performance under compression test shown in stress-strain curve. Reproduced from [4]; (c) Glassy carbon nanolattices isotropically shrink to about 20% of their initial size during pyrolysis. Image courtesy of [5]; (d) A 3d-printed 555 lattice with an aspect ratio of 1.642. The lattice is about 92 mm on a side. Image courtesy of [6].

honeycomb formed ceramic substrates in car engines [43]. It can be subjected to the thermal shock caused by heat generated from combustion [43, 44]. Since this ceramic material has shown brittleness, especially under tension, a small crack on the surface can propagate, thus leading to catastrophic failure [45] (see in Figure 1.5). Additionally, the honeycomb

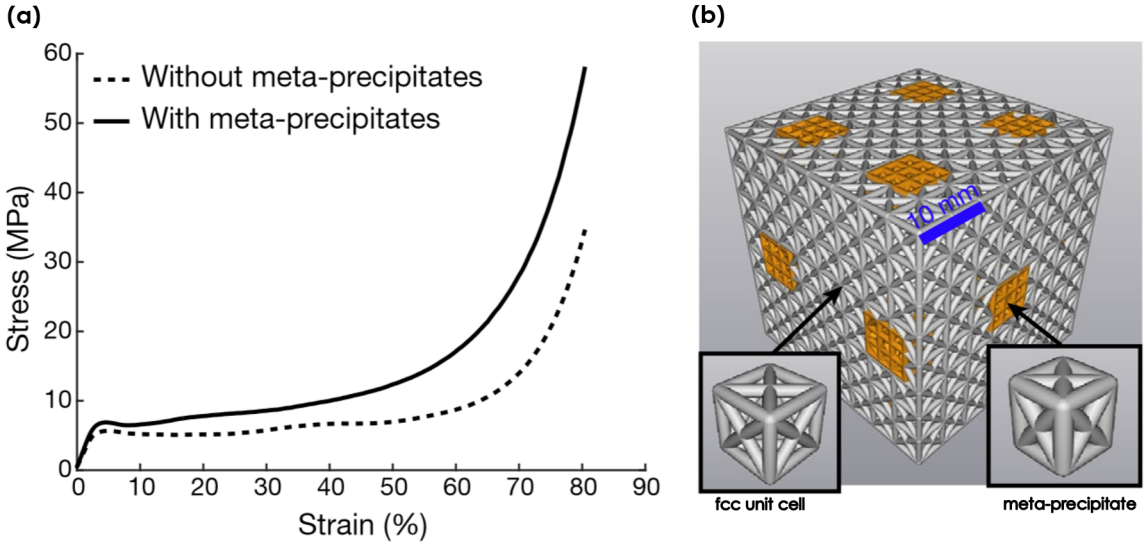


Figure 1.4: (a) Comparison between constitutive stressstrain responses of architected materials without and with meta-precipitates, (b) A configuration of meta-precipitate lattice (orange) embedded in the matrix built from face-centred cubic (fcc) lattice. Image courtesy of [7].

composite panels on an aircraft can suffer severe loading while flying at a high altitude [9]. The temperature at this height can be lower than -40^oC, which causes the standing water (see in Figure 1.6) to freeze [9]. This can stress and expand the honeycomb cell wall, leading to fracture of the structure induced by an onset of damage [9, 46].

Another widely known example of catastrophic failure caused by the fracture of foams is the Space Shuttle Columbia catastrophe in 2003 [47]. A one-pound piece of broken foam came out of the fuel tank insulation and hit the left wing of the shuttle as shown in Figure 1.7. This led to a crash of the whole aircraft as the shuttle re-entered the Earth's atmosphere.

Therefore, in order to prevent catastrophic failures, it is necessary and worthwhile to study the fracture mechanics of cellular solids and to find ways to evaluate their damage tolerance.

Maiti et al. [48] and Gibson et al. [28] derived approximate formulas via micro-structural models to evaluate the fracture toughness of hexagonal honeycomb based on three assump-

Figure 1.5: An example of cracked catalytic converter and broken catalytic converter. Image courtesy of Davis Recycling Inc [8].

tions: (1) continuum approximation for the lattices, (2) constant rupture modulus, and (3) negligible axial strength of strut ahead of the crack tip. Huang and Gibson, [43] modified the aforementioned example by applying Weibull statistics to the modulus of rupture to each strut under axial loading. They also discussed the size effect of the struts and the influence of short cracks. Their derivation regarding independence of cell size to normalized fracture toughness had been confirmed with the experimental data provided by Brezny and Green [29]. Ashby et al. [49] extended the expression for the fracture toughness to multi-axial loads. Huang and Chiang [50] proposed another modification to predict the fracture toughness of brittle honeycombs by applying the formula for bulk material under the three-point bending test proposed by Srawley [51]. Various methods were proposed. Chen et al. [52] introduced another formula derived from strain gradient with the K-field for triangular, hexagonal and square lattices without supported experimental data; Fleck and Qiu [53] and Fleck and Alonso [35] evaluated numerically and analytically the fracture toughness and sensitivity to a flaw for triangular, Kagome, hexagonal and diamond celled honeycomb of elastic-brittle behavior according to linear elastic fracture mechanics (LEFM) with concern of the T-stress effect; Lipperman et al. [54] evaluated the fracture toughness of 2D honeycombs via discrete Fourier transformation and the results matched the prediction obtained by Fleck and Qiu [53]. Since the growing number of studies in fracture response

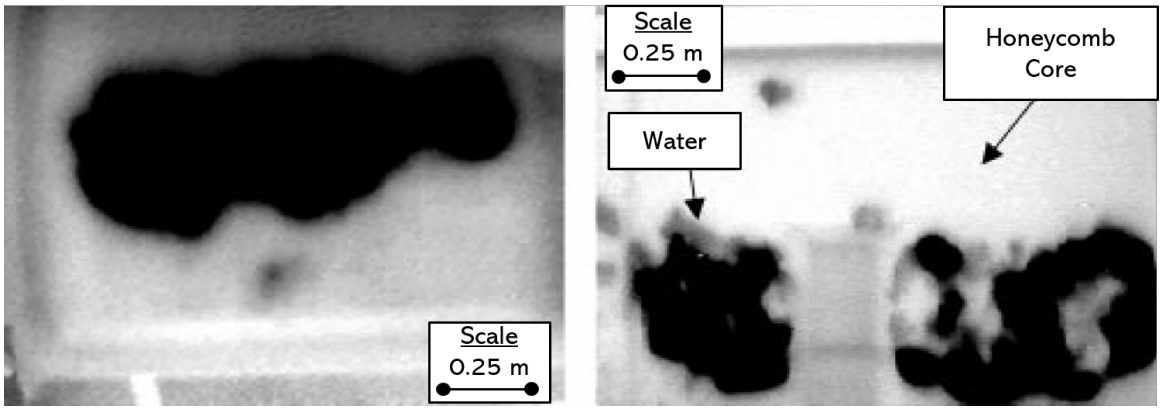


Figure 1.6: Infrared images of two 767 upper fixed wing panels showing the location of water in honeycomb sandwich structures. (Bottom view.) (Dark areas represent liquid water.) Image courtesy of [9].

of lattices, Alonso and Fleck [26] wrote a review regarding the fracture of brittle lattices, which contained most of the previous studies, and gave a organized overview elaborating the progress of development. A year later, Fleck and Ashby [11] advanced a relation between the fracture toughness of lattices and fracture toughness of cell-wall materials in order to provide developmental history of cellular materials and future expectation. Also, some experimental data was demonstrated to verify the analytical predictions of fracture toughness for lattices [55,56]. More researchers have extended this field of study including foam structures [57], imperfection sensitivity [24], Voronoi-constructed honeycombs [58], hierarchical self-similar honeycombs [59], etc.

The foregoing studies on toughness and fracture toughness of lattice materials are limited to the elastic-brittle response. While assuming a perfectly brittle behavior enables the formulation of relatively simple evaluations, this is a significant limitation since most of the materials used in cellular materials are indeed quasi-brittle. Quasi-brittle materials are characterized by a complex, heterogeneous micro-structure that typically lead to a fracture process zone (FPZ) that cannot be neglected. This FPZ typically is then followed by a gradual degradation of the structure performance and strain softening behavior, with stress decreasing gradually and increasing strain after the peak load [60]. This is in contrast

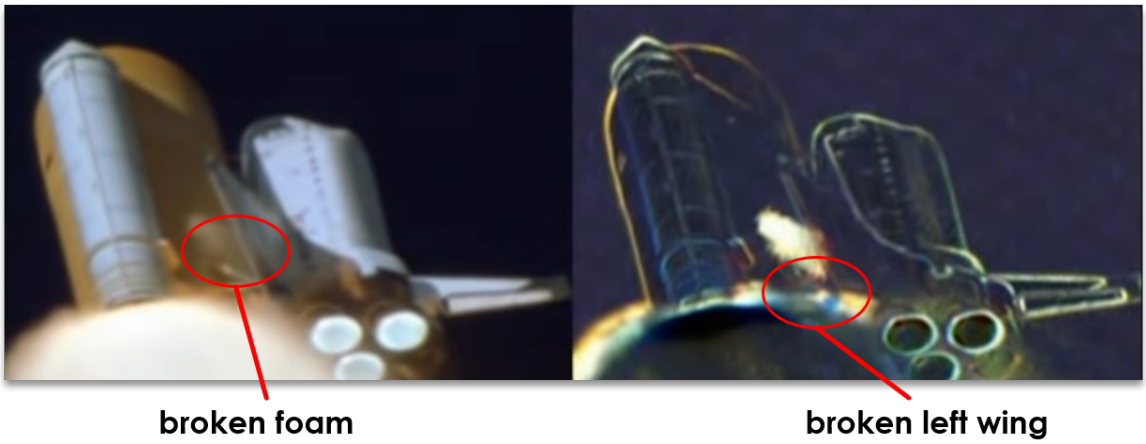


Figure 1.7: The Space Shuttle Columbia catastrophe in 2003 killed 7 crew members due to a broken foam causing damage to the left wing of space shuttle. Image courtesy of BBC [10].

to brittle material which features unstable load drop after peak. Fleck et al. [61] first introduced the traction-separation relation in linear softening law for ductile metal foam. Onck and Mangipudi [62] presented the fracture toughness for Voronoi foams of quasi-brittle material implementing constitutive law with traction separation law during linear softening. Maimi et al. [25] incorporated linear softening behavior to express quasi-brittle crack model via R-curve theorem for periodic lattices. They identified the critical strut length which determines stress dominant or energy dominant crack propagation via the Bažant size effect law.

All of the work that has been done in the aforementioned studies aims to avoid the chance of serious failure in applications made of cellular materials. These studies, however, focused more on characterizing the brittle fracturing behavior of lattices, providing few solutions for the need to toughen the lattice structure.

To increase the toughness and prevent a collapse in the honeycomb-formed type of application, two concepts, (1) multiple length scale and (2) in-homogeneity can be employed in conjunction to obtain the desired damage tolerance and to increase the fracture energy by applying force redistribution. However, the concept of multiple length scale is different from the hierarchical structures mentioned in [63]. These multi-scale lattice materials have

every single strut made up of lattice of smaller scale as shown in Figure 1.8. The effect is that tht buckling strength of the lattice of larger length scale is increased. The finer the length scale the higher the buckling strength [11].

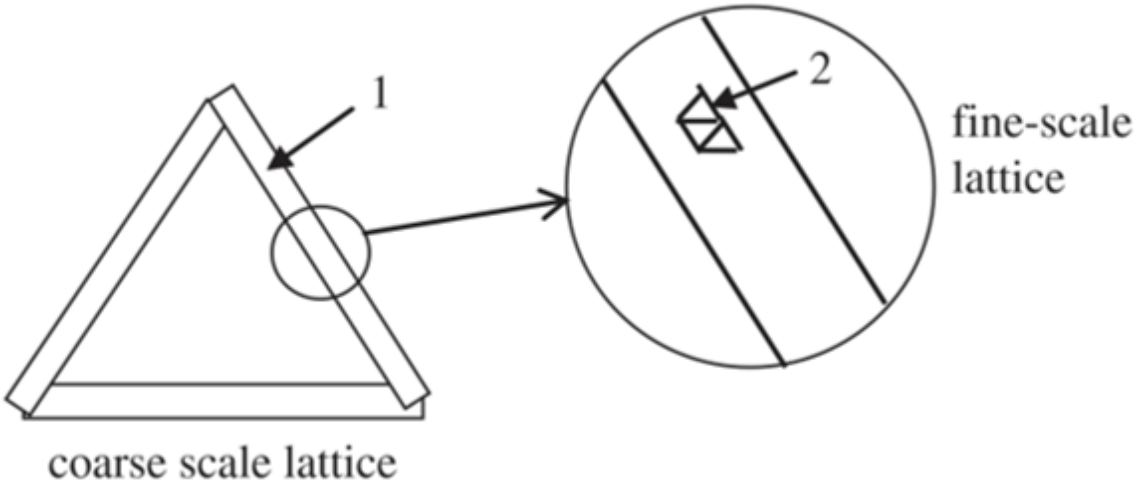


Figure 1.8: Schematic of a hierarchical lattice on two length scales. Image courtesy of [11].

On the other hand, in order to fully redistribute force in one lattice, another lattice of higher length scale is add on the lattice of lower length scale, which is considered as an idea of multiple length scale here. With this perspective, the present study introduces a new benchmark of cellular materials, Ultra-Cellular Materials (UCMs) to study fracture behavior. The objective of this study is to improve the damage tolerance for the lattices via their topology only, and not their materials. With the knowledge and efforts contributed by previous studies, the potential of cellular solids should be extended to achieve an improvement in damage tolerance, especially in the case of quasi-brittle lattices. Doing so may not only open a new door for the design of better structural materials, but it may also fill the gap between the study of quasi-brittle lattice and the conventional fracture mechanics study of cellular materials.

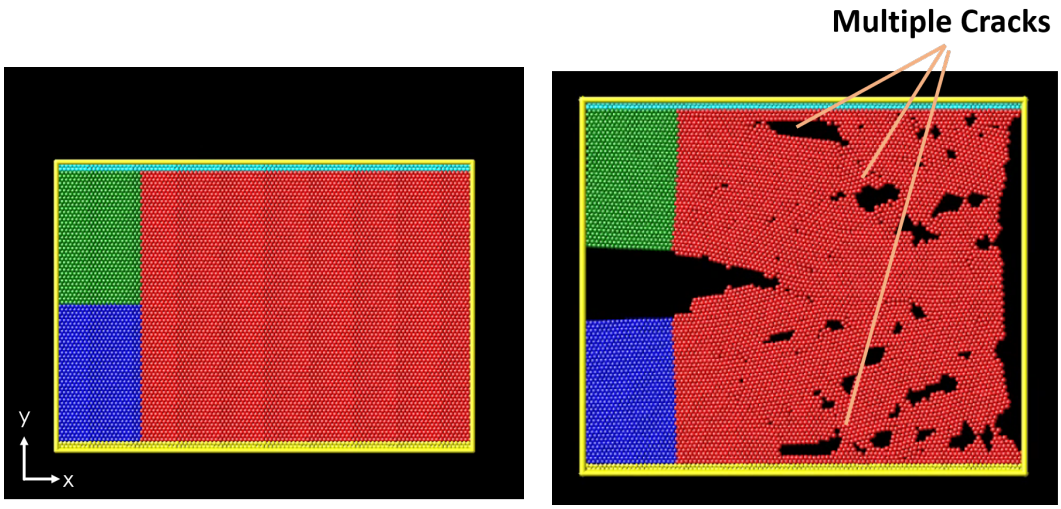
1.1 Motivation of UCM

1.1.1 Inspiration of Non-locality

Non-locality, the first conceptual element of ultra-cellular materials, came from the simulation using the molecular dynamics (MD) method to study fracture behavior. MD is used to simulate the evolution of the position of atoms in time by the relevant interatomic potentials that can be identified through an empirical force field [64]. Such potential is different from contact force which cannot transmit force once a split in an object has been caused by fracture. As seen in Figure 1.9, at the end of the test, multiple cracks are revealed in the plate of atoms. This suggests that, although the interaction between two adjacent atoms is broken, the interaction between the second closest set of atoms can continue to carry the load. When this second order interaction is broken, the connections between the third closest atoms will keep redistributing the force, as do the atoms that are located continually farther away as illustrated in Figure 1.10. This phenomenon introduces the concept of non-locality in fracture, which refers to the distribution of damage throughout the plate.

A similar mechanism can be seen in ripstop fabrics. Unlike traditional plain weave fabrics, ripstop fabrics are woven in fabrics of different length scale as shown in Figure 1.11. The thicker fiber has extra strength, which can help stop a tear in the first square of thinner fibers and thus makes the material stronger then the plain weave fabrics as shown in Figure 1.12 [65].

The method proposed here to achieve non-locality is to combine two different cellular materials with different length scales. When the localization occurs on the smaller scale, the larger scale should help redistribute the local force so that the crack is slowed and scattered over the structure. It can be seen that the larger scale bridges one unit cell of smaller scale to another remote unit cell, which acts as remote interaction holding cells together. This remote interaction holds cells together. Therefore, instead of one single crack propagating destructively through the first lattice, incorporating the second lattice of larger length scale should lead to multiple, less harmful cracks around the crack tip (Figure 1.13).



(a) Before crack. (b) Multiple cracks.

Figure 1.9: Molecular dynamics simulation of crack distribution. This MD simulation was conducted in LAMMPS to simulate tensile pull on a 2-D solid using the Lennard-Jones (LJ) potential, a type of pair-wise potentials models provided by LAMMPS. The whole model used shrink-wrap boundary conditions on x- and y- direction, and periodic boundary condition on z- direction [12].

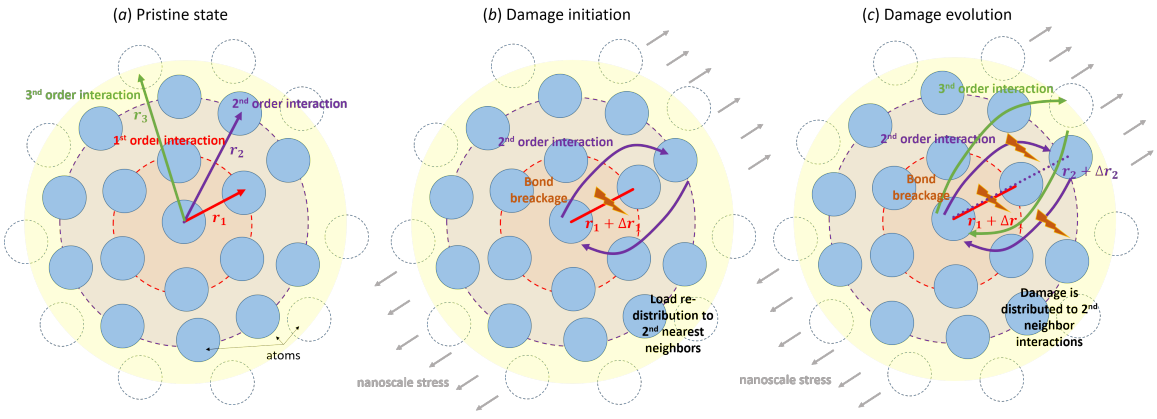


Figure 1.10: Schematic of interactions among a bulk of atoms showing that the remote interactions continue to act on a broken pair of atoms following the onset of damage under nano-scale tensile stress.

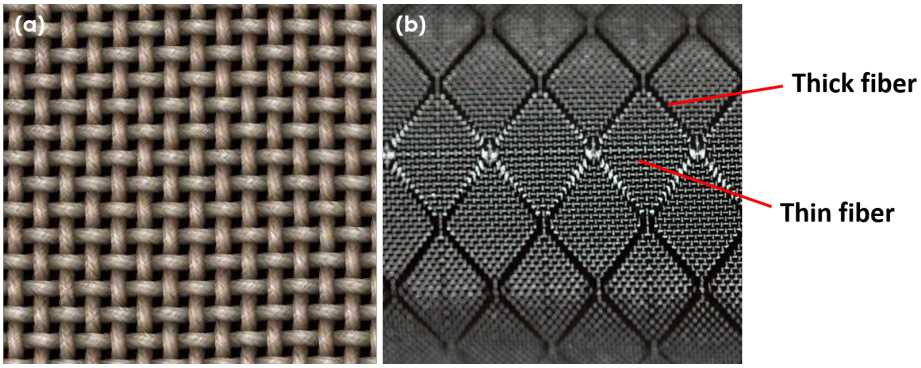


Figure 1.11: Example of (a) plain weave fabrics [13] and (b) Nylon diamond ripstop fabrics 420D [14].

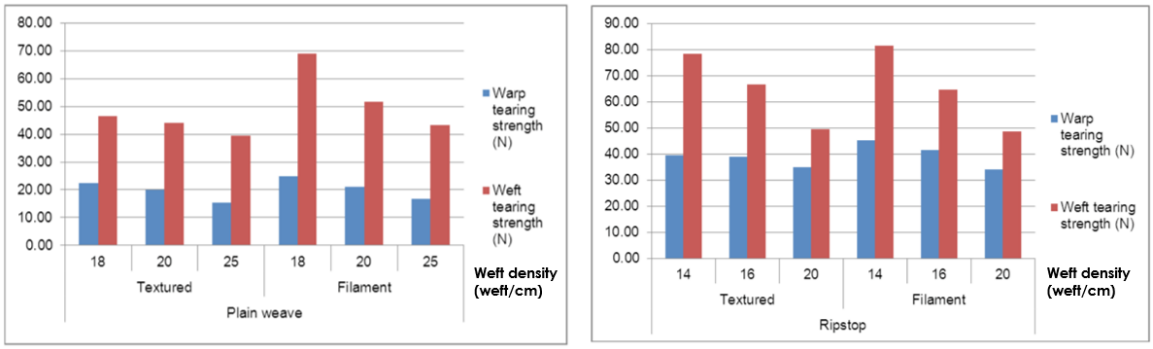
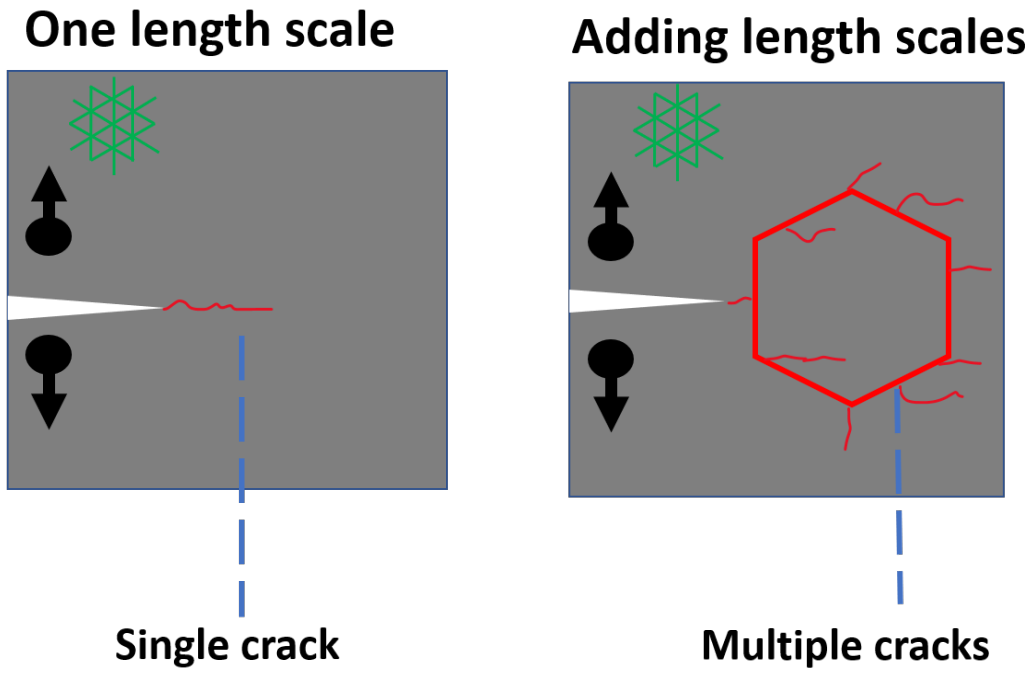


Figure 1.12: Comparison of tear strength between plain weave and ripstop fabrics. Charts are taken from [15].

1.1.2 Inhomogeneity inspired from bio-materials

Inhomogeneity, the second idea behind the concept of UCMs, is found in the bio-materials such as nacres and marine mussels. They take advantage of in-homogeneity by combining soft and strong materials to enhance their damage resistance. The byssus threads of mussels are composed of tissue which is Distal (stiff, though brittle) and Proximal (soft, but ductile) that can efficiently dissipate the impact of sea waves [16] (Figure 1.14). This ability is the result of a mechanism operating between two types of materials. The ductile material provides viscosity and damp out the external impact (kinetic energy), while the stiff material transfers the kinetic energy into potential energy. On the other hand, nacre



(a) Single crack propagates through the lattice of one length scale.

(b) Multiple cracks occur over lattices of multiple length scales.

Figure 1.13: Schematic of the idea of patching larger scale of lattice to distribute damage and to prevent single localized band.

consists of thicker, brittle, layers (500 nm) of aragonite crystals which are separated by the thinner, but ductile, layers (30 nm) of highly cross-linked protein [66] (Figure 1.15(a-c)). These alternating layers are constructed as shown in Figure 1.15(c-e). This structure allows the proteinaceous layers to absorb energy by deforming elastically and distribute the bulk of the energy as crack formation in many other aragonite crystals [17]. This mechanism prevents the crack propagation in generally homogeneous material and toughens a nacre by $1,000$ times that of its main constituent, chalk [67–69] through creation of many small microcracks [17].

In the case of ultra-cellular materials, instead of using different cell-edged materials, different deformation types of lattices are combined. This means that the advantage of

heterogeneity in UCMs is achieved by discriminating among the capabilities of different lattices including softness and stiffness.

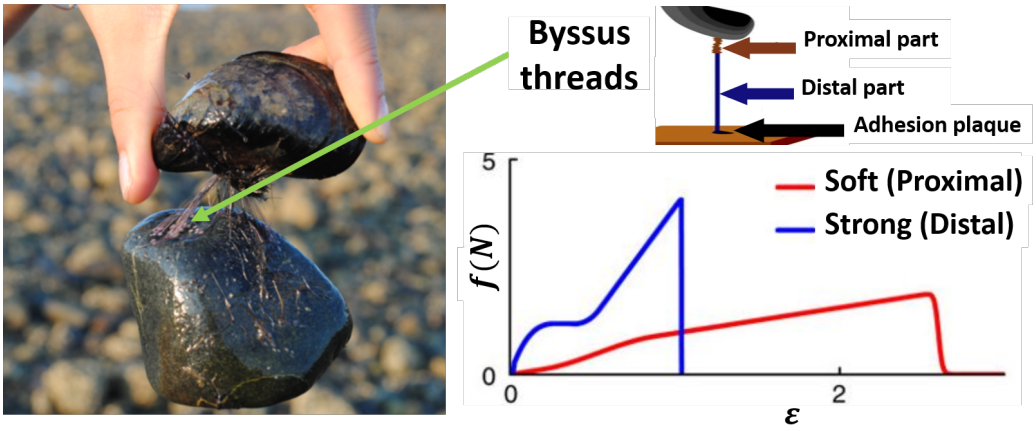


Figure 1.14: Mussel with byssus threads composed of Proximal and Distal tissues. Reproduced with permission from [16].

In conclusion, the development of UMCs has been motivated by the need for damage tolerance improvement which is achieved via mainly two inspired concepts, non-locality and inhomogeneity.

The objective of this study and its methods are summarized in Figure 1.16. First, the non-locality is attained by employing multiple length scale to increase damage tolerance. At the same time, the use of multiple length scale can bridge different lattices when applying the second idea of inhomogeneity which is expected to enhance toughness of cellular material. The quasi-brittle materials are implemented to comprise lattice leading to a concern of size effect. These toughened architectured materials is referred as UCMs.

1.2 Research Process

The present thesis started from lattice selection by determining softness and stiffness in cellular structures using the connectivity criterion (the higher connectivity it has, the stiffer it is) to decide the certain portfolio of UCMs.

Central to the process of lattice selection for the present thesis is the consideration of

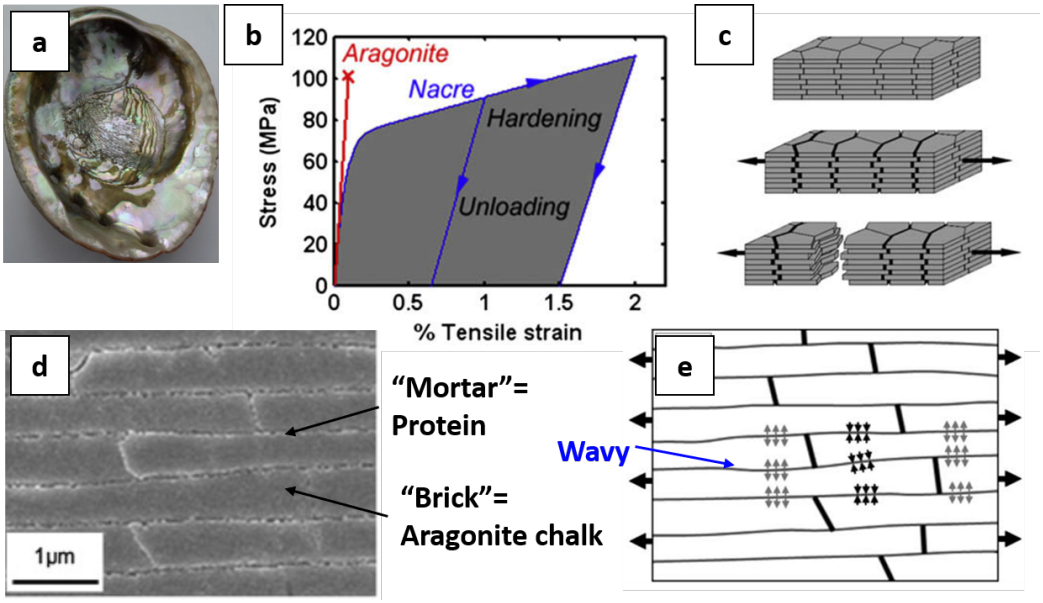


Figure 1.15: (a) Nacre (mother-of-pearl) on the inside of an abalone shell; (b) Comparison of tensile stressstrain response between nacre and its constituent showing an improvement in toughness for nacre; (c) Corresponding deformation mode under the tensile test; (d) Scanning electron micrograph of staggered structure in nacre; (e) Configuration of the nacre tablets showing stresses involved under stretching along the tablets. Reproduced with permission from [17, 18].

softness and stiffness in cellular structures. These characteristics have been determined using the connectivity criterion (the higher connectivity it has, the stiffer it is) in order to select a particular portfolio of UCMs.

Only two-dimensionl cases have been considered for preliminary study. Their equivalent capabilities have been evaluated through reviewing in-plane properties including relative density, Young's modulus, fracture strength and fracture toughness. This provides analytical prediction to verify the results of simulation models of honeycombs used in the study. Both the periodic boundary condition (PBC) and multi-cell lattice model were applied separately to estimate the geometric model built for the sake of understanding elastic behavior for each type of honeycomb.

Following this estimation, two specific honeycombs: (1) Triangular lattice and (2) Hexagonal lattice were chosen in order to represent stretching-dominated and bending-dominated

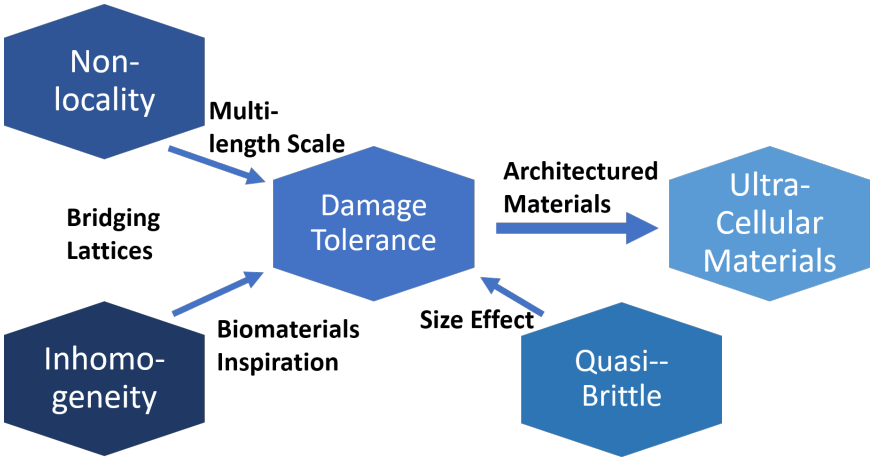


Figure 1.16: Road map for objective.

lattice, respectively, for the engineering UCM model. Secondly, the cohesive model was used for the cell-wall materials that respond with quasi-brittle behavior to research the quasi-brittle lattice. Thirdly, the employment of multiple length scales in the engineering of the UCM model was introduced and the configuration of UCMs was presented as a model in later simulations.

The Python script was developed for the generation of UCM geometries and for pre-processing. The constraint equations for UCMs were written into the input file in an outside-Abaqus environment with Python script. These constraint equations are built to combine triangular and hexagonal lattices.

The quasi-static tensile test was considered under displacement control with a smooth amplitude, and all simulations were submitted into the Abaqus 14.6 (2018) explicit solver environment in the Hyak supercomputer system at the University of Washington. The specific toughness and the specific peak load comparisons between pure lattice and UCMs are examined and evaluated with a specific load displacement curve to confirm the early presumption of toughness performance.

Last, but not least, the Bažant size effect law is adjusted by relative density and is applied to calculate fracture energy. A compromise design is accordingly presented to summarize the outcomes of UCMs with a statement of achievement and suggestion for future works.

The research process and its related tools are summarized in Figure 1.17.

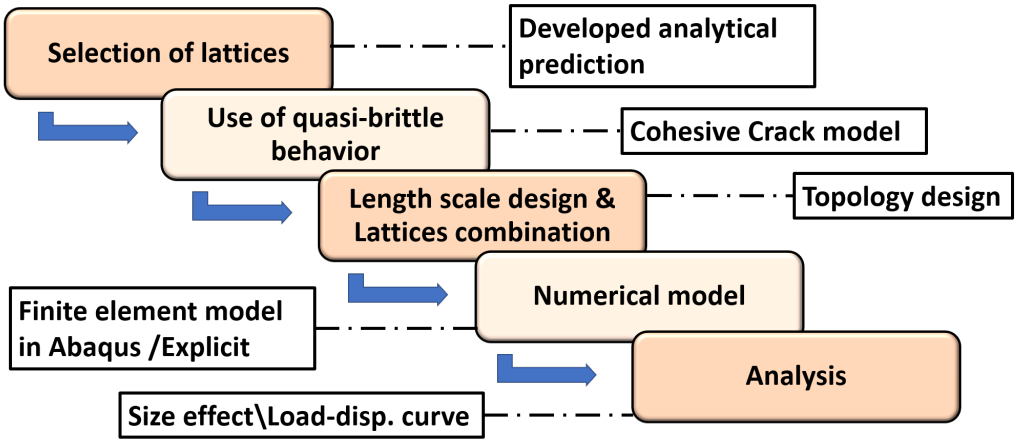


Figure 1.17: Research process and related tools.

Chapter 2
SELECTION OF LATTICES

Cellular materials can be concisely characterized as an architecture constructed by a system of struts and joints that can be made from variant materials such as metal, polyesters, glass and ceramics. Cellular materials in two-dimensions are noted as honeycombs and are often referred to as foams when in three-dimensions [23]. To choose the desired cellular materials, several ways to classify lattices are provided. Among these, the categorization using a deformation type of cell-wall is the primary concern.

2.1 Classification of Cellular Materials

There are many ways to categorize the lattices and foams. They can be described according to their cell arrangements: (1) periodic lattice and (2) stochastic lattice. Periodic lattices are organized by repeating unit-cells whose shapes are regular or irregular to fill a plane (Figure 2.1). In contrast, stochastic lattices are constructed by randomly repeating unit-cells with unusual shapes and sizes (Figure 2.2). The periodic lattice can be further classified into (a) regular lattices, (b) semi-regular and (c) irregular lattices [11, 70–73]. Regular lattices are filled with one regular polygon, and there are only three types of regular lattices: triangle, square and hexagon for planar lattice [11, 70–73]. As described by Syôzi in 1972, Kagome [74] for example, belongs to the semi-regular which is filled with more than two types of regular polygons [75]. Only 8 types of semi-regular planar lattice exist [11, 70–73].

Considering the materials in three-dimensions, the foam can be classified into (1) open cell and (2) closed cell. The open cell has edges only and its room can connect to the room of adjoining cells (Figure 2.3(a)), while the closed cell has solid faces which separate the rooms of adjacent cells (Figure 2.3(b)). Because this is a preliminary study, only the periodic planar lattices are considered for UCMs.

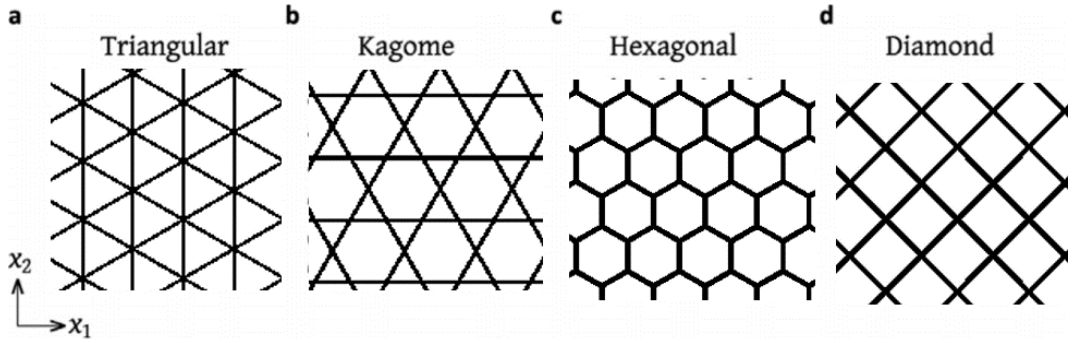


Figure 2.1: Typical examples of periodic lattices.

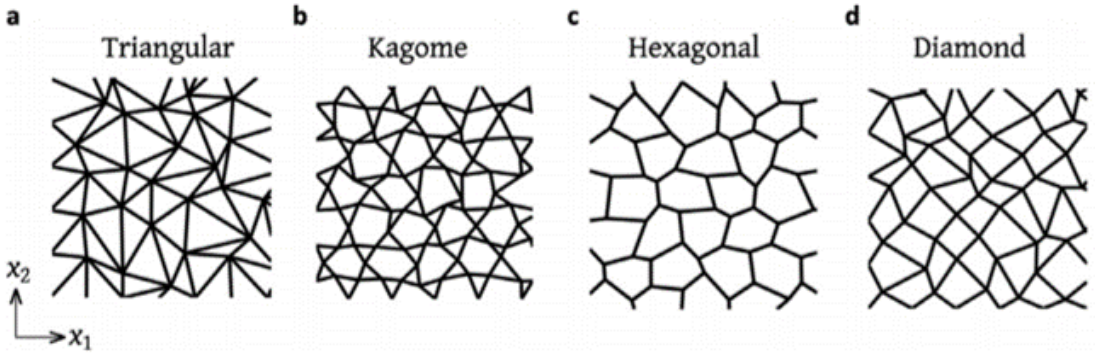
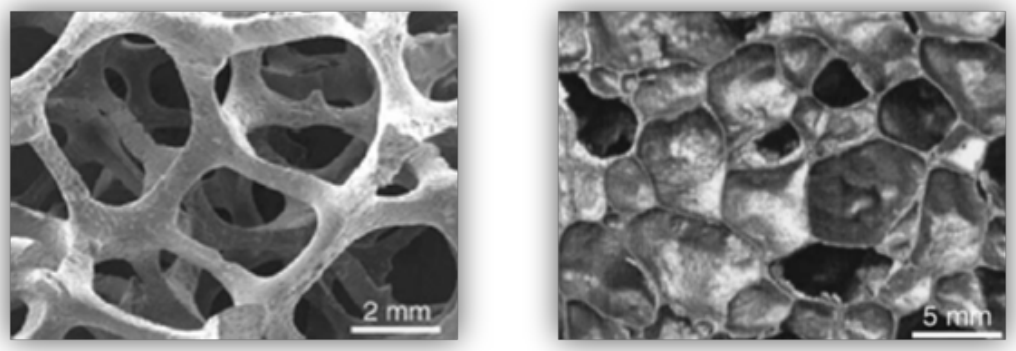


Figure 2.2: Typical examples of stochastic lattices. Image courtesy of [19].



(a) Open cell foam.

(b) Closed cell foam.

Figure 2.3: Examples of (a) open cell foam and (b) closed cell foam. Images are taken from [20].

2.1.1 Stretching and Bending Dominant

In addition to their topology, the lattices can also be characterized by the force-bearing condition of their struts under loading. Such conditions of the solid walls of lattices can be sorted as either stretching or bending deformation. Hence, the lattice generally can be characterized as exhibiting either (1) stretching dominated or (2) bending dominated mechanical behavior [11]. In the stretching dominant case, each strut of the lattice carries the loads, including tension and compression primarily in the axial direction of the bar. This generally leads to outstanding stiffness and strength but also to a brittle response [32]. Typical examples of this class of lattice are the triangular lattice and Kagome lattice [76]. On the other hand, the struts of bending dominant lattice can also carry the bending moment and transmit the shear force through the cross-section of a beam. This means that the bending dominant lattices display magnificent ductile behavior and energy absorption but provide relatively lower strength and stiffness [32]. Here, examples are the diamond and hexagonal lattices. To mathematically identify stretching and bending dominated lattices, the criterion of cell topology formulated by Maxwell (1864) [77] can be used. This model gives the following relation between the number of struts, b_s, of a pin-jointed frame and the number of friction-less joints, j, to attain a rigid cell:

$$\begin{cases} b_s = 2j - 3, & \text{in two-dimensions;} \\ b_s = 3j - 6, & \text{in three-dimensions.} \end{cases} \tag{2.1}$$

Otherwise, the cell can be considered as a mechanism. Unfortunately, not all lattice microstructures satisfy Maxwell's criterion, which means it is only a necessary condition for rigidity. Several years later, Calladine (1978) [78] derived the following generalized Maxwell's criterion:

$$\begin{cases} b_s - 2j + 3 = s - m, & \text{in 2 dimensions;} \\ b_s - 3j + 6 = s - m, & \text{in 3 dimensions.} \end{cases} \tag{2.2}$$

,where s and m are the numbers of states of self-stress and of mechanisms respectively. Each can be determined by calculating the rank of the equilibrium matrix that describes the frame [79]. Self-stress is a state of prestress when a strut involves compression or tension under zero

external force [78]. On the other hand, mechanism or infinitesimal mechanism represents a mode of motion the frame can experience. Therefore, the degree of freedom (dof) of a certain framework can be determined by $dof = s - m$, which implies that a rigid cell has zero degrees of freedom. However, this is not a sufficient but instead a necessary condition since it doesn't insure that $s = m = 0$ under the vanishing of the LHS in Eq. 2.2. As a consequence, Deshpande et al.(2001) [80], recently, have investigated a topological criterion for deciding the type of architecture and have proved that the connectivity $Z = 6$ and $Z = 12$ represents a necessary and sufficient condition for rigidity in 2D and 3D respectively. Stated simply, this criterion implies that the higher connectivity provides more rigidity where connectivity, Z, is defined as the number of struts connecting to a node. This is illustrated simply in Figure 2.4, where the structure (a) has less connectivity leading to lower stiffness than structure (b). In this model, each strut was discretized into 28 Euler-Bernoulli ('B21') beam elements using ABAQUS/Standard (2018). In the case of planar lattices, stretching dominant lattices have a connectivity value, $Z = 6$,and bending dominant lattices have a connectivity value of 3. However, some lattices whose connectivity is a transition value numbering between 3 and 6, can exhibit stretching, bending dominating or mixed, depending on the loading condition and the perfection of the lattices. Examples of this are Kagome and diamond lattice with a connectivity of 4. If the Kagome lattice contains defects or missing struts, a large region around those imperfections shows bending dominated deformation [11, 70, 81]. On the other hand, the diamond lattice behaves as stretching dominated when the external load is applied along the strut's axial orientation. Otherwise, it will display bending dominated deformation [23, 80].

2.2 Quantification of 2 Dimensions (2D) Lattice

2.2.1 Representative Volume Element (RVE) and Relative Density of Periodic Cellular Materials

The design of structures featuring cellular materials requires the knowledge of the elastic behavior in the context of the cell topology, strut geometry and bulk material behavior. Further, the density of the lattice is another parameter useful in evaluating the material's

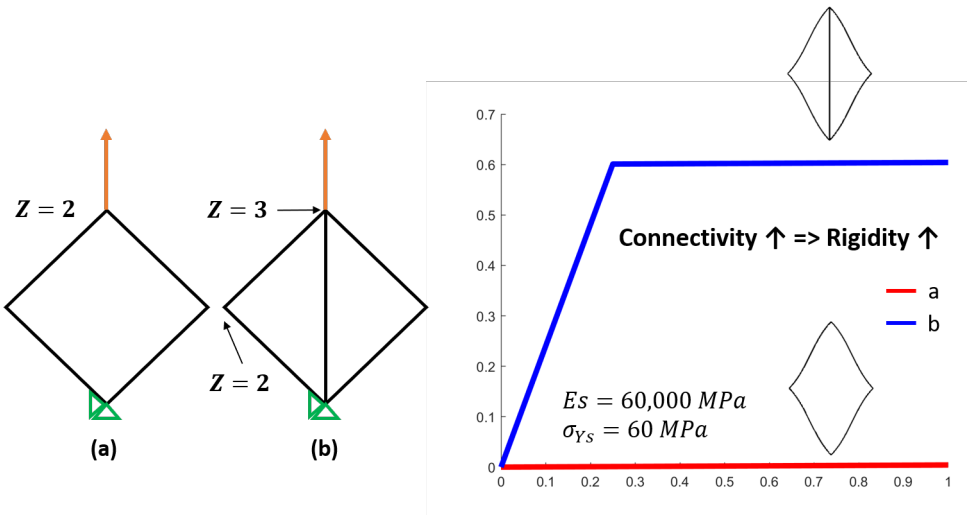


Figure 2.4: Qualitative illustration of the relation between connectivity and rigidity. Force-displacement plot obtained by Finite Elemet (FE) simulation assuming an elastic-perfectly plastic behavior for each strut. The Young's modulus and the yield stress of materials was $E_s = 60000$ MPa and $\sigma_{Ys} = 60$ MPa respectively with Poison ratio of 0.3.

performance. In periodic lattices, the mechanical properties of lattice can be calculated via a micro-mechanic model of a Representative Volume Element (RVE). This element represents the smallest volume that can describe the entire lattice structure. Some examples of RVE of periodic planar lattices are shown in Figure 2.5.

From the definition of RVE, each lattice structure exhibits a geometric characteristic that relates the volume occupied by a member of the lattice structure to the total volume occupied. This parameter is referred as relative density, $\bar{\rho}$, which plays an important role in evaluating mechanical properties of lattice materials. This parameter is defined as below :

$$\bar{\rho} = \rho^*/\rho_s \tag{2.3}$$

with ρ^* as the equivalent density of lattice and ρ_s as the solid wall density. Since the mass of lattice equals the total mass of strut walls, the relative density can also be expressed as the volume fraction of a solid :

$$\bar{\rho} = V_s/V^* \tag{2.4}$$

The ratio of solid wall volume, V_s, to topological volume, V^*, can be described as $1 - p$,

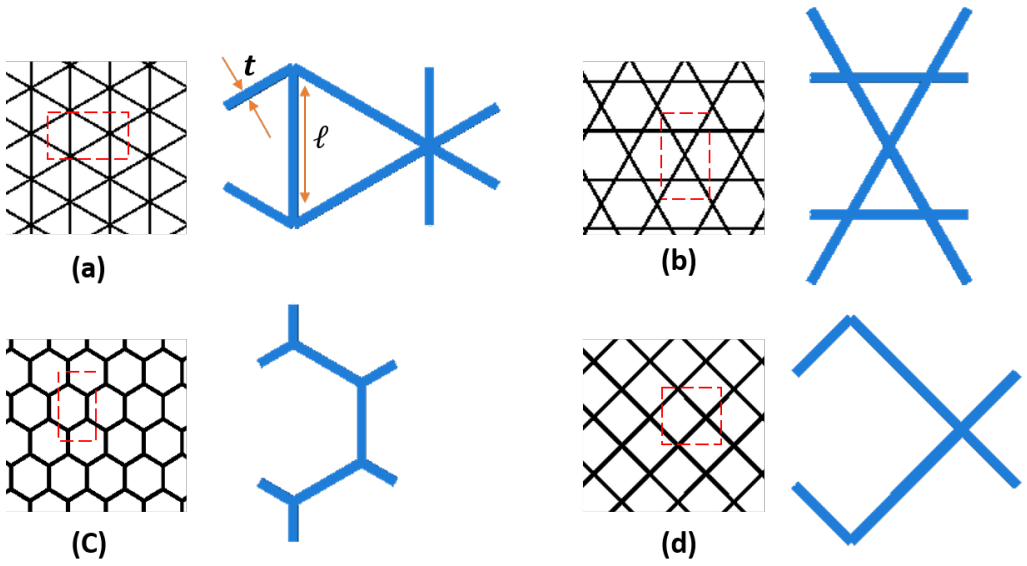


Figure 2.5: Representative cells of (a) Triangular (b) Kagome (c) Hexagonal (d) Diamond lattice

where p is the porosity of the material. Note that the porosity has to be high enough to be recognized as being a cellular structure. Otherwise, it is seen as a solid containing isolated pores.

2.2.2 In-plane Properties of 2D Lattices

Knowing the RVE and $\bar{\rho}$, the in-plane properties can be formulated accordingly as described in the following section. First of all, the definition of the volume fraction in Eq. (2.4) implies that the volume of solid wall is proportional to the dimension of strut, $\ell \cdot t$, and that the topological volume is proportional to the unit cell size ℓ^2. Therefore, $\bar{\rho} \propto t\ell/\ell^2 \propto t/\ell$, and it is consistent with the formula. Thus, the relative density can be expressed as:

$$\bar{\rho} = \hat{A}\left(\frac{t}{\ell}\right) \tag{2.5}$$

, where $\hat{A}$ is a coefficient relating to cell topology, t is the in-plane thickness of a strut, ℓ is the length of a strut and $t << \ell$. Again, from the Eq. (2.5), it is noted that the relative density depends only on the strut geometry and cell topology. However, when the relative density

is larger ($\bar{\rho} > 0.2$), the Eq. (2.5) will overestimate the relative density. This deviation is caused by calculating the corners of the cell twice, a so-called double-counting [23]. For 2D lattices, the Eq. (2.5) is adjusted and becomes:

$$\bar{\rho} = \hat{A}\frac{t}{\ell}\left(1 - \hat{a}\frac{t}{\ell}\right) \tag{2.6}$$

, where the second constant in the correction term is determined via a trivial geometry. Under the uni-axial tension, the stiffness of the lattice can then be expressed with $\bar{\rho}$ as

$$E = \hat{B}\bar{\rho}^{\hat{b}}E_s \tag{2.7}$$

and so does the strength as

$$\sigma_f = \hat{C}_1\bar{\rho}^{\hat{c}}\sigma_{f_s} \tag{2.8}$$

for fracture strength or

$$\sigma_Y = \hat{C}_2\bar{\rho}^{\hat{c}}\sigma_{Y_s} \tag{2.9}$$

for yield strength. All the coefficients ($\hat{A}, \hat{a}, \hat{B}, \hat{b}, \hat{C}$, and $\hat{c}$) in the aforementioned formulas depend only on the geometry of the structure and can be found in Table 2.1. The variation of exponent in $\bar{\rho}$ as a function of this topology can be explained by the following dimensional derivations. For stretching dominant cases such as the triangular and the Kagome lattice, only the axial force in the strut is considered and the global load, P, is proportional to the local load in the strut, P_s. From the relations $P_s \propto E_s bt$ and $E \propto P/(b\ell)$ with b as out-of-plane thickness, it follows that:

$$E \propto E_s\,(t/\ell) \tag{2.10}$$

Following a similar argument with the relations $\sigma \propto P/(b\ell)$ and $\sigma_s \propto P/(bt)$, the relation between global stress σ and local stress σ_s can be written as $\sigma b\ell = \sigma_s bt$. The fracture strength can be seen as the ultimate universal load, and thus the relation turns into:

$$\sigma = \sigma_s(t/\ell) \tag{2.11}$$

On the other hand, for bending dominant cases such as hexagonal and diamond lattices, all struts are considered as cantilever beams. The moment acting on the beam is

proportional to the local stress as shown in:

$$M \propto \sigma_s b t^2 \tag{2.12}$$

The nominal load causes a bending moment and the relation between them is:

$$M \propto \sigma b \ell^2 \tag{2.13}$$

From the Euler-Bernoulli beam theory, each strut has the following relation with the deflection δ_s and moment:

$$\delta_s \propto M\ell^2/(E_s I) \tag{2.14}$$

, where I is the inertia of the beam strut. Furthermore, the relation between the extension of the lattice and the deflection of the strut is approximated as $u \propto \delta_s$. Given that $E \propto \sigma/\epsilon \propto \sigma\ell/u$, the following stiffness relation:

$$E \propto E_s I/\ell^3 \propto E_s(t/\ell)^3 \tag{2.15}$$

with $I \propto bt^3$, can be obtained by substituting $\sigma - M$ relation and $\delta_s - M$ relation. Again, repeating the same operation and rewriting the relation between local and global stress via the bending moment, produce:

$$\sigma_f \propto \sigma_{f_s}(t/\ell)^2 \tag{2.16}$$

assuming that the rapture stress has been reached. Replacing t/ℓ with $\bar{\rho}$ by Eq (2.5), the value of exponent for the bending dominant case is clarified.

In conclusion, the properties of cellular solids only depend on the geometry of lattice, relative density and the solid wall materials. A similar demonstration can also be found in [25], while alternative derivations can be found in [23,82]. Thanks to previous researchers, in Table 2.1, the value of coefficients $\hat{A}$ and $\hat{a}$ have been proved in [23, 53]; the value of $\hat{B}$ and $\hat{b}$ are taken from [23]; the value of $\hat{c}, \hat{C}_1$ and $\hat{C}_2$ are taken from [23, 53, 82, 83], and the value of $\hat{B}, \hat{b}, \hat{c}, \hat{C}_1, \hat{C}_2$ have also been verified at $\bar{\rho} = 0.1 \sim 0.3$ in [82]. Furthermore, it is noted that when $t/\ell < 0.2$ under uni-axial loading, the effects from axial and from shear deformation of a cell-edge remain insignificant for bending dominant lattices. Otherwise a correction term must be calculated [23].

Table of Constant

Lattice	Z	ν_{12}	$\hat{A}$	$\hat{a}$	$\hat{B}$	$\hat{b}$	$\hat{C}_1$	$\hat{C}_2$	$\hat{c}$	$\hat{D}_I$	$\hat{D}_{II}$	$\hat{d}$
Tri	6	1/3	$2\sqrt{3}$	$\sqrt{3}/2$	1/3	1	1/3	1/3	1	0.607	0.404	1
Kago	4	1/3	$\sqrt{3}$	$\sqrt{3}/8$	1/3	1	1/2	1/2	1	0.205	0.115	1/2
Diam	4	1	2	1/2	1/4	3	1/6	1/4	2	0.216	0.225	1
Hex	3	1	$2/\sqrt{3}$	$2/(2\sqrt{3})$	3/2	3	1/3	1/2	2	0.902	0.408	2

Table 2.1: Coefficients for relative density $\bar{\rho}$, stiffness E, strength σ_u(for elastic-brittle or perfectly plastic case), and fracture toughness K_{IC}(for elastic-brittle case). Noting that $\hat{C}_1$ and $\hat{C}_2$ represent fracturing and yielding and that I and II mean Mode I and Mode II. The table is modified from [23] [24] [25] [19].

2.3 Fracture Toughness of The Lattice

Unlike the other mechanical properties mentioned above, the prediction of fracture toughness of cellular material depends on the brittleness of the cell-wall materials. Generally, there are three types of fracture modes: (1) opening mode (Mode I) which occurs when applied load is perpendicular to the crack plane which the stress field is symmetry to (Figure 2.6(a)), (2) shear mode (Mode II) which allows one crack face slide with respect to the other along the crack plane (Figure 2.6(b)), and (3) antiplane shear mode (Mode III) which makes the crack faces slide along the plane perpendicular to the crack plane (Figure 2.6(c)). The Mode I fracture toughness, which represents the critical value of the Mode I Stress Intensity Factor (SIF) at crack onset, can be calculated as follows [23]:

$$K_I = \hat{D}\bar{\rho}^{\hat{d}}\sigma_{fs}\sqrt{\ell} \tag{2.17}$$

where $\hat{d}$ and $\hat{D}$ are coefficients dependent to the type of lattice.

2.3.1 Definition of Stress Intensity Factor (SIF) and Stress Field around a Crack Tip

Assuming a center crack with length of $2a$ in an infinite honeycomb plate (Figure 2.7), with the semi-inverse method developed by Westergaard the opening-mode stresses of this

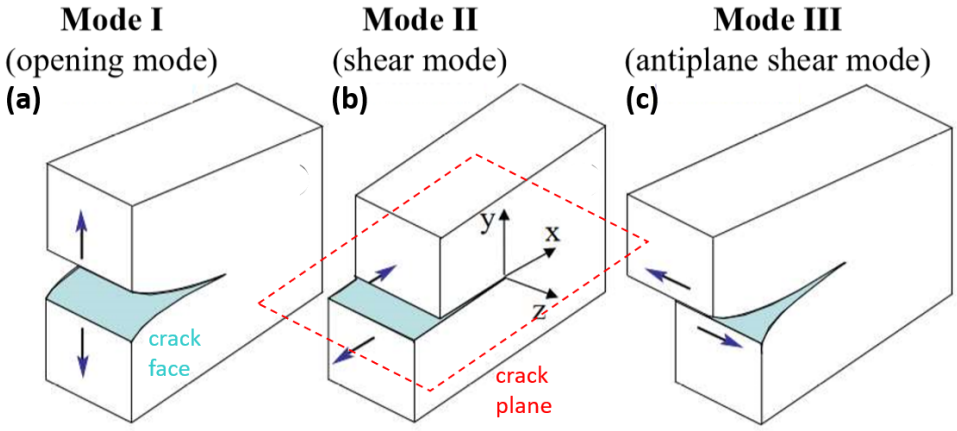


Figure 2.6: Three types of fracture mode: (a) Mode I (b) Mode II and (c) Mode III. The red dash line and blue faces represent the crack plane and the crack faces respectively.

plate are described as:

$$\begin{cases} \sigma_x = \frac{K_I}{\sqrt{2\pi|\vec{r}|}} cos(\frac{\theta}{2}\left(1 - sin(\frac{\theta}{2})sin(\frac{3\theta}{2})\right) + ...; \\ \sigma_y = \frac{K_I}{\sqrt{2\pi|\vec{r}|}} cos(\frac{\theta}{2}\left(1 + sin(\frac{\theta}{2})sin(\frac{3\theta}{2})\right) + ...; \\ \tau_{xy} = \frac{K_I}{\sqrt{2\pi|\vec{r}|}} cos(\frac{\theta}{2})cos(\frac{3\theta}{2})sin(\frac{\theta}{2})... \end{cases} \tag{2.18}$$

where θ is an angle between the crack and vector, $\vec{r}$, directed to an arbitrary point around crack tip from the crack tip. The vector $\vec{r}$ has length of $|\vec{r}|$. For a region close enough to the crack tip, the second and higher order terms in Eq. (2.18) are ignored and the mode I SIF can thus be approached as:

$$K_I = \sigma\sqrt{\pi a} \tag{2.19}$$

where a is crack length and σ is remote stress. It shows that SIF depends on the dimension of crack and load.

2.3.2 Derivation of K_{I_C} of The Lattices

Considering a singular stress field, σ_r, which develops ahead of the crack tip, gives the a local stress:

$$\sigma_r = \frac{\sigma\sqrt{\pi a}}{\sqrt{2\pi r}} \tag{2.20}$$

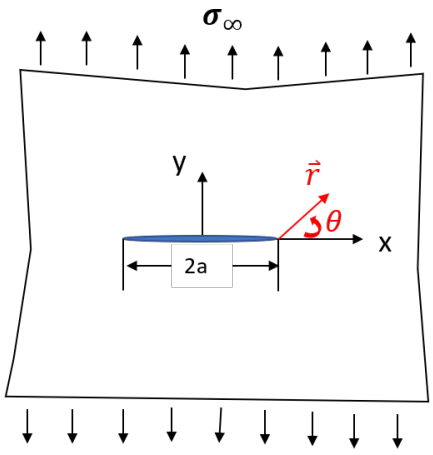


Figure 2.7: Configuration of an infinite plate with center crack under stretching at y-direction.

where r is the distance ahead of the crack tip. Assume that the first failure on strut occurs at fracture strength of the solid wall σ_{fs}. This gives $\sigma \propto \sigma_r \propto \sigma_{fs}$. Substituting for σ with σ_f in Eq 2.19, gives

$$K_I \propto \sigma_f \sqrt{2\pi r} \tag{2.21}$$

with $r \propto \ell$. By Eq. (2.5), the Eq. (2.11) can be written as

$$K_I \propto \bar{\rho}^c \sigma_{fs} \sqrt{\pi \ell} \tag{2.22}$$

However, since the Kagome lattice has a connectivity of 4, it performs both bending and stretching deformation. This leads to a different exponent value, which is elaborated in [53]. The diamond lattice also has a different exponent value as has been discussed in [24]. To date, many studies have been proposed to evaluate the fracture toughness of planar lattices. All previous works have been organized by [26], see Table 2.2.

To improve the accuracy of the prediction of fracture toughness for elastic-brittle cellular materials, other methods have been developed. These include the boundary-layer analysis for cracked lattice under K-control [53], the stochastic process using Weibull distribution [43], the continuum theorem with K-field [52], the representative cell method using discrete Fourier transform [54] and the approach accounting for fracture toughness of base materials

[25].

2.4 Validation of Numerical Model Through Periodic Boundary Conditions (PBCs)

One of the common methods to evaluate the properties of lattice is implementing the periodic boundary condition using finite element analysis (FEA). Since modeling a lattice in large scale uses an excessive number of elements, which leads to the enormous computational cost, the use of PBCs on a small RVE enables a significant reduction of computational cost.

Table of Constant

-	Mode I	-	Mode II	-	-
Lattice	D	d	D	d	Reference
Tri	0.607	1	0.404	1	Fleck and Romijn [24]
-	0.5	1	0.38	1	Fleck and Qiu [53]
-	1.328	1	0.433	1	Chen et al. [52]
Kago	0.205	1/2	0.115	1/2	Fleck and Romijn [24]
-	0.212	1/2	0.133	1/2	Fleck and Qui [53]
Hex	0.902	1	0.408	1	Fleck and Romijn [24]
-	0.8	2	0.37	2	Fleck and Qui [53]
-	0.156	1	0.667	1	Chen et al. [52]
-	0.4	2	–	–	Gibson and Ashby [23]
-	0.3225	2	–	–	Huang and Chiang [50]
Diam	0.216	1	0.225	1	Fleck and Romijn [24]
-	0.125	1	0.25	1	Lippenm et al.] [54]
-	0.32	2	–	–	Huang and Chiang [50]
-	0.415	2	–	–	Huang and Gibson [43]

Table 2.2: Coefficients of fracture toughness for various lattices. The table is adapted from [26].

2.4.1 Theory of PBCs

PBC, briefly speaking, means all the cells in the lattice will response and deform in the same way that representative cell does as demonstrated in Figure 2.8(a). The relation of each boundary on certain RVE can be described as Figure 2.8(b). u_{ij} is displacement of node on boundary $_{ij}$ and u_k represents displacement of vertex v_k. By doing so, all nodes at the boundary of RVE will deform periodically to present the deformation of infinite plate.

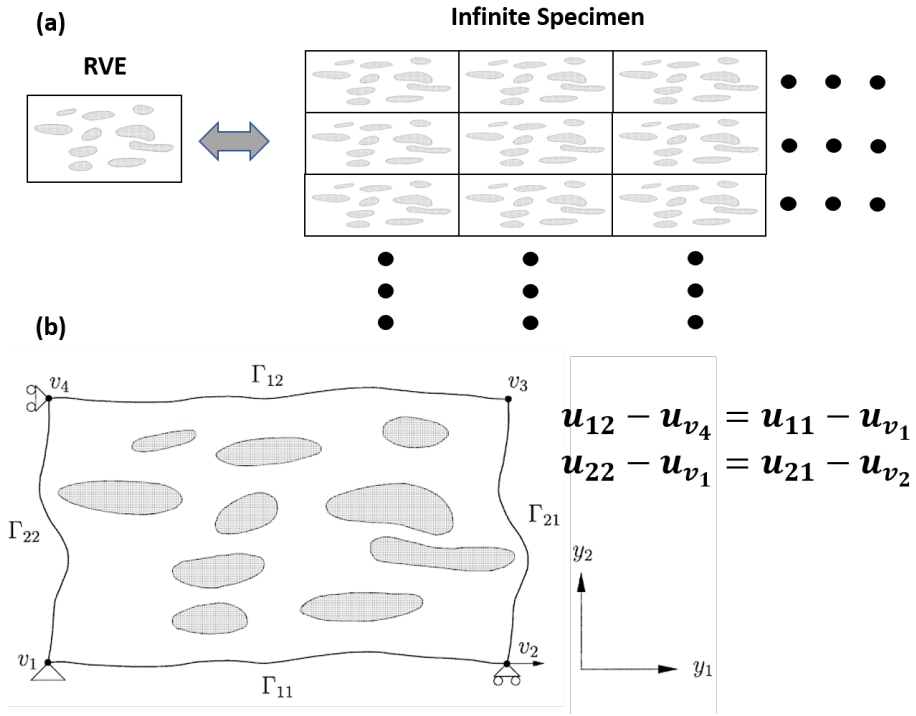


Figure 2.8: (a) The configuration depicting the relation between infinite plate and its RVE; (b) Schematic of a periodically deformed unit cell showing how a PBCs model works; v represents a vertex and Γ is a boundary at edge of unit cell. Adapted from [21]

2.4.2 Limitation of PBCs When It Comes to Damage in Lattices

However, given the features of the PBC that one representative unit-cell characterizes an infinitely large lattice, an appearance of damage in either one strut could be treated as the periodic damage everywhere in the infinite lattice simultaneously, which is unreasonable in the reality. This consequence can lead to an extra energy dissipation from formation of

periodic crack which should not exist in the structure as illustrated in Figure 2.9. In fact, the localization will start from the certain region and the crack will propagate in forming a localized band structure. As a result, PBC cannot be applied to capture the localization during the damage propagation in nature when analyzing the fracture behavior.

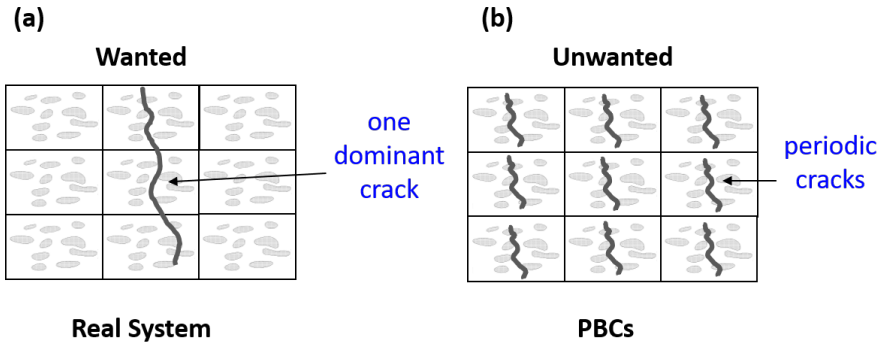


Figure 2.9: Comparison of damage process in (a) a real system and (b) a PBCs model.

2.4.3 Python Script for PBCs

In spite of limitation in simulating fracture behaviors, due to its simplicity in pre-processing, PBC becomes a common numerical approach for simulation containing mechanical periodicity and has been used to approximate the mechanical properties of 2D lattice including stiffness, yield stress, etc. Fleck et al. [19] used PBC to predict the tensile response of 2D lattices composed of elastoplastic material at finite elongation. The type of lattices includes triangular, Kagome, diamond and hexagonal in a certain direction. The result of their simulations had a good agreement with the analytical prediction, which is derived by the micro-mechanics model in a view of RVE. This numerical method demands no large or complicate geometry generation but its ease of use may depend on the selection of the RVE. The Python script for generating the PBC model for 4 types of periodic planar lattices shown in Figure 2.5 is provided. It is typed in pseudo-code for understanding the algorithm in Figure 2.10.

Algorithm 1: Algorithm of PBC

Input : A string of the name of the lattice and three floats of geometry features including characteristic angle θ, dimension of strut L and the unit cell size factor k

Output: A unit cell of certain lattice with PBC

```
1  # Sketch of RVE
2  Sketch.line(theta,L,k) %strut a1 of RVE
3  ...
4  # Build Vertices sets for Constraint equations
5  Part.Set(name=v1, vertices=Part.vertices.findAt(a tuple of coordinates))
6  ...
7  # Collect nodes from edge and build a set of nodes
8  for i in AllNodesonEdge do
9  |  NodeList = NodeList+[(i.coordinates[y],i.label)]
10 end
11 NodeList.sort() # Arrange nodes in order
12 # Create sets of single node with label in order
13 for j in NodeList do
14 |  # create a set of single node j
15 end
16 # Building relation equations for matching nodes of the pair of corresponding edges
17 for i in range(1, len(NodeList)-1) do
18 |  # Writing PBC node by node in rise order
19 |  NodeofEdgev1v2
20 |  NodeofEdgev6v8
21 |  order = order+1
22 |  # Writing boundary relations with vertices via equations
   |   model.Equation(constraint displacement in x-direction)
   |   model.Equation(constraint displacement in y-direction)
23 end
24 # Repeat line 10-22 for each pair of corresponding edges of RVE
```

Figure 2.10: The Python script for the PBC model is typed in pseudo code.

2.5 Multi-cell Lattice Model

To well simulate the damage in lattices, it is necessary to utilize a multi-cell lattice model. This model can realistically present the damage process that takes place in lattices. In order to generate the geometry of lattices with multiple cells in Abaqus, a built-in Python module in Abaqus provides efficiency and flexibility for the user's needs. Instead of plotting each unit cell one by one manually in the Abaqus graphic user interface(GUI), the Python script can easily run a loop to do a routine job such as an arrangement of unit cells, which saves lots of time and effort on geometry generation. In this section, a pseudo code is demonstrated for understanding the algorithm (Figure 2.11). For more detail, check the Python script for the generation of the multi-cell lattice model in Appendix A.

Algorithm 2: Algorithm of geometry generation for the multiple cells model

Input : A string of the name of the lattice, two nonzero positive integers *TotalRow* and *TotalColumn* ,and three floats of geometry features including characteristic angle θ, dimension of strut L and the unit cell size factor k

Output: A system of geometries of the lattice

```
for i in range(1,1+TotalColumn) do
    for j in range(1,1+TotalRow) do
        Sketch.line(theta,L,k) #strut a of RVE
        Sketch.line(theta,L,k) #strut b of RVE
        Sketch.line(theta,L,k) #strut c of RVE
        ...
    end
end
```

Figure 2.11: The Python script for the multi-cell lattice model is typed in pseudo code.

2.6 Computational Comparisons Between the result of Multi-cell Model and that of PBCs Model

To conduct the comprehensive study of the cellular materials, both the RVE model with PBC and the full scale multi-cell lattice model were developed for numerical analyses under Finite Element (FE) framework. The comparison of the results of theoretical prediction,

PBC and the multi-cell lattice model were presented. All lattices reported here were simulated at $\bar{\rho} = 0.1$ (both PBC and multi-cell lattice model) so that their beam slenderness is under a condition $t/L < 0.2$. By doing so, the prediction of $\bar{\rho}$ can be approached by Eq. (2.3), and the entire model can be approximated as a network of interconnected, linear, one-dimensional beams in a 2D system [84]. These beams can be interpreted as Timoshenko beam elements (of type B21 in Abaqus element code) with the same material properties as those used in [19]. In addition, at $t/L < 0.2$, the axial and shear deformation of the strut of bending dominant lattices are insignificant [23]. Accordingly, the prediction of modulus is not recalculated with the correction term. Each type of lattice has at least 10×10 representative cells in the multi-cell lattice model. The boundary conditions for PBC are shown in Figure 2.12(a), and those for the multi-cell lattice model are shown in Figure 2.12(b). A mesh convergence study revealed that a strut's mesh number of 150 in the multi-cell lattice model and of 50 in the model using PBC are adaquate for relative density at $\bar{\rho} = 0.1$ due to the errors being within 1% with respect to Young's modulus and the model having a yield strength prediction of perfect lattice derived in [19].

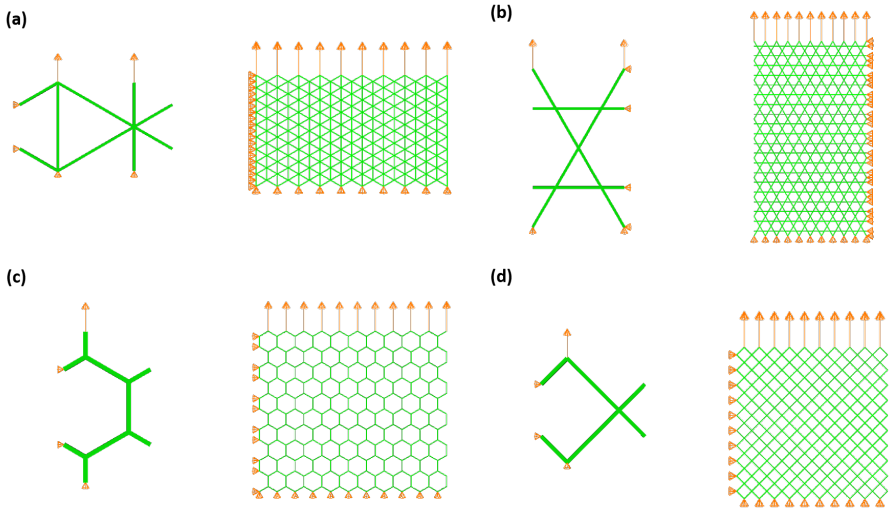


Figure 2.12: The boundary conditions for PBCs and multi-cell lattice models.(Render beam profiles scaled by 1.)

The results are summarized in Figure 2.13. The measured microscopic nominal stress σ normalized by the yield stress of cell-wall material Y_s is plotted versus nominal strain. The nominal stress is defined in terms of the sum of reaction force P_{sum} on the top nodes over the out-of-plane cross-section A as $\sigma = P_{sum}/A$. Here A is determined by gauge width w and out-of-plane thickness b denoted as $A = wb$. The nominal strain is measured as the extension u of the gauge length h as $\epsilon = u/h$.

The agreement between these two models is valid due to an error of less than 0.1% for the linear elastic region and 1% for yield stress. In general, the nominal stress of the lattice increases due to densification. This means that the lattice gets squeezed and it becomes solid and bulky. This causes the measured stress close to the strength of cell-edge materials. The further details of each type of lattice are discussed below.

2.6.1 Triangular Lattice

The two models align well during Stage I and Stage II. However, the densification takes place later in the Multi-cell lattice model than in the PBCs model when plotted in stress-strain response as shown in Figure 2.13(a). This also leads to deviation even during post-densification (Stage IV). The multi-cell triangular lattice starts to densify from its boundary, and than gradually towards the inner part. By contrast, all unit cells of lattice in PBCs model start to densify simultaneously, causing the earlier densification phenomenon.

2.6.2 Kagome Lattice

Note that the pit shown in the curve of the Kagome lattice represents buckling behavior. This behavior occurs earlier in the results of the multiple cell lattice model than in the PBCs model because such buckling happens on each unit cell simultaneously when using PBCs. In the case of the Kagome lattice, the buckling failure occurs first because the free edges at the boundary layer in the model cannot carry the load. Thus, the other struts have to carry more stress and they reach buckling strength faster. Such deviation would be compensated for if the size of the structure in the multiple cell lattice model were increased. In doing so, the area fraction of those stress-free (non-load-carrying) cell walls decreases, moving the

results closer to that of PBCs. In addition, the densification response is not obvious in Kagome lattice due to the buckling failure. This can be depicted by the less-steep climbing stress, due to the densification, from the stress-strain curve in Figure 2.13(b).

2.6.3 Hexagonal Lattice

It is worth mentioning that the deviation of these two models happens only during densification (Stage III) and they align again during post-densification (Stage IV). This phenomenon is caused by the struts at the boundary. Those struts are load free and make the stress lower during the process of densification until the lattice has been fully squeezed. This deviation will reduce when using a lattice of larger size.

2.6.4 Diamond Lattice

For the case of diamond lattice, the two models align well with each other from Stage I through Stage IV (Maximum error is less than 1%). Since, diamond lattices are more compliant than even hexagonal lattices at the same relative density, all cells are more subject to deforming at the same time, which can mitigate the boundary effect.

In conclusion, the triangular lattice and the Kagome lattices have higher stiffness and strength. The hexagonal lattice, on the other hand, has a longer plateau before it starts to densify. This implies that it potentially has better ductility.

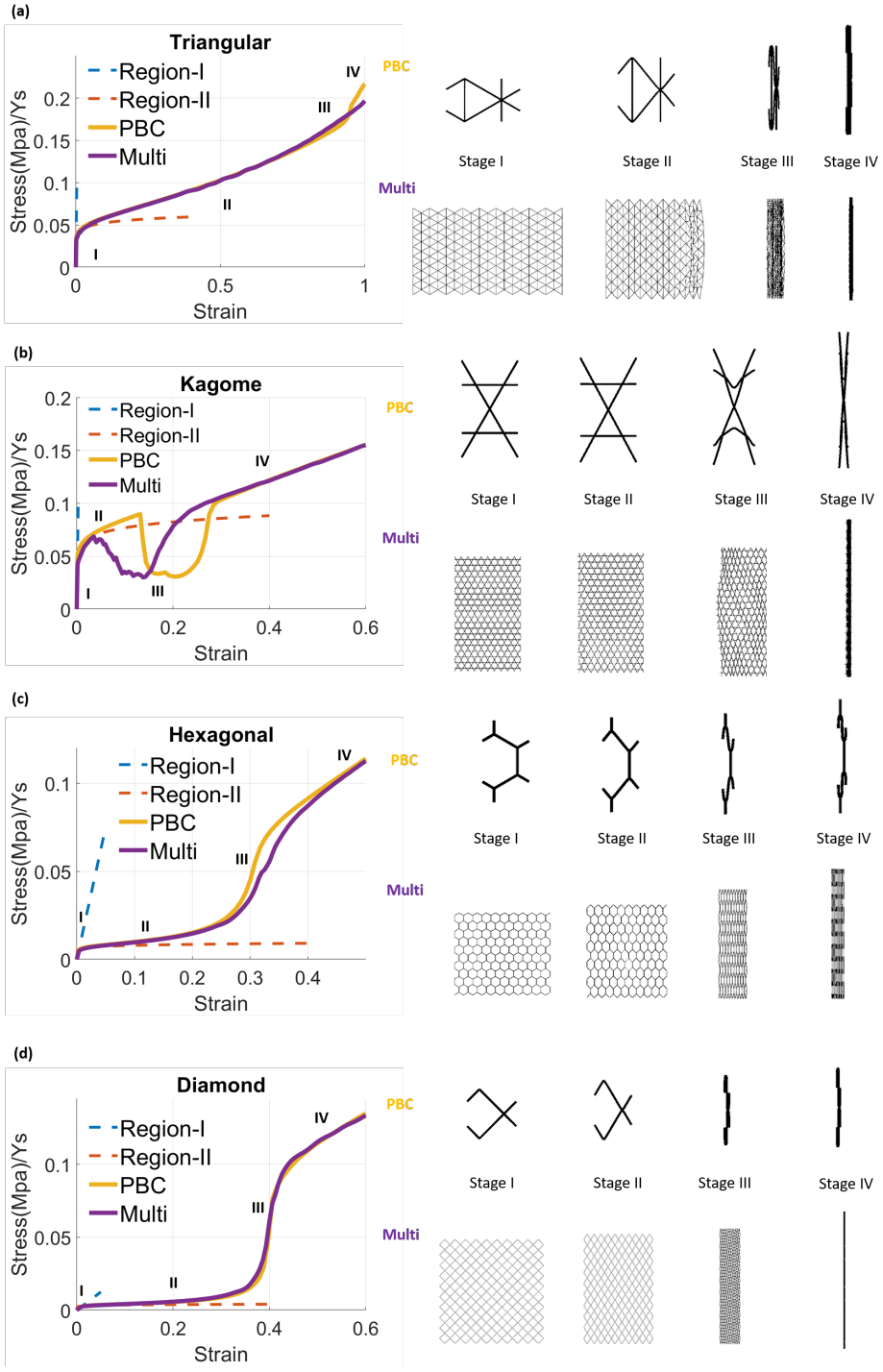


Figure 2.13: The results of the simulation including (a) Triangular lattice, (b) Kagome lattice, (c) Hexagonal lattice and (d) Diamond lattice; Region I and Region II represent linear elastic behavior and yield behavior respectively. Stage I represents linear response; Stage II represents plastic behavior. Stage III represents densification behavior and Stage IV is post-densification showing severe distortion.

Chapter 3

COHESIVE MODEL OF QUASI-BRITTLE MATERIAL

In this chapter, a quasi-brittle material response modeled using cohesive law will be presented, and the mesh convergence problem will be addressed. This will be followed by a simple, but efficient solution. To more fully understand quasi-brittle materials, its material properties are discussed and compared to that of elastic-brittle materials, with a particular focus on the differences between their fracture process zone (FPZ).

3.1 Definition of Quasi-brittle Fracture Behavior

A Quasi-brittle response is distinguished by a linear elastic behavior that is expressed up to the ultimate strength, followed by a gradual softening, with notable energy dissipation during the damage process. This is shown in the stress-strain curve in Figure 3.1(a). Typical quasi-brittle materials are defined as heterogeneous materials with brittle constituents. This type of material includes concrete, mortars, toughened ceramics, bone, wood, shell, rigid foams, polymers and porous printed materials. When the dimensions narrow down to micro- or nano-scale, the materials can exhibit quasi-brittle behavior.

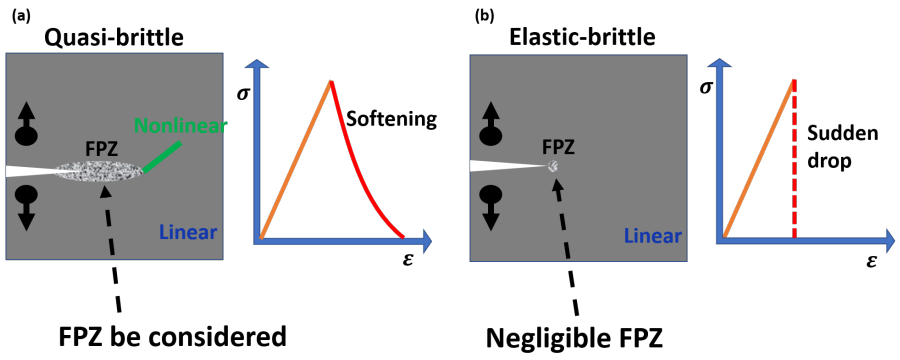


Figure 3.1: Comparison of the sizes of FPZ between Quasi-brittle and Elastic-brittle material and their effect on fracturing response.

3.2 Fracture Process Zone In Quasi-brittle Material

When a fracture takes place in material under an increasing load process, the material undergoes progressive softening damage in the form of randomly distributed micro-cracking and frictional micro-flips. This zone is referred to as the fracture process zone (FPZ) where the densely distributed damage induces in the material a non-linear response with notable dissipated energy [22, 85]. Such FPZ is dominant in quasi-brittle material and can be described phenomenologically in Figure 3.2 and can be explained as follows [22, 85].

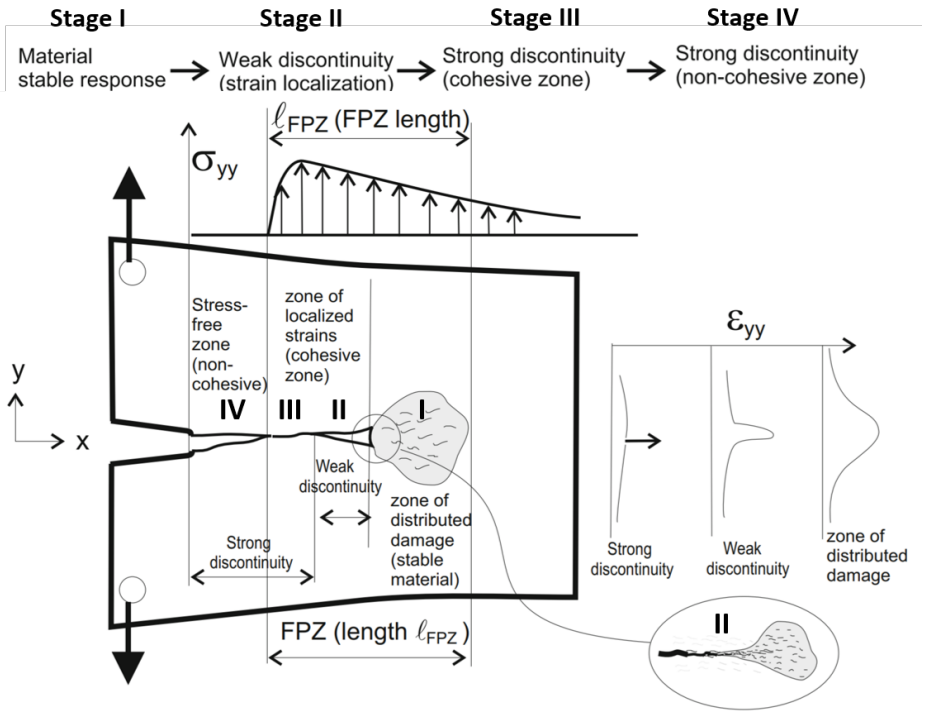


Figure 3.2: Schematic of an idealized crack advancing in quasi-brittle materials. Stress σ_{yy} is displayed along the fracture process zone and distribution of strains ϵ_{yy} is depicted in three different stages of the process evolution.Adapted from [22]

In the beginning of the quasi-brittle fracture process, the material remains macroscopically stable even when the strains are becoming larger in a small bounded region. This stage (stage I) is shown in gray in Figure 3.2. When the loading increases, micro-cracks take place and start to grow. The interaction among those short cracks causes instability

(stage II). Some micro-cracks merge together to form one dominant macro-crack around the cohesive region. Here the strain localization becomes observable at the macroscopic scale in this transition zone, as zoomed in, in Figure 3.2. With increasing loading, the primary macro-crack propagates through the material surface while continue to carry the load. The propagation of a macro-crack at this stage leads to a strong discontinuity under the cohesive field (stage III). This crack later splits the material surface into two pieces, and the cohesive force goes to zero and forms a stress-free zone on the crack surfaces (stage IV).

3.3 Comparison Between Elastic-brittle and Quasi-brittle Material

Unlike quasi-brittle material, elastic-brittle material is characterized by a linear elastic behavior up to the strength followed by a sudden collapse without any energy dissipation during the fracture process (Figure 3.1(b)). The size of the fracture process zone(FPZ) is relatively small compared to the size of the entire structure. Hence, the effect of this nonlinear damage zone is negligible, and the linear elastic fracture mechanics (LEFM) provides an accurate description of the fracturing behavior. Such elastic-brittle manners can be found in ceramic foams or glassy foams at room temperature or polymers at low temperature ($T < T_g$ where T_g is the glass transition temperature). On the other hand, FPZ can develop into non-negligible size compared to the entire structure for the quasi-brittle materials. As a result, cohesive stress inside the FPZ needs to be overcome for the crack to propagate. This phenomenon leads to the gradual reduction of stiffness in quasi-brittle materials in contrast to the elastic-brittle materials, which show a sudden drop in stiffness after the onset of damage, see Figure 3.1(b).

The size of a fully developed FPZ is constant. It is the important characteristic parameter for each material in describing fracturing behavior. Therefore, the nonlinear effect from this damage zone to the structural response is highly dependent on the size of the structure, as shown in Figure 3.3.

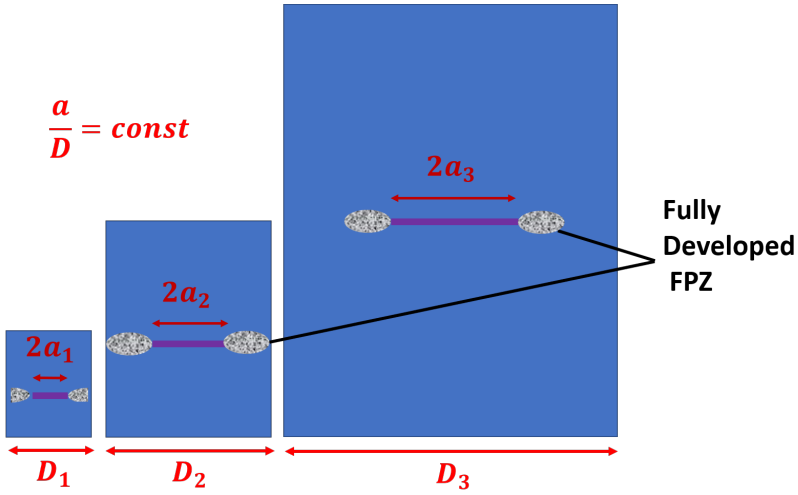


Figure 3.3: Comparison of the sizes of FPZ for each specimen with different scales

Quasi-brittle materials are used as the cell wall materials for UCMs in order to study the relationship between fracture behavior and material properties. The implementation of quasi-brittle behavior is achieved through the cohesive crack model with the linear-softening response for the sake of simplicity.

3.4 Quasi-brittle Cohesive Law Implementation in Finite Element Model

The simple linear cohesive law is implemented in the commercial computer aided engineering (CAE) software, Abaqus/Explicit (2018), in conjunction with the user subroutine, VUMAT, in order to model the softening behavior of the quasi-britte material. First, the linear elastic behavior described by Hook's Law was assigned as shown in Eq. (3.1), where σ is the engineering stress, E is the Young's Modulus and ϵ is the engineering strain.

$$\sigma = E\epsilon \tag{3.1}$$

The use of Hook's Law can be validated by an evaluation of the relationship between true strain and engineering strain. The true strain is defined as:

$$\epsilon_t = \int_{\ell_0}^{\ell_F} \frac{\mathrm{d}\ell}{\ell} = \ln \frac{\ell_F}{\ell_0} \tag{3.2}$$

for an object under tension or compression where ℓ is the length of the object, ℓ_0 is the original length and ℓ_F is the final length. Using definition of engineering strain, $\epsilon = \Delta\ell/\ell_0$ where $\Delta\ell$ is the total extension, Eq. (3.2) can be written as.

$$\epsilon_t = \ln(1+\epsilon) \tag{3.3}$$

This equation can be approached as $\epsilon_t = \epsilon$ by Taylor's expansion when ϵ is small enough. To maintain the accuracy, the elongation of 0.1% is defined when it reaches the ultimate stress so that its elongation is smaller in the linear elastic regime than elongation of 5%, which is recommended by the Abaqus user manual. The Eq. (3.1) expresses the relationship between the small displacement u_e and load F

$$u_e = \frac{Fh}{AE} \tag{3.4}$$

with characteristic length h and cross-section A of an element. After the elastic limit σ_u is achieved, the linear softening behavior is described as shown in Figure 3.4(a), according to

$$\sigma = \sigma_f \left(\frac{u - u_f}{u_p - u_f} \right) \tag{3.5}$$

with the total displacement u, the displacement at peak load u_p and the displacement at full failure u_f. u_f, here, is determined from a given fracture energy G_f. Hence,

$$u_f = 2G_f/\sigma_u \tag{3.6}$$

Now consider the one dimension model for a single bar meshed by a chain of n elements under tension. Based on the concept of localized zone of deformation, only one element softens while the others unload elastically. Accordingly, the total displacement δ of a single bar during damage can be expressed as

$$\delta = (n-1)u_e + u \tag{3.7}$$

Substituting u_e with Eq. (3.4) and u with Eq. (3.5), the relationship between δ and F becomes

$$\delta = (n-1)\frac{Fh}{AE} + u_f - (u_f - u_p)\frac{F}{A\sigma_u} \tag{3.8}$$

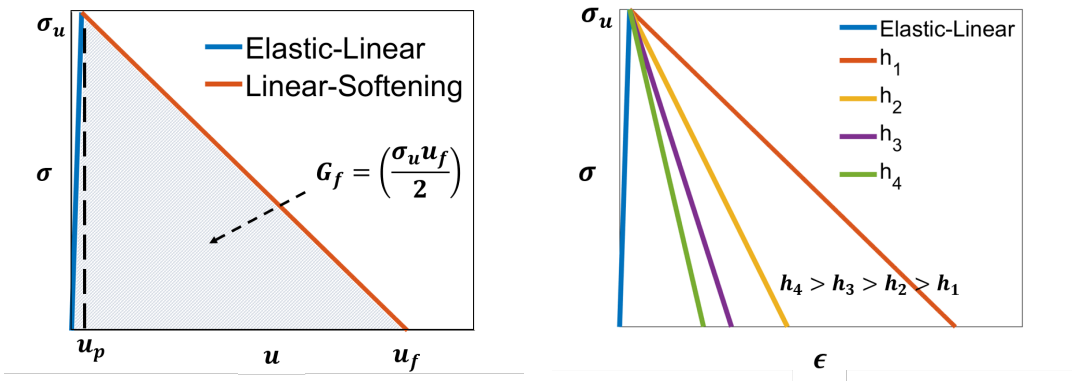
Again, by using Eq. (3.6) and the fact that $u_p = \sigma_u h/E$, Eq. (3.6) can extend to

$$\delta = \frac{Fnh}{AE} + \frac{2G_f}{\sigma_u} - \frac{2G_f F}{\sigma_u^2 A} \tag{3.9}$$

Since $L = nh$ the equation above can be finalized as:

$$\delta = \frac{FL}{AE} + \frac{2G_f}{\sigma_u} - \frac{2G_f F}{\sigma_u^2 A} \tag{3.10}$$

This implies that the cohesive crack model is independent of the length of the mesh element. The same equation can be derived in the cohesive traction-separation law as well. See [86,87].



(a) Cohesive law for crack model

(b) Scaled stress-strain relation

Figure 3.4: Comparison of (a) the idealization of load-displacement curve of cohesive crack law and (b) the use of stress-strain relation on cohesive crack law resulting to the scaled strain response

Since Abaqus VUMAT provides an incremental strain at each incremental time step, it is easy to achieve a constitutive relationship for the softening curve

$$\sigma = \sigma_u \left(1 - \frac{\epsilon - \frac{\sigma_u}{E}}{\frac{2G_f}{\sigma_u h} - \frac{\sigma_u}{E}}\right) \tag{3.11}$$

However, if the softening curve is defined by the scaled crack strain law, shown in Figure 3.4(b), the strain at the ultimate failure will scale in a manner corresponding to the element length. The larger the element size is, the shorter the extension is. The force displacement response will, thus, be sensitive to the mesh. To demonstrate this induction using FE model

in Abaqus, a single bar of 1 mm is discretized into different numbers of elements (1, 10, 150, and 200) under a displacement control tensile test. The bar is clamped at the bottom to allow all degrees of freedom. The element type B21 is used to represent the Timoshenko beam in Abaqus/Explicit solver. These two-noded linear elements with one integration point can involve both bar stretching and bending and allow transverse shear deformation. The Young's modulus of $60,000$ MPa, Poisson's ratio, ν, of 0.3, strength of 60 MPa and fracture energy of 1.0 N/mm are given as the mechanical properties of materials with boundary condition shown in Figure 3.5(a). As expected, when the mesh is getting finer, the curve of the softening response goes up and its deviation clearly increases obviously as seen in the Figure 3.5(b).

It is worth noting that this phenomenon could cause a mesh convergence problem. To solve the issue, the cohesive law must be defined through the load-displacement relation. Instead of calculating the strain by incremental strain directly, the true strain ϵ_t is transferred to engineering strain ϵ to capture the displacement of element. This can be imported into Eq. (3.5). By definition of the true strain, Eq. (3.3) is applied. Since $\epsilon = u/h$, displacement, u, can be rewritten as

$$u = h\left(e^{\epsilon_t} - 1\right) \tag{3.12}$$

such that, by importing Eq. (3.6), the Eq. (3.5) becomes

$$\sigma = \sigma_f \left(\frac{h\left(e^{\epsilon_t} - 1\right) - 2G_f/\sigma_u}{\frac{\sigma_u h}{E} - 2G_f/\sigma_u} \right) \tag{3.13}$$

The verified results of this cohesive model is discussed in the following section. Both load displacement and stress-strain cohesive law for user-defined material properties are written in the programming language, Fortran 77, for the Abaqus subroutine called VUMAT and the scripts are attached in Appendix B.

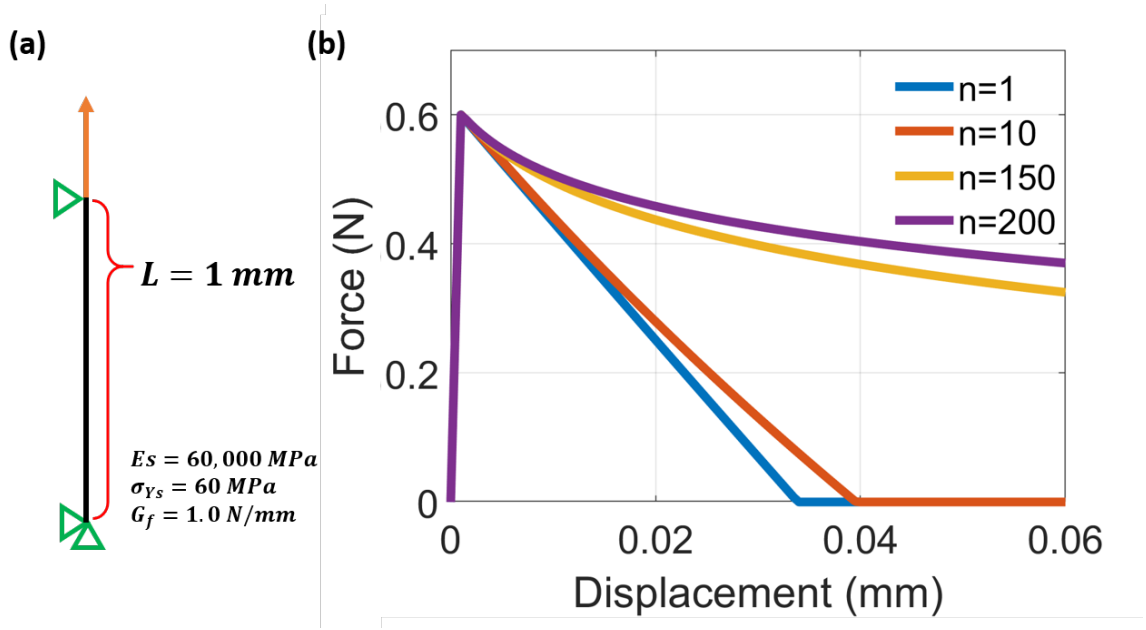


Figure 3.5: A single bar under a uniaxial tensile test using the stress-strain traction law. (a) Configuration and boundary condition of the single bar under the tensile test; (b) The corresponding force displacement response showing the sensitivity to element characteristic length (n=number of element.)

3.5 Cohesive Model Verification

To perform the model verification, the same tensile test model and material properties in Figure 3.5(a) are used, but under modified cohesive law instead of stress-strain traction law. As shown in Figure 3.6, the corresponding load displacement curves, for all the cases of element size, perfectly align to the assigned cohesive law for the material because the maximum deviation in the resulting peak force and stiffness between the two curves is a negligible 0.1%. This shows the objectivity of the element length.

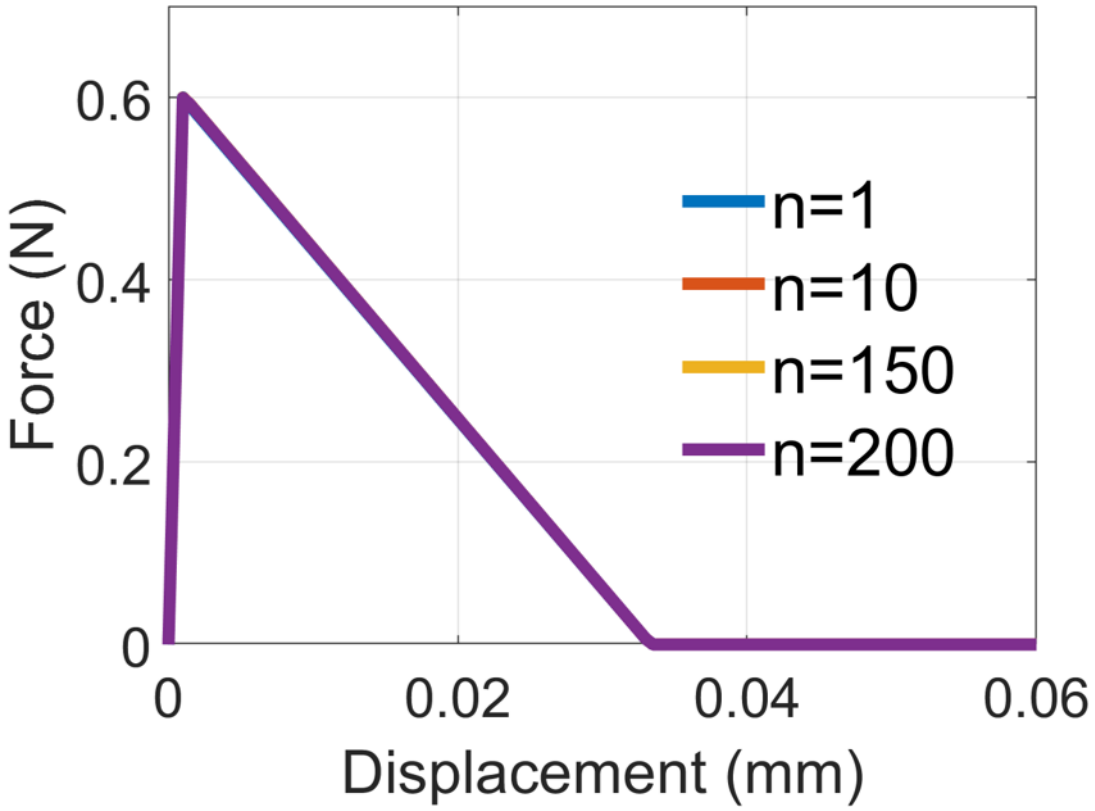


Figure 3.6: A single bar tensile test under stress-displacement traction separation law showing force-displacement response which can be noted that the results are independent to the size of element.

It is important to point out that the Abaqus/Explicit does not automatically compute the transverse shear modulus for the user when the user-defined subroutine such as VUMAT is used for material definition. Since the transverse shear stiffness is crucial to transmit force by cross-section of the beam, it is necessary to identify this property factor for beams manually according to the user manual. So it is clearly imperative to evaluate the imported shear stiffness, and the Cantilever beam bending model serves as a good test (Figure 3.7). The shear stiffness K_{13} and K_{23} can be determined as, $K_{a1} = k(GA)_{a1}$ where a represents either 1-direction or 2-direction, A is the cross-section area, G is elastic shear modulus determined by $G = \frac{E}{2(1+\nu)}$ and the section dependent shear factor k. Since the square

cross-section is used for beam elements in the aforementioned models, k is given with a recommended default value of 0.85 for any rectangular cross-section. The result of the bending test shows good agreement with the analytical solution (Figure 3.7(c)). Within the small deflection of 0.004 mm the maximum error is only 6%.

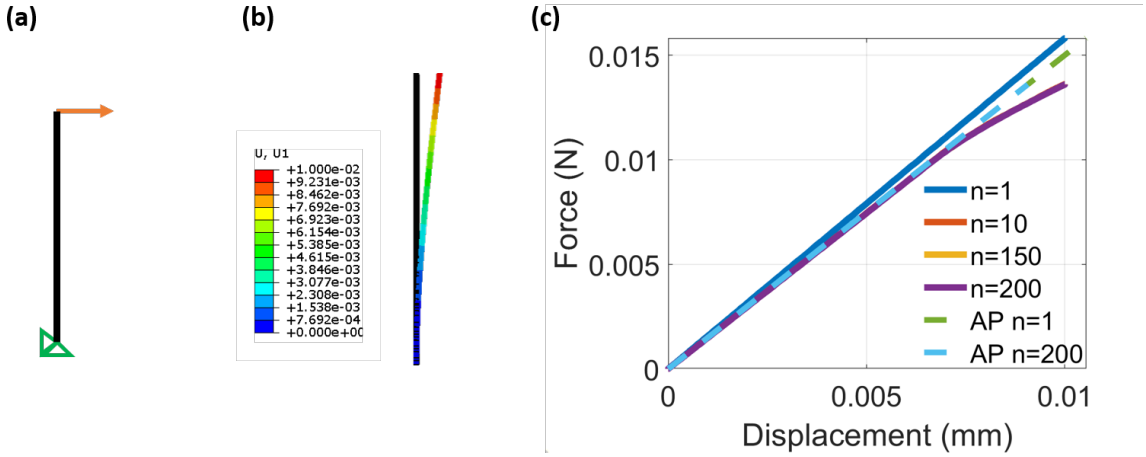


Figure 3.7: The Cantilever beam bending test under stress-displacement traction separation law; (a) Configuration and boundary condition of bending test; (b) Contour plot showing deflection; (c) Force-displacement response where the dash lines represent the analytical prediction (AP).

Chapter 4

MODELING AND DESIGN OF ULTRA-CELLULAR MATERIALS (UCMS)

4.1 Decision of Component Lattices for UCMs

Based on the previous results in Chapter 2, the following choice of lattice types are made to construct UCMs made of a quasi-brittle material and to study its damage tolerance and fracturing behavior. To improve the cellular structures, both stretching dominant (stiff) lattices and bending dominant (ductile) lattices are used. The base lattice is a triangular lattice which has stretching dominated deformation and high stiffness, but little compliance due to its connectivity of 6. The second choice is a hexagonal lattice. It has bending dominated deformation and ductility, but less rigid performance owing to its connectivity of 3. Such in-homogeneity coming from the combination of two types of lattice can take advantage of both stretching and bending dominant lattices. The UCM should gain the stiffness and strength from the stretching dominant lattice (triangular), and acquire the high energy absorption as well as the damage tolerance from the bending dominant lattice (hexagonal). Therefore, with in-homogeneity achieved via topology design only, a good compromise can be obtained in UCMs.

4.2 Schematic of UCMs

To combine two lattices (triangular and hexagonal), the concept of various length scales is discussed and applied. Owing to the higher damage tolerance in a hexagonal lattice, such lattice has a larger length scale (see Figure 4.1) in order to compensate for the crack propagation. By doing so, hexagonal lattice should help redistribute the load around the crack tip during the damage in a brittle lattice (triangular lattice). This stress re-distribution leads to non-locality and can help dissipate more energy.

These two lattices with different length scales are constructed by connecting each of the

coincident vertices via an imaginary ideal rigid body of negligible length. These connecting rods are assumed to act as a pinned-joint mechanism such that the translation of vertices is fixed, but a degree of freedom in rotation is provided. By doing so, there is only one way to combine these two lattices due to the topological limit for regular triangular and hexagonal honeycombs (matching edge length and angles at the same time).

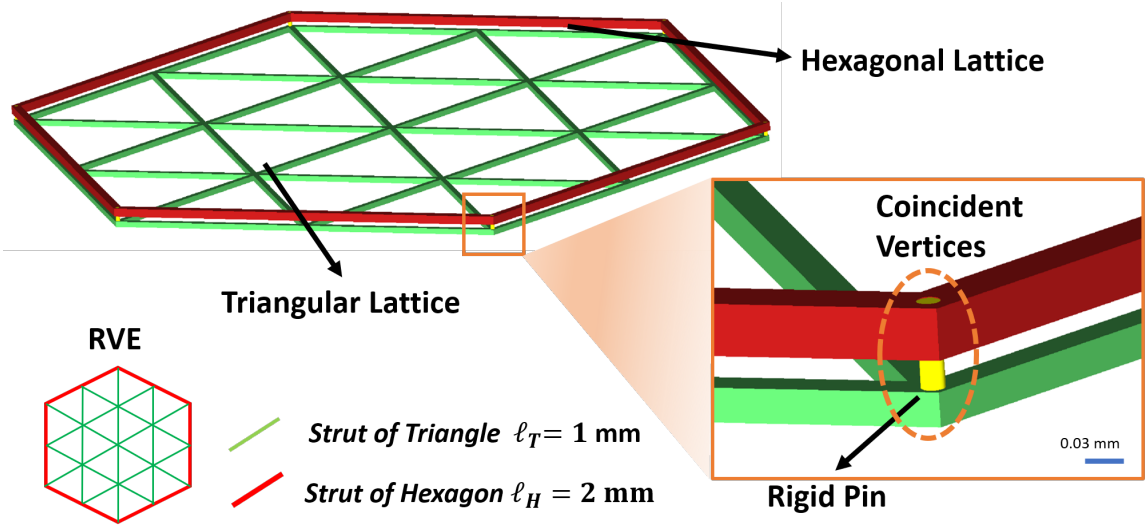


Figure 4.1: RVE of the UCM and its CAD draft drawn in SolidWorks 2019. A coincidence of two lattices can be connected via a pinned-joint mechanism.

4.3 Finite Element Modeling of UCMs

4.3.1 Definition of the Multiple Cells Model for UCMs

As discussed in Chapter 2, the multi-cell lattice model is employed to avoid unfeasible periodic damage distribution and to capture the correct fracture behavior of a full scale lattice structure. The combination of triangular and hexagonal honeycombs is achieved by building constraint equations between coincident vertices of triangular and hexagonal lattices through the following relation

$$\begin{cases} U_{T_x} - U_{H_x} = 0, & \text{in x-direction;} \\ U_{T_y} - U_{H_y} = 0, & \text{in y-direction.} \end{cases} \tag{4.1}$$

By doing so, a strut of the triangular lattice is expected to interact with all struts of the hexagonal lattice meeting at the same vertex in Abaqus model. Only the translation in two orthogonal directions is fixed, because the vertices should be free to rotate, like other non-constrained vertices. Theses vertices can act as plastic hinges, so that the struts of hexagonal lattice are able to have bending deformation. Unlike snapping two vertices together, where the hexagonal lattice is unable to perform bending dominant deformation, the pinned-joint connection can transmit internal force while maintaining the deformation type of different lattices. The procedures for constraint equations generation in Abaqus are typed in pseudo-code in Figure 4.2.

As described in previous chapters, each strut is meshed into Timoshenko beam elements (of type B21 in Abaqus element code) to allow for both strut stretching and bending [19]. For the sake of simplicity, the type of cross-section will not change the conclusion of this work. Therefore, the beam cross-section is square-shaped and constant. All lattices have relative density no bigger than 0.1 to make sure its stockiness t/ℓ is smaller than 0.2. By doing so, the following simplifying assumptions made in [84] are used: (1) constant cross-section for each beam element, which is a default option give by Abaqus (2) the dimension of the cross-section is small compared to the beam length (3) the cross-section remains planar and its shape and size does not change under loading, but can be non-orthogonal to the beam axis. For all models provided in this chapter, the constituent material of lattice is defined as a quasi-brittle material. Since the fracture response of quasi-brittle material is an in-continuous and non-linear problem, the Abaqus/Explicit solver is used for all simulation. The material properties are implemented via the VUMAT subroutine introduced in Chapter 3 with script attached in Appendix B.

Algorithm 3: Algorithm of constraint equations generation in Abaqus

Input : collection of vertices of certain lattice: BaseV and PlugV

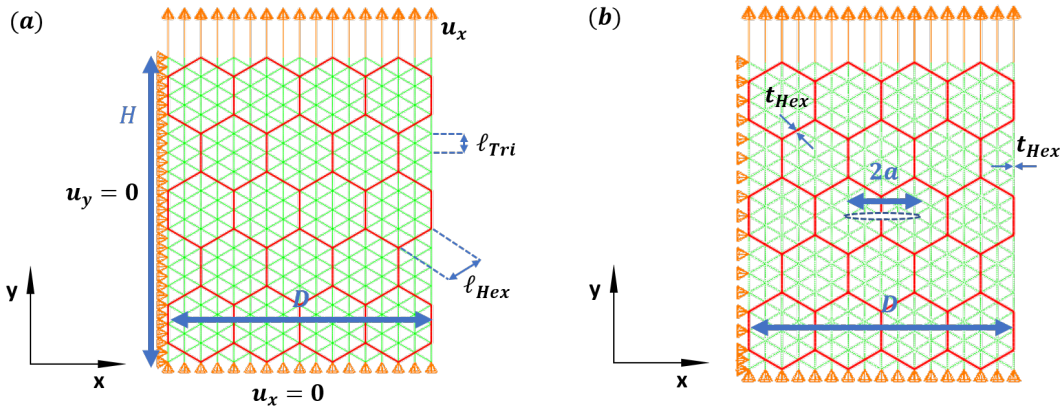
Output: constraint equations

```
# collection of vertices of 1st lattice
BaseV=assembly.instances[BaseName].sets[BaseName+'AllVertices'].vertices
# collection of vertices of 2nd lattice
PlugV=assembly.instances[PlugName].sets[PlugName+'AllVertices'].vertices
for j in PlugV do
    n = n+1 # label for a set of matching vertices
    # build and label a set for vertex-j in 2nd lattice
    assembly.Set(vertices=vertex-j.findAt(coordinate of vertex-j),
     name=PlugName+str(n))
    # build and label a set for a matching vertex in 1st lattice
    assembly.Set(vertices=BaseV.findAt(BaseV.getClosest(coordinates of vertex-j),
     name=BaseName+str(n))
end
D2 = [1, 2] # two dimension case
for k in range(1,1+n) do
    for d in D2 do
        # build an equation for each pair of matching vertices
        mdb.models['Model-1'].Equation(name, terms=((-1.0, PlugName+str(k), d),
         (1.0, BaseName+str(k), d)))
    end
end
```

Figure 4.2: The Python script for constraint equations generation in Abaqus is typed in pseudo code

4.3.2 Boundary Conditions for Tensile Test

In order to estimate fracture behavior of UCMs, the FE model is loaded with a uniform uniaxial tensile stress. Since the model is meshed by elements of B21, Abaqus/Explicit package is implemented for a quasi-static tensile test. The boundary conditions are applied on the vertices of the plate for the tensile test as illustrated in Figure 4.3. The lattice plate has no open cell or incomplete cell on its two sides in order to reduce the possibility of free surface effects. Again, the vertices at the boundary have a degree of freedom in rotation to allow for bar bending. The tensile test is conducted under displacement control using a smooth amplitude option to avoid a sudden jump in the Abaqus/Explicit solver.



$D = 13.86$ mm $\quad \bar{\rho}_{Tri} = 0.1 \quad t_{Tri} = b_{Tri} = 0.02887$mm $\quad \ell_{Hex} = 2 \cdot \ell_{Tri} = 2$ mm

$H = 16.0$ mm $\quad \bar{\rho}_{Hex} = 0.025 \quad t_{Hex} = b_{Hex} = 0.04330$ mm $\quad 2a = 0.3D$

Figure 4.3: Examples of configuration of UCM plate and its boundary conditions for tensile testing: (a) non-central cracked plate (NCP) and (b) central cracked plate (CCP). b is the out-of-plane thickness of strut and t is the in-plane thickness.

4.4 Geometry Generation in Python Script

In order to generate geometry for UCMs in different scaled sizes, another Python script is written to automatically and efficiently generate a larger structure. The process is shown in Figure 4.4.

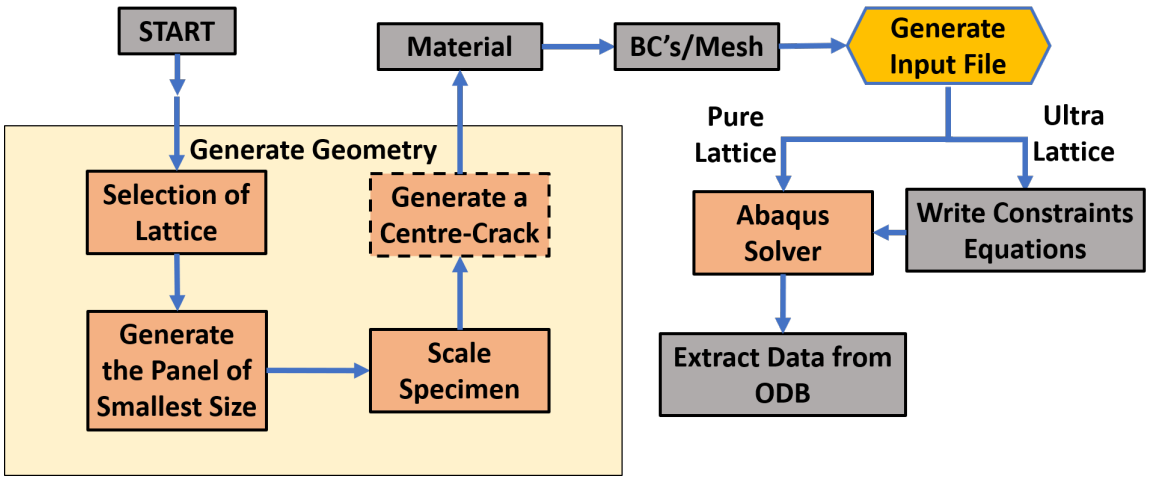


Figure 4.4: Schematic for the pre-processing of modeling lattice structures using Python script

The operation starts from the generation of the smallest panel composed of the selected type of lattice. This plate is duplicated, arranged, and merged to form another larger panel of the desired size. The detail is elaborate via the pseudo code in Figure 4.5.

Algorithm 4: Algorithm of scaling size of specimen in Abaqus

Input : lattice, scale and unit cell dimension (ℓ, h)

Output: A scaled honeycomb panel

```
instancesList = [ ] # An empty space to store new unit panel
# When scale = 1, no need to generate new unit panels
if scale == 1 then
    assembly.Instance(dependent=ON, name=lattice.name,
     part=model.parts[lattice])
    instancesList.append(assembly.instances[lattice])
end
# Clone certain number of unit panel
for row in range(0,scale) do
    for column in range(0,scale) do
        # Generate instance for each unit panel Name =
         lattice.name+str(row)+str(column)
        assembly.Instance(dependent=ON, name=Name, part=model.parts[lattice])
        # translate unit panel to certain location distx = ℓ * row
        disty = h*column
        if distx+disty   0 then
            assembly.instances[Name].translate((dist_x, dist_y, 0))
        end
        # Collect new unit panels
        instancesList.append(assembly.instances[Name])
    end
end
# Assemble all replicated unit panels to create new big panel
assembly.PartFromBooleanMerge(name=Name, instances=instancesList)
print 'Finished Merging'
# delete all duplicated unit panels
for i in assembly.features.keys() do
    assembly.deleteFeatures(featureNames=(i,))
end
```

Figure 4.5: The Python script for scaling up geometry is typed in pseudo code

The center-crack is generated later, as needed. The method for crack generation is explained in Figure 4.6.

Algorithm 5: Algorithm of crack generation for modeling lattice structure in Abaqus

Input : lattice, location of crack center (x,y) and crack dimension (a,b)

Output: A crack

```
# Sketch a geometry of crack
model.ConstrainedSketch(nam='ellipse', sheetSize)
model.sketches['ellipse'].EllipseByCenterPerimeter(axisPoint1=(x+a, y),
 axisPoint2=(x, y+b), center=(x, y))
mdb.models['Model-1'].Part(dimensionality=TWO-D-PLANAR, name='Crack',
 type=DEFORMABLE-BODY)
mdb.models['Model-1'].parts['Crack'].BaseShell(sketch=mdb.models['Model-
 1'].sketches['ellipse'])
del model.sketches['ellipse']
# Generate new instances
assembly.Instance(dependent=ON, name='Crack', part=model.parts['Crack'])
assembly.Instance(dependent=ON, name=lattice.name, part=model.parts[lattice])
# Generate a new plot with crack via PartFromBooleanCut function in Abaqus
assembly.PartFromBooleanCut(name=lattice.name,
 instanceToBeCut=assembly.instances[lattice],
 cuttingInstances=[assembly.instances['Crack']])
print 'Finished Cracking'
```

Figure 4.6: The Python script for central crack generation is typed in pseudo code

The material properties are taken into account as the whole part is meshed. Boundary conditions are applied accordingly, and the model is exported for Abaqus Explicit solver. These pre-processes are detailed in Figure 4.7.

Algorithm 6: Algorithm of pre-processing for modeling lattice structure in Abaqus

Input : information of lattice and material

Output: Input file

```
# Generate the panel of the smallest size
Implement Algorithm 2 to generate multi-cell model
# Scale Specimen
Implement Algorithm 4 to generate a larger pane of desired size
# Generate a crack (Optional)
Implement Algorithm 5 to generate a crack in the geometry
# Apply material definition (user defined material)
mat1 = [E, Ys, Gf, nu] # material properties
model.Material(name='MAT1')
model.materials['MAT1'].Density()
model.materials['MAT1'].Depvar(deleteVar=state variable controlling element
  deletion flag, n=number of state variables)
model.materials['MAT1'].UserMaterial(mechanicalConstants=tuple(mat1))
# Create and assign Section
  # Create profile for Beam section
  # Assign Orientation (for beam element)
# Generate Instances
# Create explicit step
# Set Amplitudes for Explicit solver
# Apply BC's
  # Set Initial BC's
  # Set loading BC's
# Mesh
  # Select element tyoe
  # Decide mesh density
model.parts[lattice].generateMesh()
# Create job
# Generate input file
mdb.jobs[na].writeInput(consistencyChecking=OFF)
```

Figure 4.7: The Python script for pre-processing in Abaqus is typed in pseudo code

If the model is for UCMs, the constraint equations are written into the input file via another Python script outside the Abaqus environment (Figure 4.8). This is done rather

than creating the equations in the Abaqus Python module (Figure 4.2). The Abaqus Python module provides an embedded function, 'getClosest', to find the closest vertex entity via the given coordinates. Although this is a convenient way to find a pair of coincident vertices, it becomes inefficient as the total number of vertices increases.

Algorithm 7: Algorithm of generation of constraint equations outside Abaqus environment

```
Input  : Input file
Output: Input file containing constraint equations
import shutil
import re
dir=name of input file
tempDir= dir+".txt"
# Copy paste content from original file to temp file
shutil.copy(inpDir+'.inp',tempInpDir)
# Rewrite inputfile
# strings to house all lines, label of matching vertices and constraint equations
wstr = ""; wset1 = ""; wset2 = ""; weq = ""
hex = [ ]; tri = [ ]; hvex = [ ]; tvex = [ ] # lists to house vertices
with open(tempInpDir,'r') as r:
lines=r.readlines()
# collect information of vertices in each lattice
for  i in range(len(lines)) do
    # collect all vertices
    if re.search(keyword, line[i]) then
        hvex = [int(num) for num in lines[i+1].split(',')]
    end
    if re.search(keyword, line[i]) then
        for line in lines[i+2:] do
            hex.append(data of vertex)
        end
    end
    ...(same for Triangular lattice)
end
```

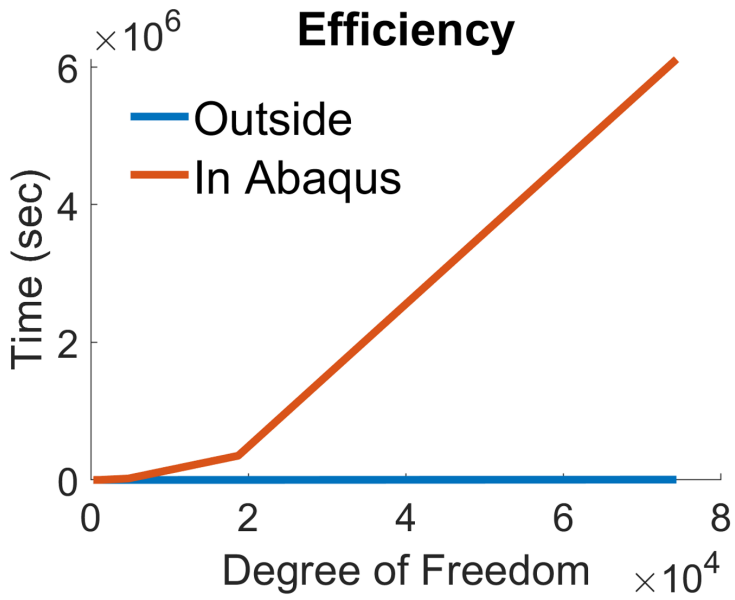
```
(...continue)
# find vertices which share the same coordinates
for j in range(hvex[1]+1) do
    for k in range(tvex[1]+1) do
        if ‖j − k‖ < 0.001 then
            n=n+1
            # Generate set of matching vertices for constraint Equations
            wset1 = wset1+'*Nset, nset=Hexagon'+str(n)+', instance=Hexagon'+'
             '+str(hex[j][0])+','
            ...(Similar for Tri)
            # Create line for constraint equations
            weq = weq+'** Constraint:
             Const-'+str(n)+'-1'+'*Equation'+'2'+'Hexagon'+str(n)+', 1,
             -1.'+'Triangle'+str(n)+', 1, 1.'
            ...(same for y-direct)
        end
    end
end
r.close()
# Integrate all lines in temp file
with open(tempInpDir,'r') as f:
for line in f do
    if re.search("End Assembly", line) then
        line=re.sub("End Assembly",wset1+wset2+weq+'*End Assembly',line)
        wstr+=line
    end
    else
        wstr+=line
    end
end
```

```
(...continue)
# copy paste to final file
with open(tempInpDir, "w") as w:
   w.write (wstr)
   shutil.copy(tempInpDir,golDir)
os.remove(tempInpDir)
```

Figure 4.8: The Python script for constraint equations generation outside Abaqus environment is typed in pseudo code

Figure 4.9 shows an impressive advantage of using outside environment processing when the size of the geometry increases. The cost of time when using in-Abaqus modulus increases

exponentially with the total number of vertices and grows to more than 2 months for the largest specimen investigated in this work. On the contrary, the implementation of outside environment processing requires significant less time (less than 90 minutes) even for the largest size of geometry. The full script for writing constraint equations outside Abaqus environment is attached in the Appendix C. The other script following similar steps but specialized for generating large scale UCMs is attached in Appendix D as well.



Scale	DOF	# of Constraint Equations
1	336	112
2	1272	424
4	4752	1584
8	6240	6240
16	24778	24778

DOF = (# of vertices) x (3)

Figure 4.9: Efficiency comparison for the generation of constraint equations between the in-Abaqus modulus and outside environment processing. The sample of geometry is the NCP of UCMs. All testing works and code development were run in the Window Server 2019 Standard with 64 bit OS using x64 processor, Intel(R) Xeon(R) CPU E5-2695 v3 2.30 GHz, along with 16.0 GB installed RAM.

4.5 Convergence tests and Validation of UCMs and Pure Lattice Model

The UCMs models could involve many constraint equations especially for a UCMs specimen of large size. This can cause a difficulty for parallel computing in the Abaqus solver. To ensure the efficiency in running simulations, it is necessary to find a desired minimum number of elements in the FE models. Accordingly, a convergence study is conducted to minimize the element size required to ensure the model convergence for the sake of preci-

sion. From the convergence study on the central cracked panel of the UCMs and non-cracked panel of the triangular lattice (Figure 4.10), it is clear that the results for different numbers of elements converge quickly when each strut is meshed into 10 to 50 elements. This is due to the fact that the modified crack bend model used in the subroutine is insensitive to the element length. The maximum deviation in peak force between these two curves is within 0.1% for the triangular lattice. Therefore, for all the following provided simulations, each strut in the model was meshed into up to 10 beam elements.

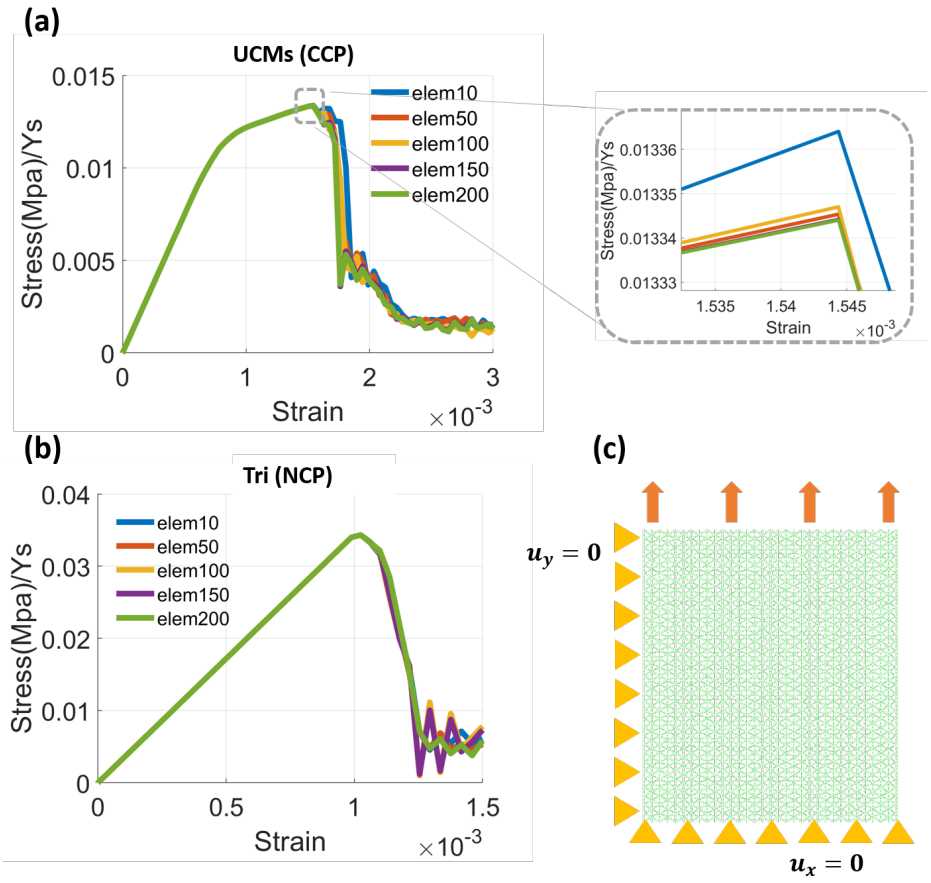


Figure 4.10: Convergence study results of (a) central cracked panel of UCMs and (b) triangular lattice at the width D of 13.86 mm; (c) Boundary conditions of triangular lattice is similar to that of UCMs.

In addition to the model verification, in order to ensure the accuracy of simulation model,

another validation, for the model at the given mesh density, is presented. The stress strain curves in Figure 4.11 also show that the numerical results (solid line) of both the triangular and the hexagonal honeycomb show a good agreement with the theoretical prediction (dash line) provided in Chapter 2. The boundary conditions of these two models are shown as Figure 4.12. The errors in stiffness and in strength are within 3% and 5% respectively at the provided mesh size. This provides an accuracy of numerical solution acquired from the FE models at the given mesh refinement.

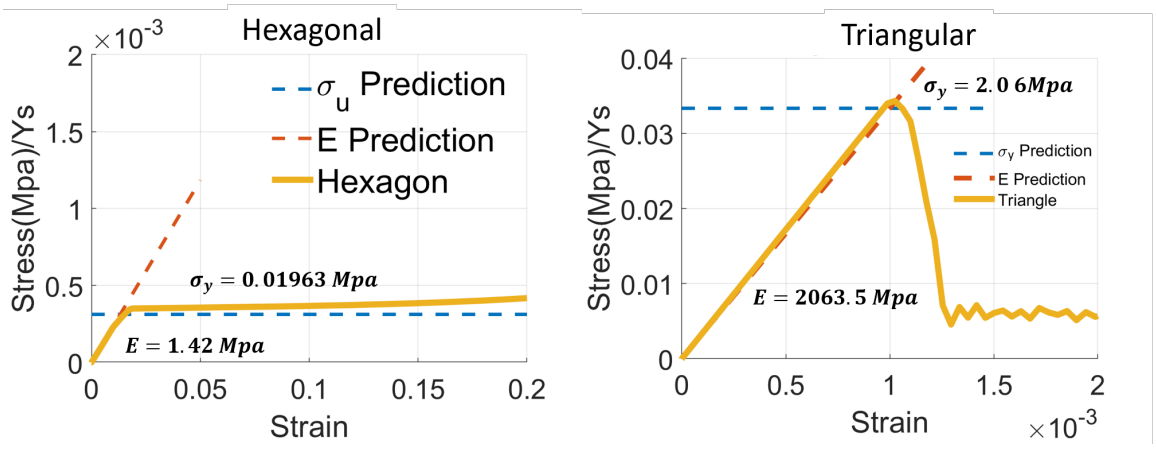


Figure 4.11: Verification for pure lattice plotted in nominal stress versus nominal strain with analytical predictions calculated by Eq. (2.7) (2.8) and (2.9) in Chapter 2. The triangular and the hexagonal lattices have relative density of 0.1 and 0.025 respectively. The constituent material of lattices has Young's Modulus of 60, 000 MPa and fracture strength of 60 MPa.

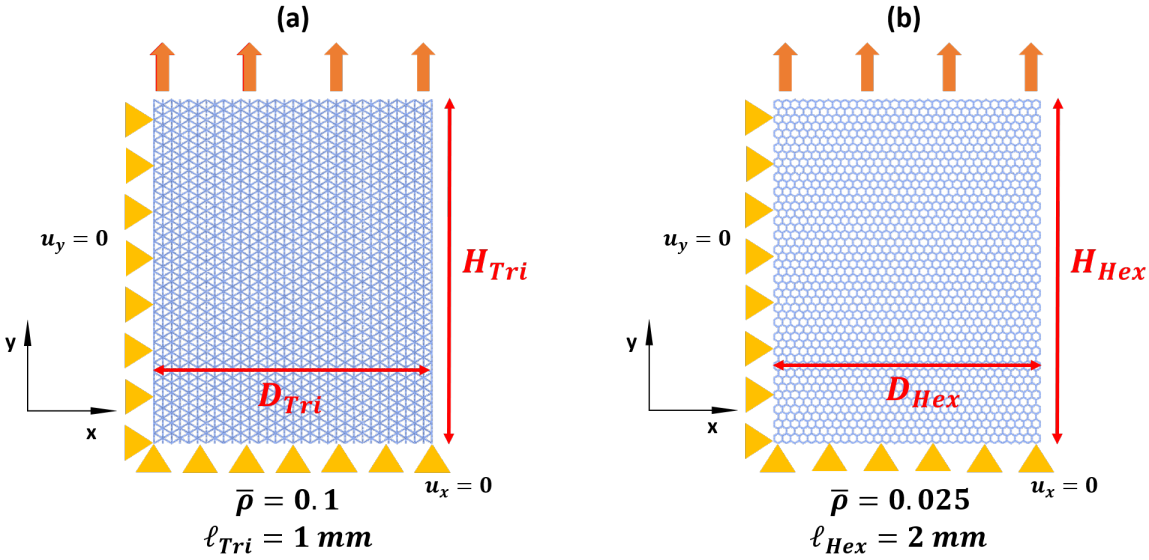


Figure 4.12: Configurations and boundary conditions for (a) triangular lattice model composed of 2100+ cells and (b) Hexagonal lattice model consisting of 1500+ cells. $D_{Tri} = 27.71$ mm, $D_{Hex} = 110.85$ mm, $H_{Tri} = 34$ and $H_{Hex} = 142$ mm.

4.6 Parametric Study of UCMs Model

4.6.1 Use of Design of Experiment (DOE)

With the robust validation and confirmation in convergence, the parametric study is conducted to understand the tensile response of UCMs, especially their damage tolerance. This ability can be expressed in forms of toughness, fracture energy or the hardening behavior. To acquire and evaluate the damage tolerance performance in UCM, a conceptual approach of design of experiment (DOE) is applied to optimize the performance of UCM. Some assumptions are made to simplify the model. DOE is a robust statistical tool for developing an objective function of input parameters. Its validation is elaborated and proved in [88]. The DOE uses a fractional factorial experiment instead of a full factorial experiment in order to reduce the number of experiments required. By applying the DOE, the total number of simulations can be reduced during the parametric study. The procedures are illustrated below.

The objective function here is the performance of the UCMs. Since the behavior of lattice

is influenced by topology, relative density (strut dimension) and constituent materials as mentioned in Chapter 2, the input parameters may include properties of cell-wall material, relative density, combination, types of lattice, use of PCB or Multi-cell model and weight ratio of constituent cellular materials. The noises can come from the mesh density of FE model and the entire size of specimen. To simplify the model, the following assumptions are made in this study: (1) the properties of constituent material remain constant to ensure the effect of structural design alone; (2) only the regular triangular and hexagonal lattices are considered and their combination structure is fixed due to the implementation of multiple length scales and the pre-selected types of lattices; (3) the type of elements are pre-selected and the cross-section of each strut is square; (4) only the multi-cell model is used to acquire the reasonable fracture behavior; (5) only one size of the entire structure is considered here because this parametric study focuses on the design of the UCM rather than the volume of lattice is used. In fact, the size effect in UCMs caused by quasi-brittle materials is anticipated and is discussed afterward. Therefore, the only controllable parameters are the relative density ratio of two lattices and the weight fraction of hexagonal lattice. However, from the Eq. (2.3) and (2.4), the density of lattice can be expressed as:

$$\rho^* = \frac{1}{V^*} W_s \tag{4.2}$$

The relative density ratio of two lattices, $\rho_{T/H}$, becomes:

$$\rho_{T/H} = \frac{\rho_T^*}{\rho_H^*} = \frac{V_H^*}{V_H^*} \frac{W_{Tri}}{W_{Hex}} \tag{4.3}$$

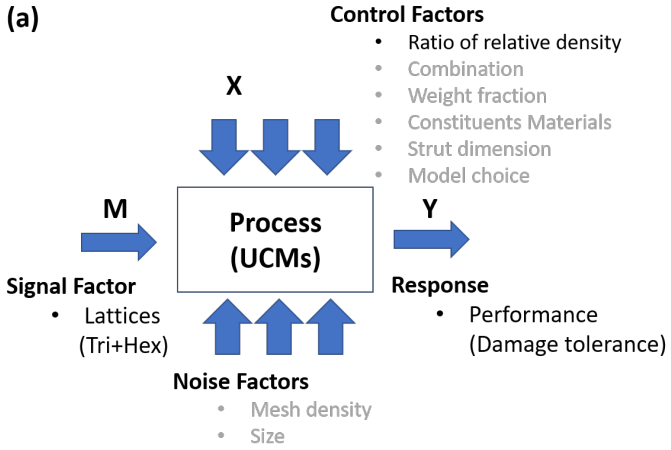
where W_{Tri} and W_{Hex} represent the weight of triangular and hexagonal lattice respectively. Replacing V^* by $H \times D \times b$, the Eq. (4.3) turn into:

$$\rho_{T/H} = \frac{H_{Hex} D_{Hex} b_H}{H_{Tri} D_{Tri} b_T} \frac{W_{Tri}}{W_{Hex}} \tag{4.4}$$

Since the cross-section of struts are fixed and the topology area of two lattices is the same, the ratio of out-of-plane thickness b is a constant. Because of this, $\rho_{T/H}$ becomes a function of the weight fraction of the hexagonal lattice .

$$\rho_{T/H} = f(\frac{W_{Hex}}{W_{Total}}) \tag{4.5}$$

That is the relative density ratio is the only variable considered in this parametric study. Knowing that non-linear relationship between relative density of hexagonal lattice and its performance, the UCMs are likely to have a non-linear response to the relative density ratio. As such, the ratio of relative density of constituent lattices (triangular and hexagonal) is taken as a 3-level parameter. The pure triangular lattice is the control group to estimate the influence of the UCM. The aforementioned process are illustrated in Figure 4.13. The gray color means that the factor no longer become a variable to the objective function.



(b)

Factor	level	L1	L2	L3
Model in use	2	PBC	Multi	x
length scale ($^{\ell_H}/_{\ell_T}$)	3	2	1	0.5
combination	3	I	II	III
cell-wall material (G_f)	3	G_{f1}	G_{f2}	G_{f2}
cell-wall material (E)	3	E_1	E_2	E_3
cell-wall material (σ_f)	3	σ_{f1}	σ_{f2}	σ_{f3}
strut cross section ($^t/_b$)	3	2	1	0.5
ratio of relative density ($^{\rho_H}/_{\rho_T}$)	3	1	4	10
weight fraction ($^{w_H}/_{W_{Tot}}$)	3	90%	40%	10%

Assumptions

Factor	level	L1	L2	L3
Model in use	1	~~PBC~~	Multi	x
length scale ($^{\ell_H}/_{\ell_T}$)	1	2	~~1~~	~~0.5~~
combination	1	I	~~II~~	~~III~~
cell-wall material (G_f)	1	G_{f1}	~~G_{f2}~~	~~G_{f2}~~
cell-wall material (E)	1	E_1	~~E_2~~	~~E_3~~
cell-wall material (σ_f)	1	σ_{f1}	~~σ_{f2}~~	~~σ_{f3}~~
strut cross section ($^t/_b$)	1	~~2~~	1	~~0.5~~
ratio of relative density ($^{\rho_H}/_{\rho_T}$)	3	1	4	10
weight fraction ($^{w_H}/_{W_{Tot}}$)	~~3~~	*Depend on ratio of relative density*		

Figure 4.13: illustration of implementation of Design of Experiment (DOE); (a) a objective function box chart showing all variables that may affect the output performance; (b) The level reduction for the experiment.

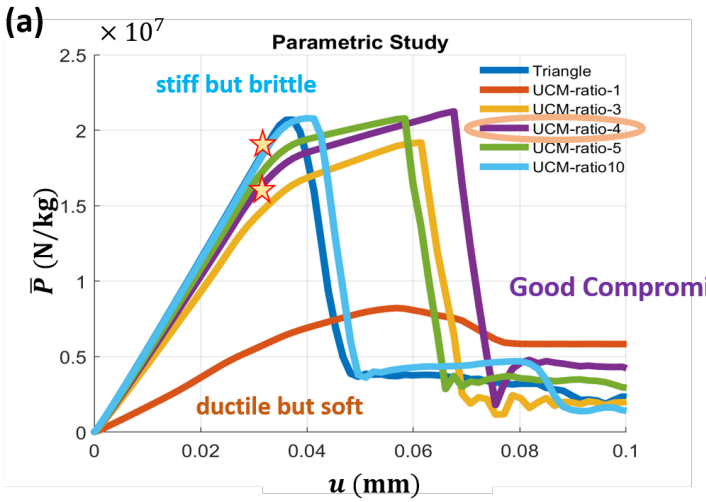
4.6.2 Analysis and Results of Parametric Study

This parametric study is executed via multi-cell model with the mesh density of 10 B21 elements per strut. The result is shown in Figure 4.14. All specimen has width of 27.712814331 (mm), denoted as D_2. The measured values of macroscopic specific force $\bar{P}$ are plotted as the function of displacement u. This specific force $\bar{P}$ is obtained in terms of total reaction force P_{sum} on the top nodes divided by the total mass M of the specimen as $\bar{P} = P_{sum}/M$. For convenience, variant ratio of relative density with corresponding weight fraction is organized in Table 4.1. The specific force-displacement curve in Figure 4.14(a), shows that if the fraction of the weight of the hexagonal lattice is 5.87%(ratio-10), the UCM has the highest stiffness among other curves, but it lacks an ability to absorb energy. On the other hand, when the UCM is made up of 86.2% hexagonal honeycomb (ratio-1), it exhibits a good ductile performance, but it sacrifices more than 50% of its stiffness and strength. When the hexagonal lattice accounts for around 28% of the UCM (ratio-4), there is a 50% drop in stiffness. And yet it displays an obvious hardening response and outstanding compliance with almost no penalty in specific peak load.

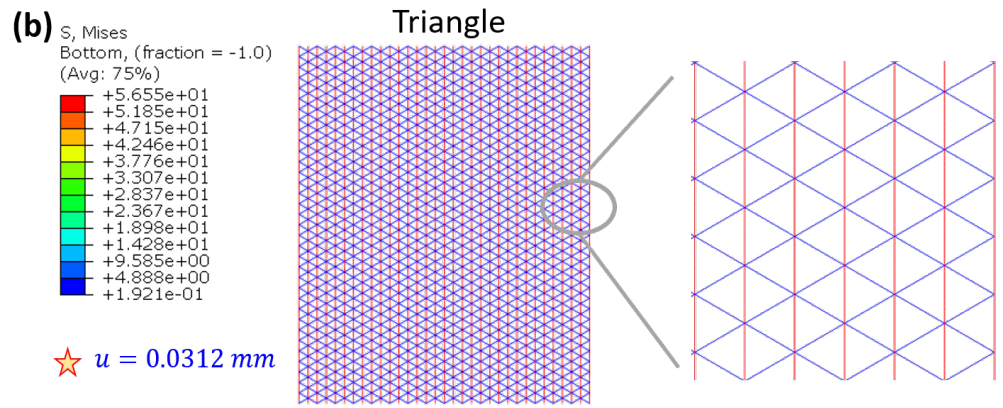
However, the first stage of parametric study of the relative density ratio of triangular and hexagonal lattice showed that the best compromise occurs when the weight fraction of hexagonal lattice is around 28%. This is due to the load distribution caused by the hexagonal lattice as shown in Figure 4.14(b) and (c). It is clear that the hexagonal lattice can help distribute load in some struts of the triangular lattice (dash line red square). These struts align along the load direction and are overlapped by the hexagonal lattice. Although the local fracture takes place earlier in the UCMs at the same extension, the hexagonal lattice in the UCMs continues to carry the load, leading to a hardening behavior in the UCMs. This hardening behavior increases the elongation of the triangular lattice at the same time.

With the fact that ratio-4 has a desired response compared to other cases, extra two values (ratio-3 and ratio-5) are tested to find the potential candidate for the UCM configuration to optimize the damage resistance and fracture toughness. Their results are plotted in Figure 4.14(a) as well. Both the ratio-3 and ratio-5 cases have a lower specific peak force

and fail earlier than the ratio-4 case. In summary, the UCM of ratio-4 has the best response for damage tolerance compared to other types of combinations.



$\frac{\bar{\rho}_T}{\bar{\rho}_H}$		$\frac{W_H}{W_{Total}}$
ratio	**1**	86.2%
	3	41%
	4	28%
	5	20%
	10	5.87%



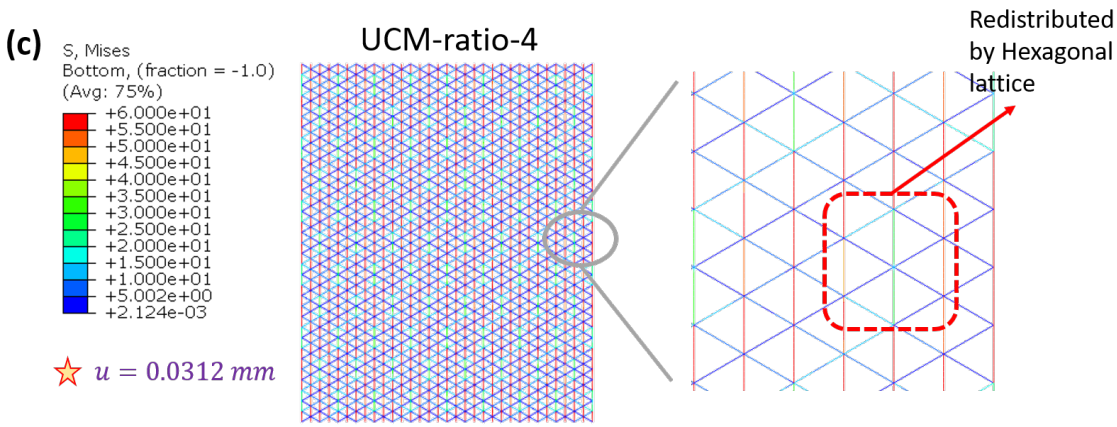


Figure 4.14: Parametric study on the ratio of relative density to find the optimal configuration of UCM: (a) the specific peak load versus extension plot showing different performance among various combination; (b) the contour plot of the triangular lattice; (c) the contour plot of the UCM where its component of hexagonal lattice was eithered for the ease of visualization.

$\bar{\rho}_{Tri}/\bar{\rho}_{Hex}$	ratio-1	ratio-3	ratio-4	ratio-5	ratio-10
W_{Hex}/W_{Total}	86.2%	41%	28%	20%	5.87%

Table 4.1: Ratio of relative density and the corresponding weight fraction for Figure 4.x(a).

4.7 Toughness Evaluation and Comparison to Pure Lattices

With the decision of desirable configuration for the UCMs, the toughness of certain UCMs is evaluated to quantified how much damage tolerance has been improved. Since the toughness represents an ability of materials to deform plastically and to dissipate energy without fracturing, the damage tolerance can be estimated in a form of toughness.

To evaluate the damage tolerance improvement, the toughness of cellular materials is examined via the simulations of unloading test. The results are plotted in specific force-displacement responses in Figure 4.15 and 4.16. The green dash line in Figure 4.15 represents the unloading path of the UCM and the area of red shadow is calculated as the specific toughness of the UCM. However, under the same elongation, the pure triangular lattice does not have much toughness, but nor does the hexagonal lattice demonstrate its toughness. In fact, hexagonal lattice does exhibit great toughness with elongation when it is fully stretched (Figure 4.16). The hexagonal is stretched and shrinks tremendously in lateral direction under load and recover back to original length. The resulting implication is that the increase in energy absorption is transferred from the hexagonal lattice.

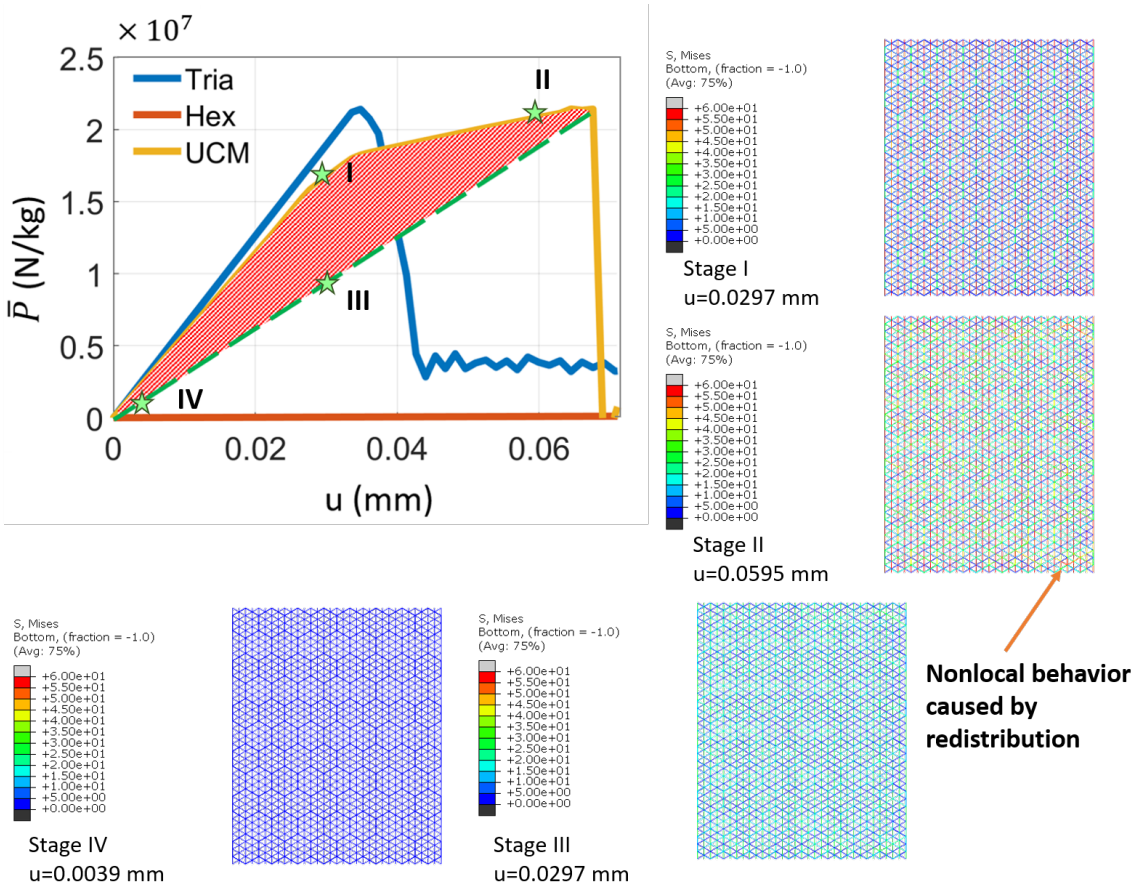


Figure 4.15: Specific force-displacement responses for the UCM, hexagonal and triangular lattice structures under tensile loading. Highlighted area implies the large toughness obtained in the UCM compared to base lattice (triangular).

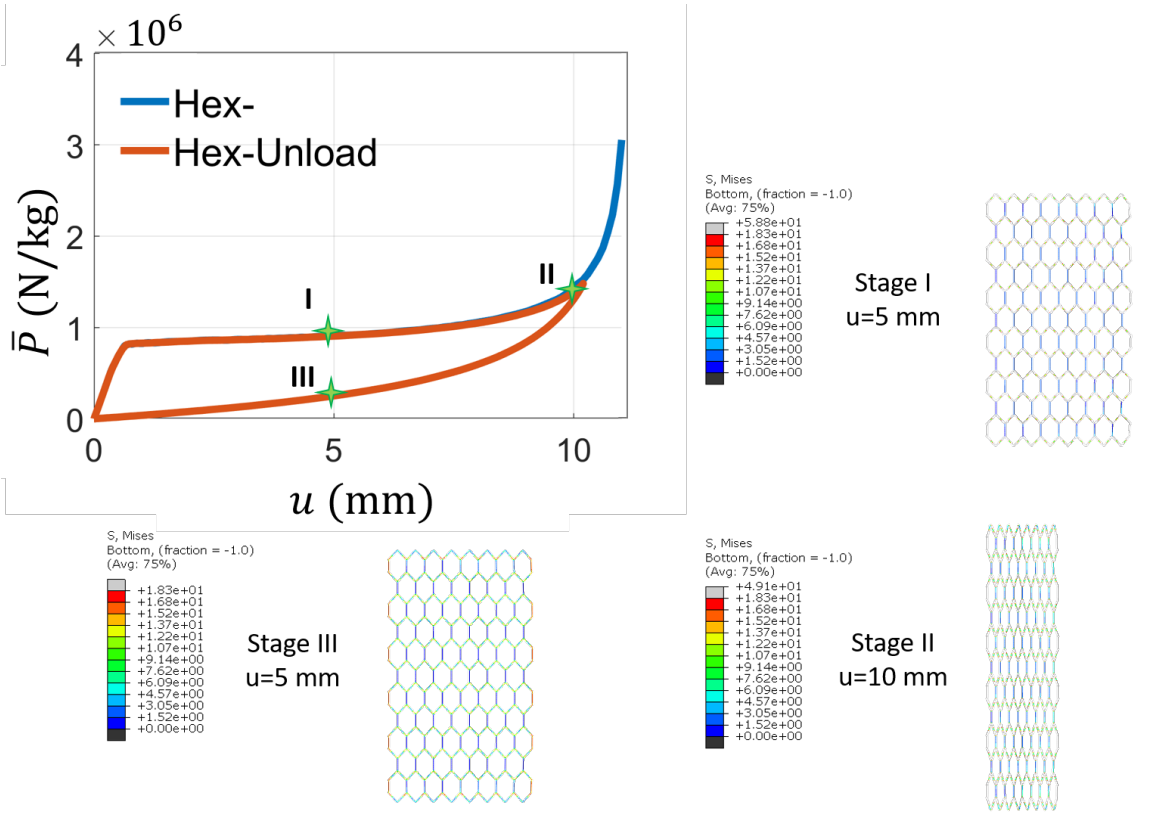


Figure 4.16: Unloading curve for the hexagonal honeycomb plotted as specific force- displacement response.

The evaluation is summarized in Figure 4.17, showing the comparison between the triangular lattice and UCM. Since triangular lattice performs brittleness and good strength, it will be impressive if it can be toughened without much penalty in strength and stiffness. Usually there is a trade off between strength and elongation for materials. In contrast, hexagonal lattice is really ductile and possess high toughness, so it is not quite necessary to increase its toughness.

Moreover, since the triangular lattice and UCM have different cross-section and mass, the toughness should be divided by the total mass for the sake of fair comparison. Accordingly, the specific peak force is considered to estimate the penalty in strength at the same time. In Figure 4.17, the specific toughness of UCM is about 22 times larger than the triangular lattice with almost 0% of drop in specific peak force. This is caused by the non-local behavior

introduced by the UCM, which helps improve the damage tolerance. To understand the mechanism of this, the load-unload curve in Figure 4.15 is characterized into 4 stages. In stage I, the hexagonal lattice helps some struts in the triangular lattice carry the load, leading to a hardening behavior as explained in Figure 4.14(c). In stage II, the hexagonal lattice in the UCM continues to redistribute the force around the struts which have reached to the ultimate strength of the cell-wall material. This prevents a failure in triangular lattice as shown in Figure 4.15. Due to this behavior, the rest of struts of triangular lattice can continue to carry the force up to the rapture limit during the hardening behavior. In stage III and IV, all struts are recovering back to unstretched status. As shown in stage IV, there is little residual stress remaining in the structure.

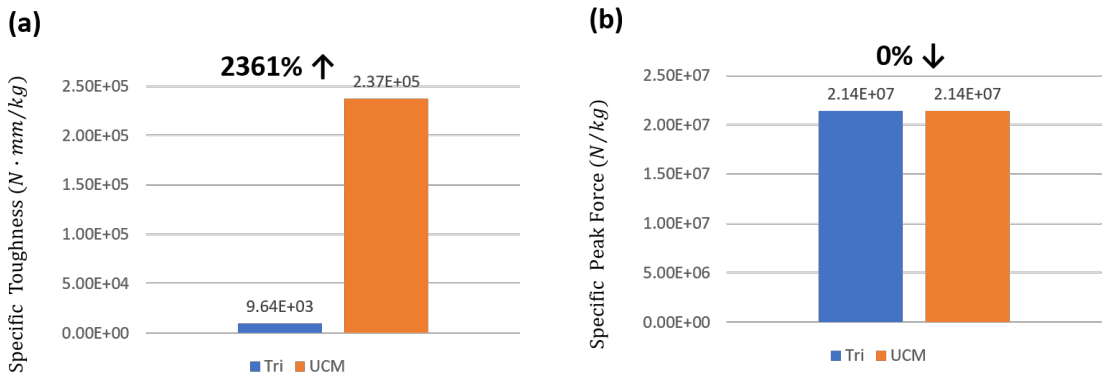


Figure 4.17: The comparison of mechanical properties for the triangular lattice and for the UCM at size of Scale-2: (a) Specific toughness and (b) Specific Peak Force.

4.8 Size Effects Other Than Type II Size Effect

4.8.1 Size Effect on Boundary Layer

To have a full comparison between the performance of the UCMs and pure lattices, some size-dependence behaviors have also been observed in cellular materials and are discussed accordingly. In the multi-cell lattice model, the cells on the boundary are not surrounded by identical cells on all sides, which can affect the calculated mechanical properties from the finite element method. This special condition is different from PBC which has no free-edge

effect. This kind of effect can soften or stiffen the cellular structure and is dependent on the size of entire structure. The result of such a size effect on the stiffness is shown in Figure 4.18. The microscopic stiffness of finite lattice is normalized by the Young's modulus of cell-wall material E_s. It is plotted versus β, which is the ratio of specimen size D to unit cell size ℓ. The solid line is the result calculated by the finite element model. In Figure 4.18(a), the stiffness of triangular lattice gradually decreases and approaches to analytical prediction when β increases. This phenomenon is the same as the result in [56]. Such stiffening size effect [89] is caused by two phenomena: (1) the decrease of the area fraction of those stress-free struts at the boundary and (2) the stretching deformation of the struts. However, the smallest specimen of triangular lattices has the biggest deviation of 6% in stiffness, which implies that the boundary layer has little influence on the elastic stiffness of the triangular lattice. A similar conclusion can be found in [81]. By contrast, the hexagonal lattice displays a softening size effect that the stiffness increases with β. This is caused by the decrease of the area fraction of those stress-free struts at the boundary. The inability of such free edges can not contribute stiffness to the whole structure, which leads to the softening influence. The same result is shown in [90] as well.

Additionally, it is noted that the hexagonal lattice converges faster than the triangular lattice although it has larger error (10.7%) in stiffness at the same β of lower value (≈ 20).

On the other hand, the UCM presents a stiffening size effect, but goes down to the asymptotic faster than the triangular lattice. The maximum error between these two sizes of specimen is less than 5%. This implies that the UCM is less sensitive to the size effect on stiffness caused by the boundary layer. Therefore, it is reasonable to determine the modulus value of the UCM as 1012.2 MPa based on the present simulation result of the largest specimen.

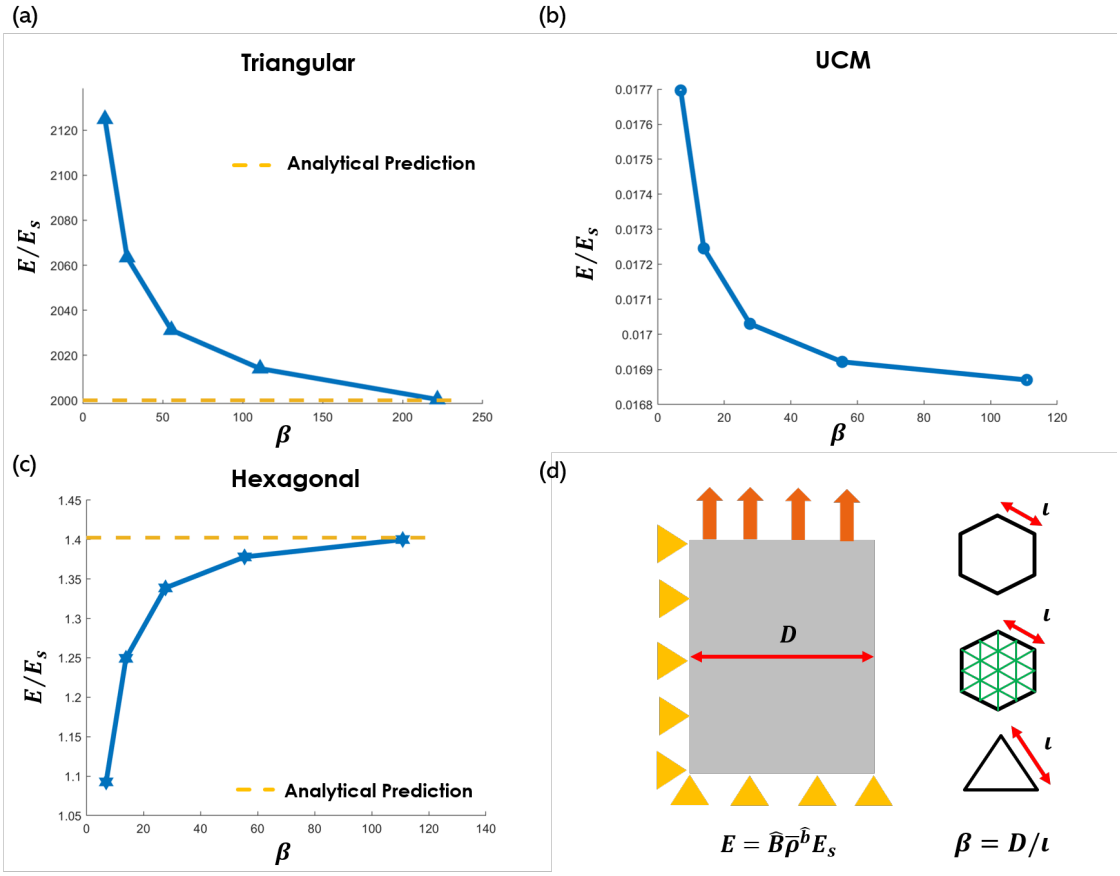


Figure 4.18: Effect of specimen size on lattice stiffness: (a) Triangular lattices (b) UCMs and (c) Hexagonal lattices. The yellow dash-line is analytical prediction of Young's Modulus calculated by Eq. (2.7)

In addition, the size effect on the ultimate strength and yield stress is summarized in Figure 4.19. It is plotted in relative strength as a function of β. The relative strength is expressed in terms of the fraction of nominal strength at peak σ_u to the ultimate strength of cell-wall material σ_{us}, as σ_u/σ_{us}). In Figure 4.19(a), the relative strength of triangular and the UCM decreases asymptotically to the analytical prediction with increasing β. However, the hexagonal lattice display an incline asymptotical trend. Therefore, the ultimate strengths of the UCM and of the triangular lattice are defined as their measured peak strengths at the largest size of specimens and noted as 1.134 and 2.01 Mpa respectively. On the other hand, the yield point of the hexagonal lattice is defined as the stress at which 0.2%

plastic deformation occurs in the stress-strain curve. The value of yield stress is measured as 0.0197 Mpa at the largest specimen size.

It is noted that the maximum deviation error of ultimate strength between these two sizes of specimen is about 5.7% and 1% for the triangular lattice and UCM respectively. This implies that the free edges has less impact on the UCM than the triangular lattice.

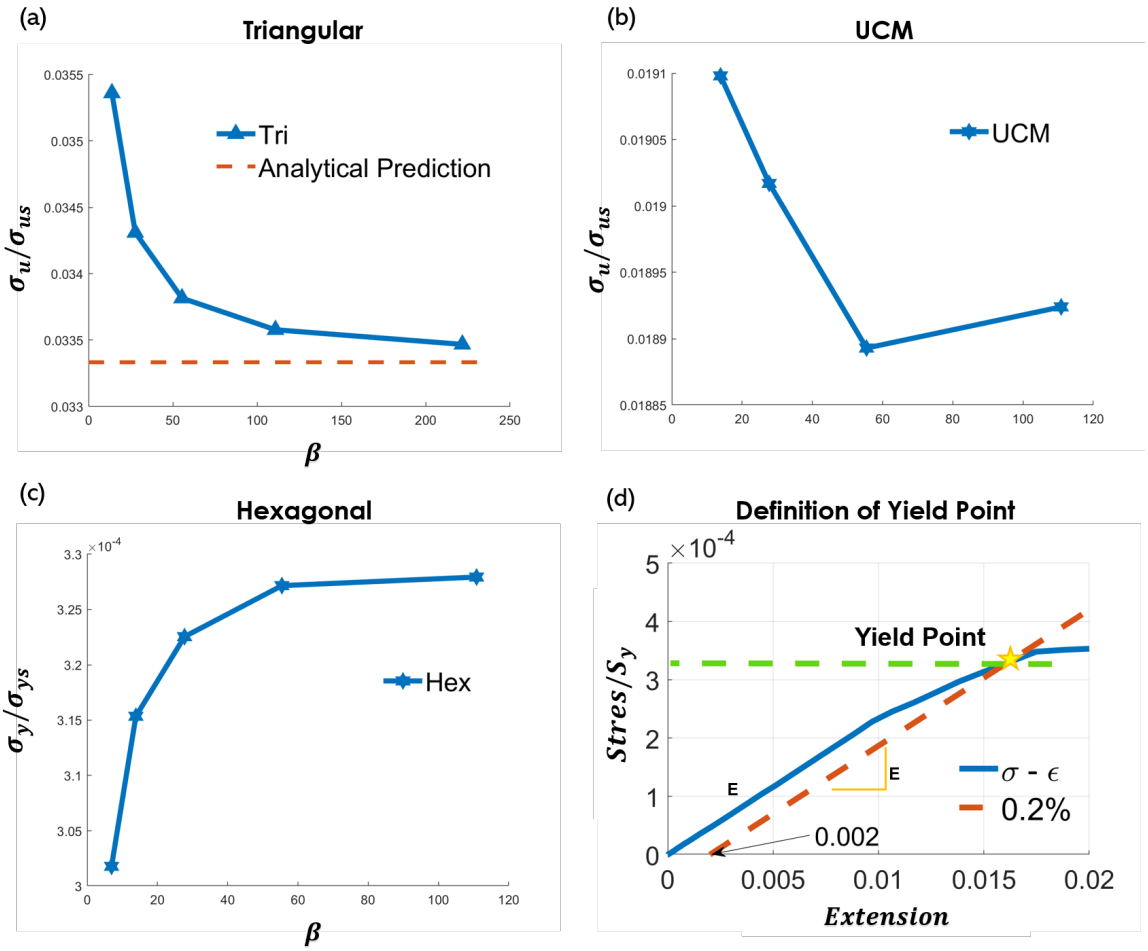


Figure 4.19: Effect of specimen size on lattice ultimate stress and yield stress: (a) Triangular lattices (b) UCMs and (c) Hexagonal lattices. The red dash-line is analytical prediction of ultimate stress calculated by Eq. (2.8)

4.9 Evaluation of Fracture Energy of UCMs

In addition to the toughness, the damage tolerance can be estimated by evaluating fracture toughness. This is an ability of materials to resist fracture which is directly related to the fracture energy. Based on the theory of fracture mechanics, cracks require energy to propagate through materials. This energy is referred as fracture energy. The tremendous energy dissipated during crack propagation displays how difficult to damage the material. As a result, the higher fracture energy of a material is, the better damage resistance this material has. The fracture energy can be evaluated through conducting a tensile test on a central cracked panel of the specimen. However, since the constituent material of UCMs has quasi-brittle performance, the size effect due to the quasi-brittle material is expected to happen in the CCP of UCMs. This size effect is called type II size effect and is caused by a fully developed FPZ as elaborated in Chapter 3. Once the region of FPZ is relatively small compared to the entire structure, the fracture behavior can be approached by LEFM. The CCP is generated as a geometry in the following models. An example of boundary conditions for CCP tensile test has been shown in Figure 4.3(b).

4.9.1 Theory of Type II Size Effect

In this section, the type II size effect is implemented to evaluate the fracture energy for the quasi-brittle lattices. The centre-cracked plate (CCP) is used accordingly. The Bažant size effect law is explained below: Recalling mode I fracture, the energy release rate G is written as

$$\mathcal{G} = \frac{\sigma_N^2}{E'} D g(\alpha) \tag{4.6}$$

, where σ_N is the applied load (nominal stress), D is the width of the specimen, $g(\alpha)$ is the dimensionless stress intensity acting as a function of relative crack length $\alpha = a/D$ with semi-crack length a, and

$$E' = E, plane stress; E' = E/(1-\nu^2), plane strain. \tag{4.7}$$

, where ν is the Poisson's ratio. Noting that $\mathcal{G}$ approximates to fracture energy G_f, gives

$$\sigma_N = \sqrt{\frac{E'G_f}{Dg(\alpha)}} \tag{4.8}$$

The cohesive crack can be approached by an equivalent LEFM so the equivalent crack can be written as $a = a_0 + c_f$ where a_0 is the original crack length and c_f is the additional crack length speaking for around half length of the FPZ. By the Taylor series, the dimensionless energy release rate transfers to

$$g(\alpha) = g_o + g_0'\theta + \frac{1}{2}g_0''\theta^2 + ... \tag{4.9}$$

, where $g_0 = g(\alpha_0)$ and $g_0' = \frac{dg(\alpha)}{d\alpha}|_{\alpha=\alpha_0}$ with $\alpha_0 = a_0/D$. Substituting g in Eq. (4.8) with the expression of Eq. (4.9) by neglecting the high order terms, gives

$$\sigma_N = \sqrt{\frac{E'G_f}{g_0'c_f + g_0D}} = f_tB\left(1 + \frac{D}{D_0}\right)^{-1/2} \tag{4.10}$$

, where $D_0 = c_f\frac{g_0'}{g_0}$ and $Bf_t = \sqrt{\frac{E'G_f}{c_fg_0'}}$. The plane stress is considered in the reported estimations due to the fact that $b << D$. For UCM, Young's modulus is determined by the specimen of the largest size (see Appendix E).

4.9.2 Scaled Size and Configuration of CCP

To scale the self-similar structure, α of 0.15 is fixed, and D is given a function of the number of cells N, $D = f(N)$. This suggests that the size of a unit cell L is not scaled with the entire structure. The effect of FPZ on the size of the entire plate of quasi-brittle materials is the main interest. Another type of size effect, that which deals with the continuum of lattice [43], is not in the scope of this study. Other types of size effects, such as edge softening [89], elastic size effects [90] and plasticity size effects [90, 91] have been discussed in previously.

To obtain the specific dimension of size where the FPZ can be fully developed within a limited number of samples, the size of the entire plate is scaled in the index of power of 2, as shown in Figure 4.20 and Table 4.2 with corresponding plate width D. The crack orientation is selected to be normal to the the direction of the non-inclined strut of the

hexagonal and of the triangular lattices as shown in Figure 4.21. This selection is made in order to minimize strain energy around the crack tip which should optimize the fracture toughness [52], despite that this assumption is not supported by the case of triangular lattice shown in [56].

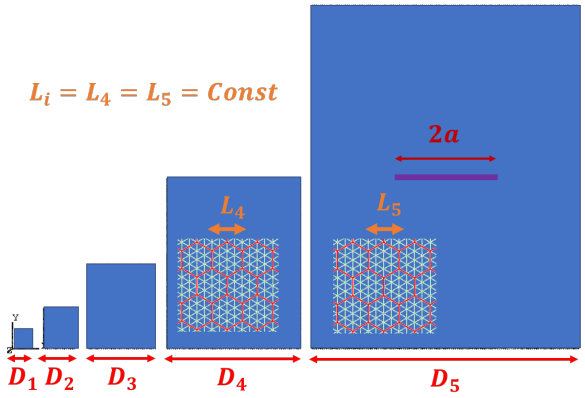


Figure 4.20: Scaled CCP

label	1	2	3	4	5
scale	2^0	2^1	2^2	2^3	2^4
D (mm)	13.86	27.71	55.43	110.85	221.70

Table 4.2: Sizes and scales of the specimen

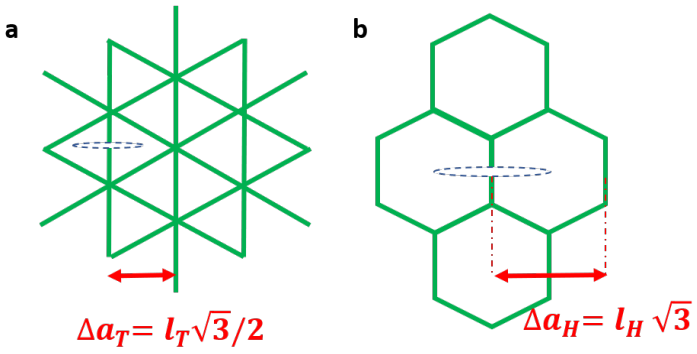


Figure 4.21: Crack morphology indicating the incremental crack length Δa and the crack orientation for (a) triangular and (b) hexagonal lattice

4.9.3 Adjusted Type II Size Effect with Relative Density of UCM

To provide a fair comparison between the pure lattice and the UCM, the Bažant size effect law is adjusted with the relative density of lattice. The $\bar{\rho}$ of a pure lattice can be calculated from the Eq. (2.5) in Chapter 2. On the other hand, to get the relative density of the UCM, the thickness of lattice needs to be considered as well. The representative cell of the UCM is first considered, see Figure 4.1. Given the definition from the Eq. (2.4) shown in Chapter 2, the solid wall volume of triangular and hexagonal lattice can be calculated respectively:

$$V_{s_T} = [3(3+3+4)\cdot L_T t_T + 6\cdot 2L_T\frac{t_T}{2}]b_T \tag{4.11}$$

and

$$V_{s_H} = 6\cdot L_H\frac{t_H}{2}b_H \tag{4.12}$$

, where V_{S_i} is the solid wall volume, L_i is the length of strut, t_i is the in-plane thickness of strut and b_i is the out-of-plane width of strut with i of T and H representing the triangular and hexagonal structure respectively. Secondly, the topology volume can be expressed as

$$V_{topology} = 6\cdot 2\cdot\frac{\sqrt{3}}{2}\cdot L_H^2 b_{UCM} \tag{4.13}$$

or

$$V_{topology} = A_p b_{UCM} \tag{4.14}$$

, where A_p is the projected area of the RVE, and $b_{UCM} = b_T + b_H$. Plugging forgoing volume expressions into Eq. (2.4), gives

$$\bar{\rho}_{UCM} = \frac{V_{s_T} + V_{s_H}}{V_{topology}} \tag{4.15}$$

Noting that $V_T = \bar{\rho}_T A_p b_T$ and that $V_H = \bar{\rho}_H A_p b_H$, gives

$$\bar{\rho}_{UCM} = \frac{\bar{\rho}_T b_T + \bar{\rho}_H b_H}{b_T + b_H} \tag{4.16}$$

As a consequence, the relative density of UCM is derived. Deviding the nominal stress σ_N in Eq. (4.10) by the $\bar{\rho}$, gives

$$\frac{\sigma_N}{\bar{\rho}} = \sqrt{\frac{E' G_f / \bar{\rho}^2}{g_0' c_f + g_0 D}} = \frac{f_t B}{\bar{\rho}} \left(1 + \frac{D}{D_0}\right)^{-1/2} \tag{4.17}$$

, where the fracture energy G_f can be expressed as

$$G_f = \frac{c_f g_0'}{E' c_2'} \bar{\rho}^2 \tag{4.18}$$

with c_2' being the constant term of the regression line. Rewriting the equation by taking the inverse on both sides with a square, gives

$$\left(\frac{\bar{\rho}}{\sigma_N}\right)^2 = \frac{\bar{\rho}^2}{(f_t B)^2} + \frac{\bar{\rho}^2}{(f_t B)^2 D_0} D \tag{4.19}$$

Recalling the least square regression, we can find a fitting line $Y = C_2 + C_1 X$. The Eq. (4.19) can be expressed in terms of a fitting line, where the criterion variable Y is substituted with $(\sigma_N/\bar{\rho})^{-2}$, and D is the predictor variable X for this least square regression equation. This gives the value of coefficients C_1 and C_2 to get $D_0(f_t B/\bar{\rho})^{-2}$ and $(f_t B/\bar{\rho})^{-2}$ respectively. By dividing C_1 by C_2, gives

$$D_0 = \frac{C_1}{C_2} = c_f \frac{g_0'}{g_0} \tag{4.20}$$

As a preliminary study, g_0 and g_0' are approximated by setting $g_0 = \pi\alpha$, which is the value for the infinite plate with a central crack. This gives $g_0' = \pi$. Plugging in these two values, c_f can be calculate accordingly. c_f is an additional crack length, which can be seen as about half the length of the FPZ.

In addition, the value of $f_t B$ is computed by the coefficients from the function of the fitting line. Now, plugging C_2 into Eq. (4.18), provides the fracture energy for the UCM

and for the pure lattice. The fitting condition is shown in Figure 4.22. Both the triangular lattice and the UCM have high related coefficient close to 1.0. The results are summarized in Table 4.3 and plotted in Figure 4.23.

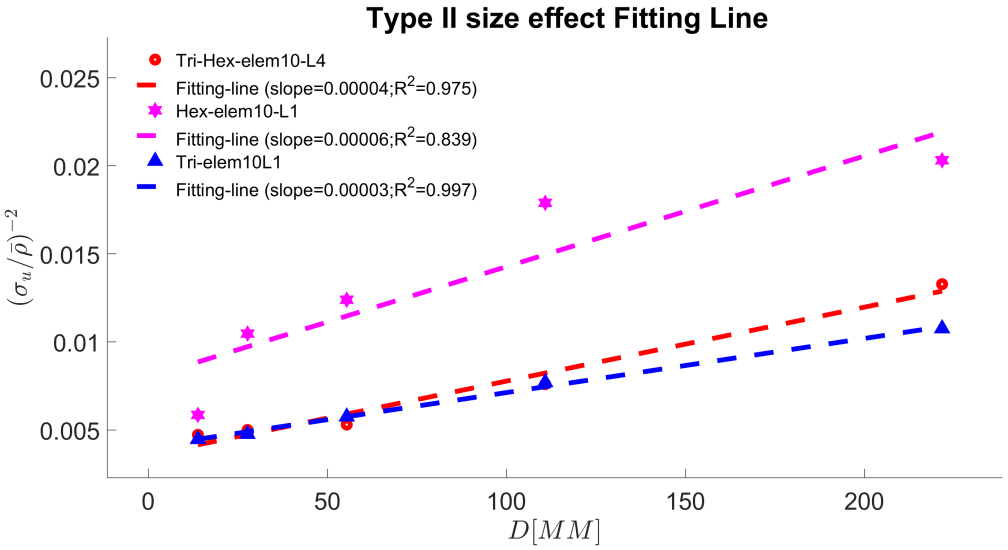


Figure 4.22: Fitting line using least-squares regression method

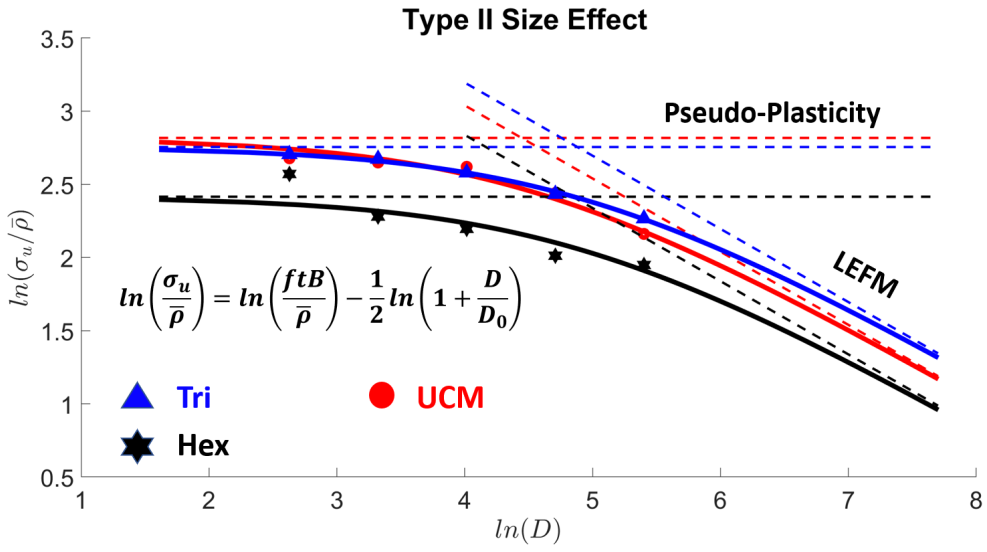


Figure 4.23: Bažant type II size effect curves

Lattice	Tri	Hex	UCM
$D_0(mm)$	131.6787	127.0705	85.1459
$c_f(mm)$	19.7518	19.0606	12.7719
$G_f(N/mm)$	0.07665	3.342	0.03367

Table 4.3: Transitional size D_0, additional crack length c_f and fracture energy G_f

4.9.4 Results Analysis of Fracture Behavior and Type II Size Effect

It is noted that the specimen of the largest size is not large enough to converge to the theoretical LEFM region as seen in Figure 4.23. Therefore, the Bažant size effect law is implemented to calculate the fracture energy and the results are plotted in logarithmic scale in Figure 4.23 as well. The ultimate nominal stress is measured at the peak. The horizontal dash line represents the Pseudo-Plasticity, the strength limit or strength-controlled region, meaning that when the dimension of specimen is close to this asymptotic, the structure remains strength dominant. On the other hand, the diagonal dash line speaks to the LEFM limit or energy-controlled region. When the size of the structure reaches this specific region, the structure is seem as being energy dominant and can be described by the LEFM. In Figure 4.23, the hexagonal lattice has the lowest specific strength due to its ductile behavior. It is noted that the curve of UCM goes below the curve of the triangular lattice when entering the transition region. Additionally, these two curves show that the distance between them remains the same when approaching the LEFM region.

Furthermore, to understand the fracture behavior in UCMs, the contour plot of CCP is provided in Figure 4.24 and compared with the case of pure lattice (triangular) (Figure 4.25). The process of failure in the CCP can be characterized into 3 stages as shown in Figure 4.24 and 4.25. More contour plots are attached in Appendix G. In stage I, the FPZ starts to develop at crack tip in both the triangular lattice and UCMs when the exerted load reaching the ultimate stress. During this stage, the hexagonal lattice component redistributes load, leading to a larger plastic deformation region in the UCMs. However this doesn't help prevent localization in Stage II where FPZ is supposed to keep developing. Instead, the

load distribution could hinder the FPZ in triangular lattice part in the UCMs. This is different from the pure triangular lattice where the FPZ continues to grow freely around the crack tip as seen in Figure 4.25. Finally, the dominant crack propagates through the lattice in stage III, .

In summary, the strain hardening behavior improves the toughness of lattice but also brings brittleness to lattice. Although the hexagonal lattice can help distribute the load during failure process, it also constraints the deformation of triangular lattice. This thus limits the growth of FPZ in UCMs, leading to A drop in fracture energy as a trade off for toughness improvement.

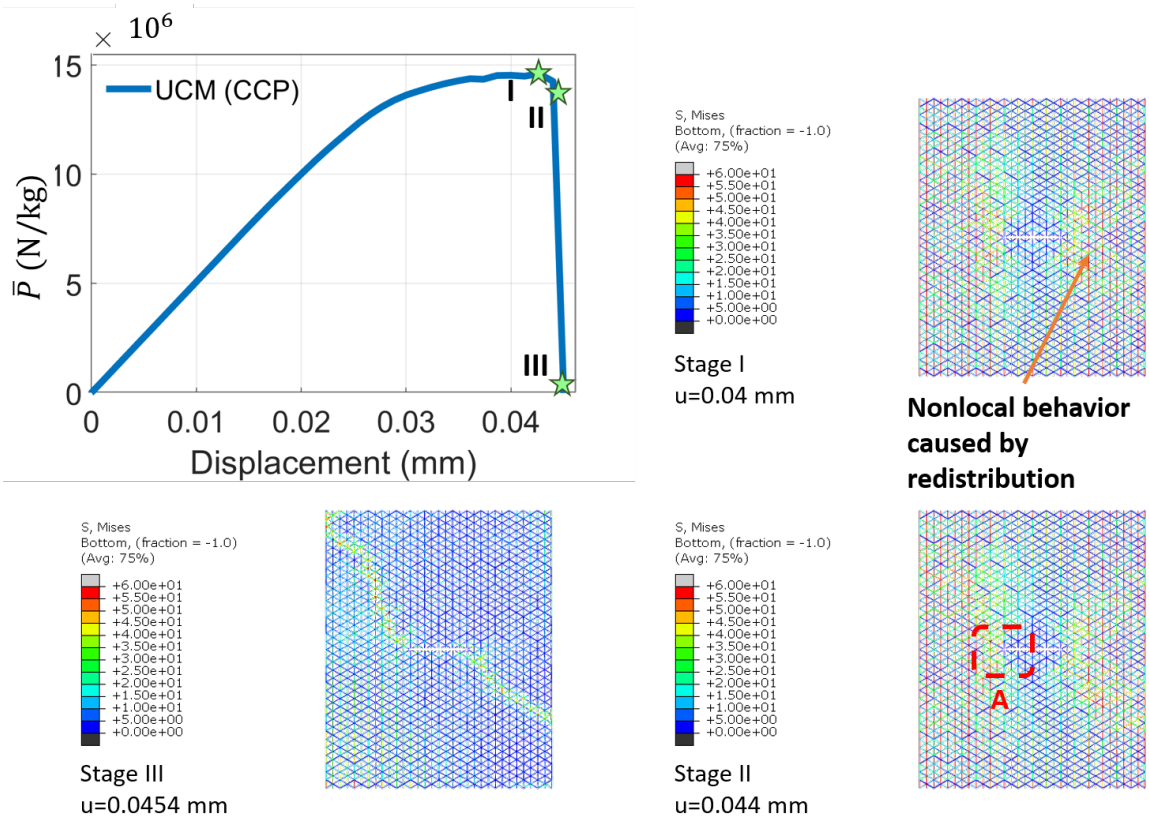


Figure 4.24: Tensile test result of CCP of UCMs plotted in specific force -displacement curve and the contour plot of each performance stages

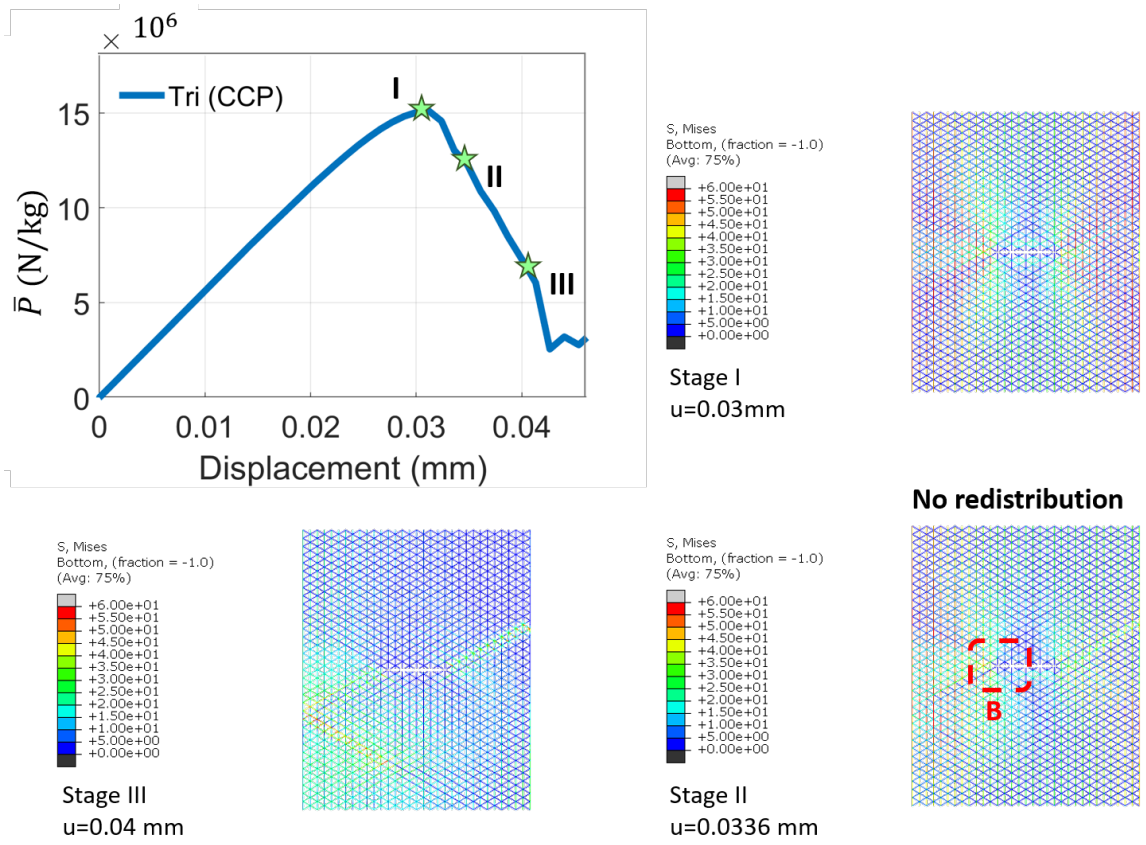


Figure 4.25: Tensile test result of CCP of triangular lattice plotted in specific force-displacement curve and the contour plot of each performance stages

Additionally, a visual estimation of the failed structure in Figure 4.26 and 4.27 reveals that the strut always fails near the node. This is also mentioned in [92]. Although the UCMs have been observed to help distribute load before the initiation of cracking, they still fail around nodes. This is possible another reason why the UCMs can not increase the fracture energy.

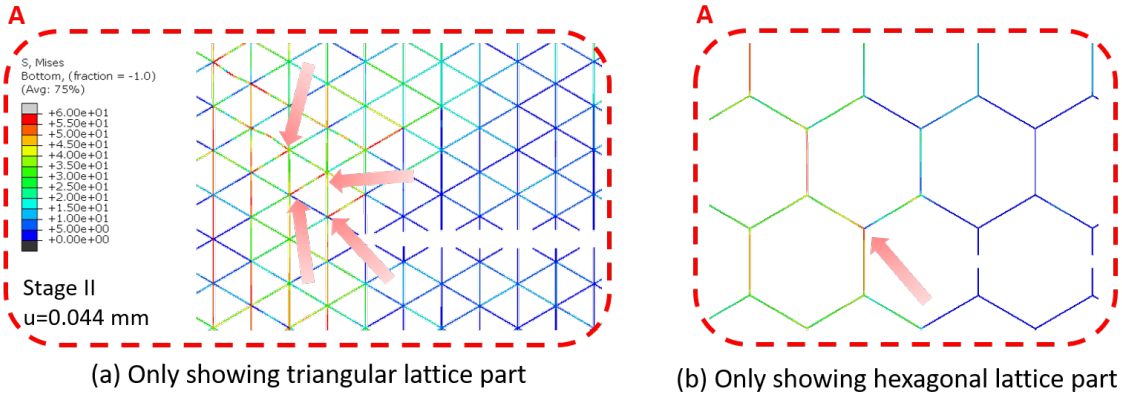


Figure 4.26: The failure occurs in a strut close to the joint. The rectangular area "A" in Figure 4.24 is zoom in here: (a) Visualization of the triangular lattice part only and (b) visualization of the hexagonal lattice part only. The red arrows indicate the location of failure elements.

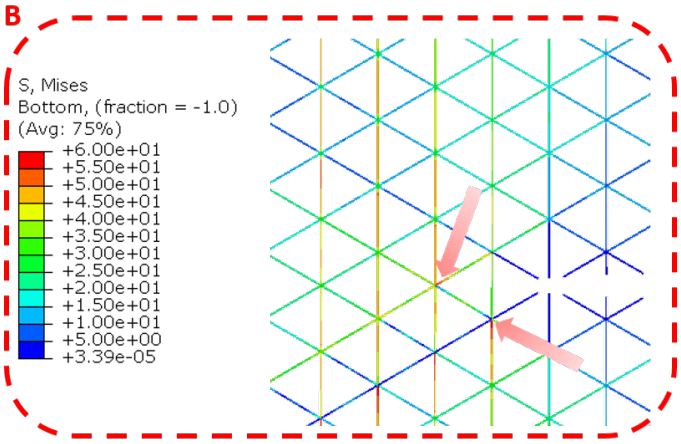


Figure 4.27: The failure occurs in a strut close to the joint. The rectangular area "B" from Figure 4.25 is zoomed in here. The red arrows indicate the location of failure elements.

4.10 Analysis and Discussion for Improvement of Damage Tolerance in UCMs

An advanced evaluation is provided in Figure 4.28 and 4.29. These are plotted in a logarithm scale for fracture energy G_f, specific strength and specific stiffness. Both specific strength and specific stiffness are defined in terms of stiffness and ultimate strength over

relative density as $\sigma_u/\bar{\rho}$ and $E/\bar{\rho}$ respectively. The value of ultimate strength of the UCM is determined as 1.137 MPa and of the hexagonal lattice as 0.44436246 MPa (see Appendix E). Compared to pure triangular lattice, the UCM has a higher specific toughness and specific strength by 2361% and by 3.175% respectively with no drop in specific peak load at the same size ($D_2 = 27.712814331mm$) along with only 0.3% drop in specific stiffness but sacrifices 56.2% fracture energy. When the UCM is compared to the hexagonal lattice, it has 20.267% increase in specific peak load at the same size ($D_2 = 27.712814331mm$) with a higher specific stiffness and specific strength by 332.365% and by 16.12% respectively, though the drops in specific toughness and fracture energy may be significant.

In the last analysis, the UCM acts as a good compromise among its component structures by applying in-homogeneity and non-locality via multiple length scales. All contour plots of provided simulation are attached in Appendix G.

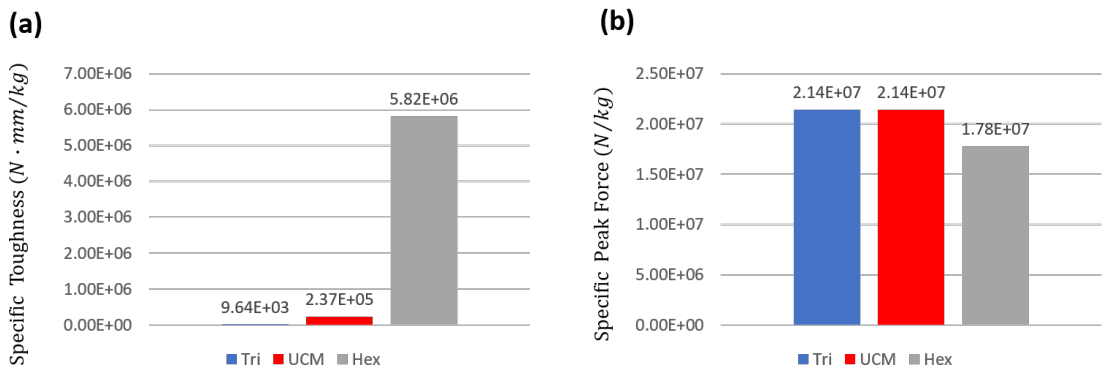
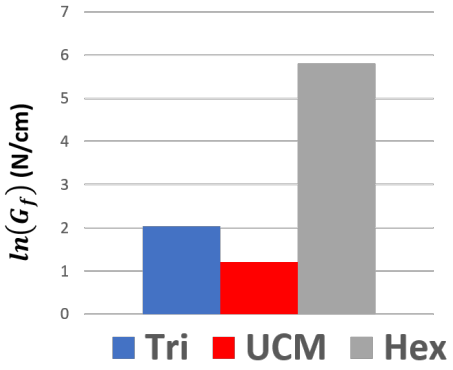
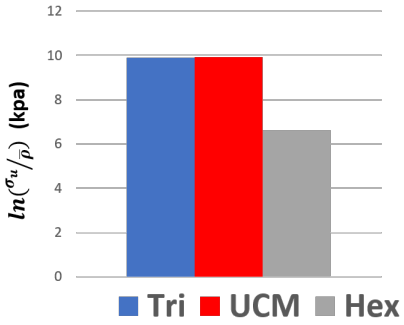


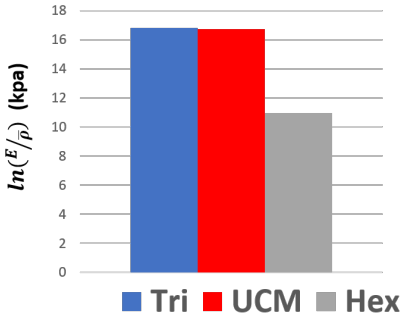
Figure 4.28: The comparison of mechanical properties for the triangular lattice, hexagonal lattice and UCMs at size of Scale-2: (a) Specific toughness and (b) Specific Peak Force.



(a) Fracture energy comparison



(b) Specific strength comparison



(c) Specific stiffness comparison

Figure 4.29: The comparison of mechanical properties for all lattices

Discussion of Damage Sensitivity Damage sensitivity can be seen as another property to estimate the damage tolerance. The material with smaller sensitivity to damage can maintain its stiffness well when there are imperfection and tiny damage in the materials.

4.10.1 Size Effect of Crack Length and Damage Sensitivity

In addition, the damage sensitivity presents the size effect as well. For the reported simulations, the ratio of specimen width D to semi-crack length a is a constant of 0.15. The result summarized in Figure 4.30 is plotted in the relative direct modulus $\bar{E}$ versus the fraction of semi-crack length a to a strut length ℓ, noted as a/ℓ.

$\bar{E}$ is defined in terms of the measured stiffness of center-cracked lattice plate E_m over

the Young's modulus of perfect lattice at the same relative density E_p as E_m/E_p. The ratio of a/ℓ varies from 2.0785 to 16.6277. It is noted that, the UCM has E_p of 1012.2(MPa) which is determined from the previous section. Both the UCM and the triangular lattice are insensitive to the crack length. By contrast, the hexagonal lattice converges as the fraction of crack length increases. The critical value is at $a/\ell \approx 4.5$. On the other hand, the expense of stiffness is about 10% in the UCM and the triangular lattice and is 12% in the hexagonal lattice. This implies that the hexagonal is slightly more sensitive to the damage than are the other two types.

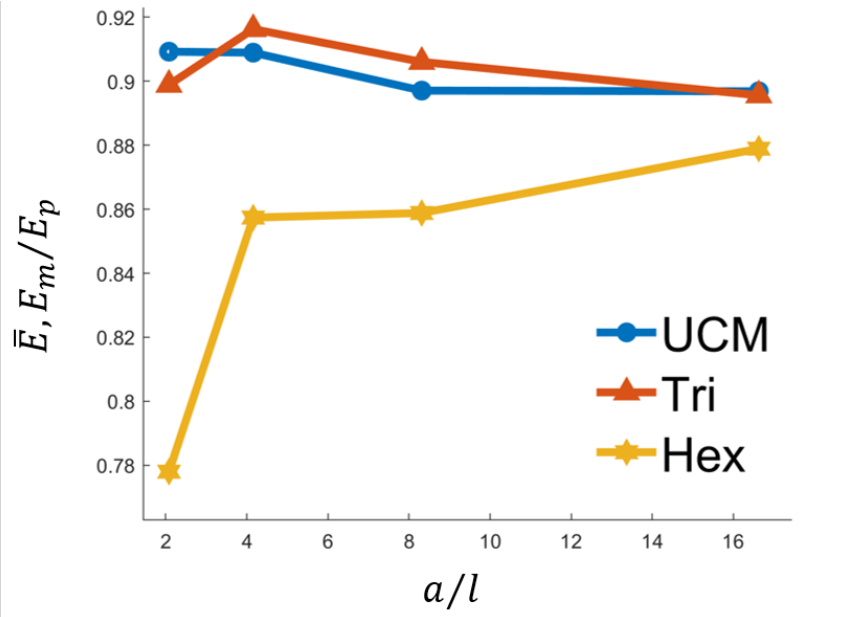


Figure 4.30: The relation between the effect of center-crack on stiffness of lattice and the specimen size

Chapter 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

5.1.1 Achievements

In pursuit of a better understanding of the fracturing behavior of cellular materials, this study first proposed a comprehensive investigation of the literature. Several analytical and computational models for estimating the strength, fracture toughness, and imperfection sensitivity were identified and analyzed critically.

A notable knowledge gap identified in this work is that most of the studies available in the literature have focused on elasto-plastic and brittle lattices whereas only a few have investigated quasi-brittle lattices. This is a serious problem considering that the vast majority of lattice and cellular structures do feature quasi-brittle materials and any structure is quasi-brittle at a certain length scale. Addressing this important problem lied at the heart of the present work which represents one of the few attempts to improve our understanding of quasi-brittle fracture in cellular materials and utilize this knowledge to enhance their damage tolerance.

Towards this goal, first a novel set of constitutive equations for quasi-brittle materials was implemented as a VUMAT user subroutine in Abaqus/Explicit. Mesh objectivity was guaranteed even in the presence of significantly large deformations by implementing a novel regularization approach that explicitly accounts for the geometrical variations of the beams. This new theory was applied to cellular materials for the first time and has been proven to be extremely effective. Thanks to the foregoing implementation damage and fracture could be simulated efficiently in a FE model featuring Timoshenko's beam elements. To enable the simulation of realistic lattices without resorting on Periodic Boundary Conditions (PBCs), which are not suitable for simulation of highly localized phenomena such as fracture, a new script for the automatic generation of the geometry, the meshing, the application of the

constraints, and the Boundary Conditions (BCs) was developed in Python. Compared to the tools available in Abaqus, this algorithm was shown to be significantly more efficient. Furthermore, the novel algorithm was shown to converge even in situations in which the in-house code did not. Thanks to the computational tools developed in this work, a better understanding of the fracturing behavior was achieved for four distinct quasi-brittle lattices: Triangular, Hexagon, Diamond, and Kagome. Extreme stiff and strong behavior with limited damage tolerance was identified in lattices featuring high connectivity such as the triangular and Kagome. On the other hand, it found that a lower node connectivity leads to a softer behavior with particularly pronounced energy dissipation and damage tolerance.

The foregoing results led to the exploration of a novel concept to enhance the fracturing behavior called "Ultra-Lattice" in which the concepts of non-locality and inhomogeneity are used in conjunction. Towards this goal, the idea is to combine two or more lattices featuring different fracture behaviors and characteristic length scales. This way, a system that combines the benefits of both lattices and promotes damage redistribution rather than localization can be obtained. Considering the results of the first phase of the project, a system featuring a triangular lattice which is very strong but brittle and an hexagonal lattice which is very soft but tough was used as a representative system. A conceptual Design of Experiment (DOE) approach was used to identify the best combination of relative density of the lattices in the UCM leading to a hardening behavior without a significant drop in specific peak load under tensile test.

Leveraging the generation algorithms and computational models developed in this work, it was shown that UCMs can be designed to provide superior combinations of toughness, stiffness and strength compared to the traditional lattices that compose them. In order to fully complete the investigation of mechanical behaviors of UCM, both the UCM and its component lattices were simulated at different sizes with and without a central crack. By doing so, first, the fracturing behavior in structural size effect on boundary layers was explored with non-central cracked specimens. Second, the Bažant type II size effect was modified to include the effect of relative density of cellular materials. As a consequence, it was to provide a mass objective comparison on the difference of fracture behavior among the UCM and its constituent lattices. The Bažant size effect plotted in Figure 4.23 showed that

the UCM and triangular lattice have comparable fracturing behavior governed by structural size effect.

5.1.2 Conclusions of Numerical Analysis for UCM

In view of the foregoing results, the UCMs were shown to improve damage tolerance without penalizing stiffness, strength and fracture toughness. The results of this study have shown an increase of 2361% in specific toughness and of 3.175% in specific strength with no measurable penalty in specific peak load and little penalty in specific stiffness. These results imply that the desired damage tolerance can be obtained by reworking the topology alone for cellular materials. Moreover, from the Bažant size effect curve, it can be seen that the performance of the UCM is not seriously altered by the size effect. When the size of the structure reaches the transition region, the UCM reveals little decline in strength. In addition, the UCM exhibits a hardening behavior, which serves as a warning precursor to a failure. As such, it is much safer to use UCM rather than the triangular lattice.

Moreover, it is found in the parametric study on the weight ratio, that the UCM exhibits the best mechanical performance when the weight fraction of the hexagonal lattice is around 28%. Although there is a 56.2% drop in the fracture energy, the UCM shows better specific toughness when compared to the triangular lattice. The UCM has a better specific stiffness than the hexagonal lattice, though with a drop in both fracture energy and in specific toughness. Moreover, in the size effects estimation, the free edges layer exhibits fewer effects on the UCM than the triangular and hexagonal lattices. Besides, the UCM displays less sensitivity to a central crack than the hexagonal lattice does. With all of these characteristics considered, the UCM has proven to be a good compromise between hexagonal lattice and triangular lattice.

This study provides a new, promising solution to improve damage tolerance in cellular materials, which is achieved via UCMs by engineering topology alone. This type of lattice-work is toughened to prevent collapse but remains adequate strength and stiffness, and is thus much safer for lattice formed applications.

5.2 Future Work

The study's prediction of fracture energy using the Bažant size effect represents a preliminary trial. The dimensionless energy release rate at the original crack length g_0 is approximated as $g_0 = \alpha\pi$. To get a more accurate measure of fracture energy, the energy compliance method should be implemented to find a closer solution of g_0. This method is demonstrated in Appendix F. It should be noted that a larger size of CCP is needed to obtain an agreement with the LEFM prediction when assessing the type II size effect.

Furthermore, doing more parametric studies can help develop a principle for the optimal combination of specific weight fraction. For the preliminary study, the UCM is combined with only two types of lattice structure, triangular and hexagonal. This opens the door for more creativity in the future to compare distinct combinations using different types of lattices, as well as using more than two length scales. Consider, also, that the UCM can exhibit in-homogeneity when using different cell-edge materials rather than using topology only. An example of this would be the use of a stiff lattice made of compliant cell-wall materials together with a use of ductile lattice composed of brittle cell-wall materials.

A further possibility is that as 2D UCMs become more fully developed, 3D UCMs can be researched and explored for potential applications. Once a 3D UCM model is developed, it is necessary to consider seriously the behavior of joints of lattices since the damage starts from regions close to nodes. As a consequence, a quick prototyping is required to visualize a practical and physical model. At the same time, experiments have to be designed to validate the provided numerical models accordingly. Finally, standards and specifications for UCM construction could be developed to provide design and manufacturing instructions.

Appendix A

PYTHON SCRIPT FOR THE MULTI-CELL LATTIE MODEL

```
class Lattice:
class Triangle(Lattice):
    def __init__(self, L = 1, k = 0, theta = pi/3.0, Part_Name = 'Triangle'):
        Lattice.__init__(self, L, k, theta, Part_Name)
    def draw(self, TotalRow = 2, TotalColumn = 2):
        self.TotalRow = TotalRow
        self.TotalColumn = TotalColumn
        # Parameter for Square
        b = 0.5*self.L
        c = cos(self.theta)
        s = sin(self.theta)
            # L = Leg length                                          /\                RVE:
            # B = side-edge length, where 0.5<c                    L /  \                    \ |^/ <~theta
            # theta = base angle                                    /   (\ <~theta            \|/
            # b = half of edge length                               ------                    /|\
        # Number of Lattice (4 small triangle make one middle triangle)    B=2b                / | \<~ L/2 = b
            # number of middle triangle = TotalRow * TotalColumn

        # Plot RVE
        model = mdb.models['Model-1']
        model.ConstrainedSketch(name='__profile__', sheetSize=200.0)
            # Duplicate and stack row
        memory = 0
        sketch_name = '__profile__'
        for row in range(0, self.TotalRow):
            for column in range(0, self.TotalColumn):
                model.sketches[sketch_name].Line(point1=(0.0+column*4*b*s, 0.0+row*4*b*c), point2=(
                    b*s+column*4*b*s, b*c+row*4*b*c))
                model.sketches[sketch_name].Line(point1=(0.0+column*4*b*s, 0.0+row*4*b*c), point2=(
                    0.0+column*4*b*s, b+row*4*b*c))
                model.sketches[sketch_name].Line(point1=(0.0+column*4*b*s, 0.0+row*4*b*c), point2=(
                    -b*s+column*4*b*s, b*c+row*4*b*c))
                model.sketches[sketch_name].Line(point1=(0.0+column*4*b*s, 0.0+row*4*b*c), point2=(
                    -b*s+column*4*b*s, -b*c+row*4*b*c))
                model.sketches[sketch_name].Line(point1=(0.0+column*4*b*s, 0.0+row*4*b*c), point2=(
                    0.0+column*4*b*s, -b+row*4*b*c))
                model.sketches[sketch_name].Line(point1=(0.0+column*4*b*s, 0.0+row*4*b*c), point2=(
                    b*s+column*4*b*s, -b*c+row*4*b*c))
```

(a) Triangle-1

```
            #   Make up hollow
        for row in range(0, self.TotalRow+1):
            for column in range(0, self.TotalColumn-1):
                model.sketches[sketch_name].Line(point1=(0.0+2*b*s+column*4*b*s, 0.0-2*b*c+row*4*b*c), point2=(
                    b*s+2*b*s+column*4*b*s, b*c-2*b*c+row*4*b*c))
                model.sketches[sketch_name].Line(point1=(0.0+2*b*s+column*4*b*s, 0.0-2*b*c+row*4*b*c), point2=(
                    0.0+2*b*s+column*4*b*s, b-2*b*c+row*4*b*c))
                model.sketches[sketch_name].Line(point1=(0.0+2*b*s+column*4*b*s, 0.0-2*b*c+row*4*b*c), point2=(
                    -b*s+2*b*s+column*4*b*s, b*c-2*b*c+row*4*b*c))
                model.sketches[sketch_name].Line(point1=(0.0+2*b*s+column*4*b*s, 0.0-2*b*c+row*4*b*c), point2=(
                    -b*s+2*b*s+column*4*b*s, -b*c-2*b*c+row*4*b*c))
                model.sketches[sketch_name].Line(point1=(0.0+2*b*s+column*4*b*s, 0.0-2*b*c+row*4*b*c), point2=(
                    0.0+2*b*s+column*4*b*s, -b-2*b*c+row*4*b*c))
                model.sketches[sketch_name].Line(point1=(0.0+2*b*s+column*4*b*s, 0.0-2*b*c+row*4*b*c), point2=(
                    b*s+2*b*s+column*4*b*s, -b*c-2*b*c+row*4*b*c))

        if sketch_name == '__profile__':
            model.Part(dimensionality=TWO_D_PLANAR, name=self.Part_Name, type=
                DEFORMABLE_BODY)
            print 'All set'
            model.parts[self.Part_Name].BaseWire(sketch=
                model.sketches['__profile__'])
            print 'Basewire'
            del model.sketches['__profile__']
            print 'Completed plot'
        else:
            model.parts[self.Part_Name].features['Wire-1'].setValues(sketch=
                model.sketches[sketch_name])
            del model.sketches[sketch_name]
            model.parts[self.Part_Name].regenerate()

        # Clean up extra edges
```

(b) Triangle-2

Figure A.1: Triangular lattice-1

```
    # Clean up extra edges
    Part = model.parts[self.Part_Name]
    Alledges = Part.edges
    Insides = Part.edges.getByBoundingBox(xMin=0, yMin=-b, xMax=((self.TotalColumn-1)*4)*b*s, yMax=(self.TotalRow*2-1)*b)
    Part.Set(edges=Insides, name='Insides')
    Part.Set(edges=Alledges, name=self.Part_Name+'_Alledges')
    Part.SetByBoolean(sets=[Part.sets[self.Part_Name+'_Alledges'],Part.sets['Insides']],name='Outsides',operation=DIFFERENCE)
    Part.RemoveWireEdges(wireEdgeList=Part.sets['Outsides'].edges)
    del Part.sets[self.Part_Name+'_Alledges']
    print 'Complete trimming'
    Part.regenerate()
    print 'Regenerate'

    # Build top&bottom Vertices sets for BC's
    Part.Set(edges=Part.edges, name=self.Part_Name+'_AllEdges')
    Part.Set(vertices=Part.vertices, name=self.Part_Name+'_AllVertices')
    bottomVertices = []
    for i in Part.vertices:
        if i.pointOn[0][1] == -b:
            bottomVertices.append(Part.vertices.findAt(i.pointOn))
    Part.Set(vertices=bottomVertices, name=self.Part_Name+'_BottomVertices')
    topVertices = []
    for i in Part.vertices:
        if abs(i.pointOn[0][1] - (self.TotalRow*2-1)*b) < b*0.001:
            topVertices.append(Part.vertices.findAt(i.pointOn))
    Part.Set(vertices=topVertices, name=self.Part_Name+'_TopVertices')
    leftVertices = []
    for i in Part.vertices:
        if i.pointOn[0][0] == 0:
            leftVertices.append(Part.vertices.findAt(i.pointOn))
    Part.Set(vertices=leftVertices, name=self.Part_Name+'_LeftVertices')
    rightVertices = []
    for i in Part.vertices:
        if abs(i.pointOn[0][0] - ((self.TotalColumn-1)*4)*b*s) < b*0.001:
            rightVertices.append(Part.vertices.findAt(i.pointOn))
    Part.Set(vertices=rightVertices, name=self.Part_Name+'_RightVertices')
    print 'Build vertex sets'
```

(a) Triangle-3

```
    # Clean up extra edges
    Part = model.parts[self.Part_Name]
    Alledges = Part.edges
    Insides = Part.edges.getByBoundingBox(xMin=0, yMin=-b, xMax=((self.TotalColumn-1)*4)*b*s, yMax=(self.TotalRow*2-1)*b)
    Part.Set(edges=Insides, name='Insides')
    Part.Set(edges=Alledges, name=self.Part_Name+'_Alledges')
    Part.SetByBoolean(sets=[Part.sets[self.Part_Name+'_Alledges'],Part.sets['Insides']],name='Outsides',operation=DIFFERENCE)
    Part.RemoveWireEdges(wireEdgeList=Part.sets['Outsides'].edges)
    del Part.sets[self.Part_Name+'_Alledges']
    print 'Complete trimming'
    Part.regenerate()
    print 'Regenerate'

    # Build top&bottom Vertices sets for BC's
    Part.Set(edges=Part.edges, name=self.Part_Name+'_AllEdges')
    Part.Set(vertices=Part.vertices, name=self.Part_Name+'_AllVertices')
    bottomVertices = []
    for i in Part.vertices:
        if i.pointOn[0][1] == -b:
            bottomVertices.append(Part.vertices.findAt(i.pointOn))
    Part.Set(vertices=bottomVertices, name=self.Part_Name+'_BottomVertices')
    topVertices = []
    for i in Part.vertices:
        if abs(i.pointOn[0][1] - (self.TotalRow*2-1)*b) < b*0.001:
            topVertices.append(Part.vertices.findAt(i.pointOn))
    Part.Set(vertices=topVertices, name=self.Part_Name+'_TopVertices')
    leftVertices = []
    for i in Part.vertices:
        if i.pointOn[0][0] == 0:
            leftVertices.append(Part.vertices.findAt(i.pointOn))
    Part.Set(vertices=leftVertices, name=self.Part_Name+'_LeftVertices')
    rightVertices = []
    for i in Part.vertices:
        if abs(i.pointOn[0][0] - ((self.TotalColumn-1)*4)*b*s) < b*0.001:
            rightVertices.append(Part.vertices.findAt(i.pointOn))
    Part.Set(vertices=rightVertices, name=self.Part_Name+'_RightVertices')
    print 'Build vertex sets'
```

(b) Triangle-4

Figure A.2: Triangular lattice-2

```
######################
class Hexagon(Lattice):
    def __init__(self, L = 1, k = 1, theta = pi/6.0, Part_Name = 'Hexagon'):
        Lattice.__init__(self, L, k, theta, Part_Name)
    def draw(self, TotalRow = 1, TotalColumn = 1):
        self.TotalRow = TotalRow
        self.TotalColumn = TotalColumn
        # Parameter for Hex
            # L = edge length                                          RVE:       h|
            # theta = angle between h & L - 90 degree                             |_ _ _ _
            # h = hight length                                                   L/ \)->theta
            #                                                                    /   \L
        h = self.k*self.L
        c = cos(self.theta)
        s = sin(self.theta)

        sketch_name = '__profile__'

        # Plot RVE
        model = mdb.models['Model-1']
        model.ConstrainedSketch(name='__profile__', sheetSize=200.0)
        model.Part(dimensionality=TWO_D_PLANAR, name=self.Part_Name, type=
            DEFORMABLE_BODY)
        model.sketches['__profile__'].Line(point1=(0.0, 0.0), point2=(
            0.0, h))
        model.sketches['__profile__'].Line(point1=(0.0, 0.0), point2=(
            self.L*c, -s*self.L))
        model.sketches['__profile__'].Line(point1=(0.0, 0.0), point2=(
            -self.L*c, -s*self.L))

        # Duplicate bottom column
```

(a) Hexagon-1

```
        # Duplicate bottom column
        for column in range(1, 1+self.TotalColumn):
            model.sketches[sketch_name].Line(point1=(0.0+column*2*self.L*c, 0.0), point2=(
                0.0+column*2*self.L*c, h))
            model.sketches[sketch_name].Line(point1=(0.0+column*2*self.L*c, 0.0), point2=(
                self.L*c+column*2*self.L*c, -s*self.L))
            model.sketches[sketch_name].Line(point1=(0.0+column*2*self.L*c, 0.0), point2=(
                -self.L*c+column*2*self.L*c, -s*self.L))

        # Duplicate and stack row
        for row in range(1, 1+self.TotalRow):
            for column in range(self.TotalColumn+1):
                model.sketches[sketch_name].Line(point1=(0.0+(row%2)*self.L*c+(column-(row%2))*2*self.L*c,
                    0.0+row*(h+self.L*s)), point2=(
                    0.0+(row%2)*self.L*c+(column-(row%2))*2*self.L*c, h+row*(h+self.L*s)))
                model.sketches[sketch_name].Line(point1=(0.0+(row%2)*self.L*c+(column-(row%2))*2*self.L*c,
                    0.0+row*(h+self.L*s)), point2=(
                    self.L*c+(row%2)*self.L*c+(column-(row%2))*2*self.L*c, -s*self.L+row*(h+self.L*s)))
                model.sketches[sketch_name].Line(point1=(0.0+(row%2)*self.L*c+(column-(row%2))*2*self.L*c,
                    0.0+row*(h+self.L*s)), point2=(
                    -self.L*c+(row%2)*self.L*c+(column-(row%2))*2*self.L*c, -s*self.L+row*(h+self.L*s)))

        if sketch_name == '__profile__':
            model.Part(dimensionality=TWO_D_PLANAR, name=self.Part_Name, type=
                DEFORMABLE_BODY)
            print 'All set'
            model.parts[self.Part_Name].BaseWire(sketch=
                model.sketches['__profile__'])
            print 'Basewire'
            del model.sketches['__profile__']
            print 'Completed plot'
        else:
            model.parts[self.Part_Name].features['Wire-1'].setValues(sketch=
                model.sketches[sketch_name])
            del model.sketches[sketch_name]
            model.parts[self.Part_Name].regenerate()
            print 'Completed plot'
```

(b) Hexagon-2

Figure A.3: Hexagonal lattice-1

```
            print 'Completed plot'

        # Clean up extra edges (old one)
        Part = mdb.models['Model-1'].parts[self.Part_Name]
        Allvertices = Part.vertices
        extraEdge = []
        for i in Allvertices:
            if len(i.getEdges()) == 1:
                extraEdge.append(Part.edges.findAt(i.pointOn))
            # Remove the corner one
        extraEdge.remove(Part.edges.findAt(((-self.L*c,-s*self.L,0.0),)))
        Part.Set(edges=extraEdge, name=self.Part_Name+'_ExtraEdges')
        Part.RemoveWireEdges(wireEdgeList=Part.sets[self.Part_Name+'_ExtraEdges'].edges)
        del mdb.models['Model-1'].parts[self.Part_Name].sets[self.Part_Name+'_ExtraEdges']

        print 'Completed trimming'
        mdb.models['Model-1'].parts[self.Part_Name].regenerate()
        print 'Regenerated'
        # Build sets
        Part.Set(edges=Part.edges, name=self.Part_Name+'_AllEdges')
        Part.Set(vertices=Part.vertices, name=self.Part_Name+'_AllVertices')
        # Build top&bottom Vertices sets for BC's
        bottomVertices = []
        for i in Part.vertices:
            if i.pointOn[0][1] < 0:
                bottomVertices.append(Part.vertices.findAt(i.pointOn))
        Part.Set(vertices=bottomVertices, name=self.Part_Name+'_BottomVertices')
        topVertices = []
        for i in Part.vertices:
            if abs(i.pointOn[0][1] - self.TotalRow*(h+self.L*s)) < self.L*0.001:
                topVertices.append(Part.vertices.findAt(i.pointOn))
        Part.Set(vertices=topVertices, name=self.Part_Name+'_TopVertices')
        leftVertices = []
        for i in Part.vertices:
            if abs(i.pointOn[0][0] + self.L*c) < self.L*0.001:
                leftVertices.append(Part.vertices.findAt(i.pointOn))
        Part.Set(vertices=leftVertices, name=self.Part_Name+'_LeftVertices')
        rightVertices = []
        for i in Part.vertices:
            if abs(i.pointOn[0][0] - 2*self.TotalColumn*self.L*c) < self.L*0.001:
                rightVertices.append(Part.vertices.findAt(i.pointOn))
        Part.Set(vertices=rightVertices, name=self.Part_Name+'_RightVertices')
        # Build sets of edges
```

(a) Hexagon-3

```
            print 'Completed plot'

        # Clean up extra edges (old one)
        Part = mdb.models['Model-1'].parts[self.Part_Name]
        Allvertices = Part.vertices
        extraEdge = []
        for i in Allvertices:
            if len(i.getEdges()) == 1:
                extraEdge.append(Part.edges.findAt(i.pointOn))
            # Remove the corner one
        extraEdge.remove(Part.edges.findAt(((-self.L*c,-s*self.L,0.0),)))
        Part.Set(edges=extraEdge, name=self.Part_Name+'_ExtraEdges')
        Part.RemoveWireEdges(wireEdgeList=Part.sets[self.Part_Name+'_ExtraEdges'].edges)
        del mdb.models['Model-1'].parts[self.Part_Name].sets[self.Part_Name+'_ExtraEdges']

        print 'Completed trimming'
        mdb.models['Model-1'].parts[self.Part_Name].regenerate()
        print 'Regenerated'
        # Build sets
        Part.Set(edges=Part.edges, name=self.Part_Name+'_AllEdges')
        Part.Set(vertices=Part.vertices, name=self.Part_Name+'_AllVertices')
        # Build top&bottom Vertices sets for BC's
        bottomVertices = []
        for i in Part.vertices:
            if i.pointOn[0][1] < 0:
                bottomVertices.append(Part.vertices.findAt(i.pointOn))
        Part.Set(vertices=bottomVertices, name=self.Part_Name+'_BottomVertices')
        topVertices = []
        for i in Part.vertices:
            if abs(i.pointOn[0][1] - self.TotalRow*(h+self.L*s)) < self.L*0.001:
                topVertices.append(Part.vertices.findAt(i.pointOn))
        Part.Set(vertices=topVertices, name=self.Part_Name+'_TopVertices')
        leftVertices = []
        for i in Part.vertices:
            if abs(i.pointOn[0][0] + self.L*c) < self.L*0.001:
                leftVertices.append(Part.vertices.findAt(i.pointOn))
        Part.Set(vertices=leftVertices, name=self.Part_Name+'_LeftVertices')
        rightVertices = []
        for i in Part.vertices:
            if abs(i.pointOn[0][0] - 2*self.TotalColumn*self.L*c) < self.L*0.001:
                rightVertices.append(Part.vertices.findAt(i.pointOn))
        Part.Set(vertices=rightVertices, name=self.Part_Name+'_RightVertices')
        # Build sets of edges
```

(b) Hexagon-4

Figure A.4: Hexagonal lattice-2

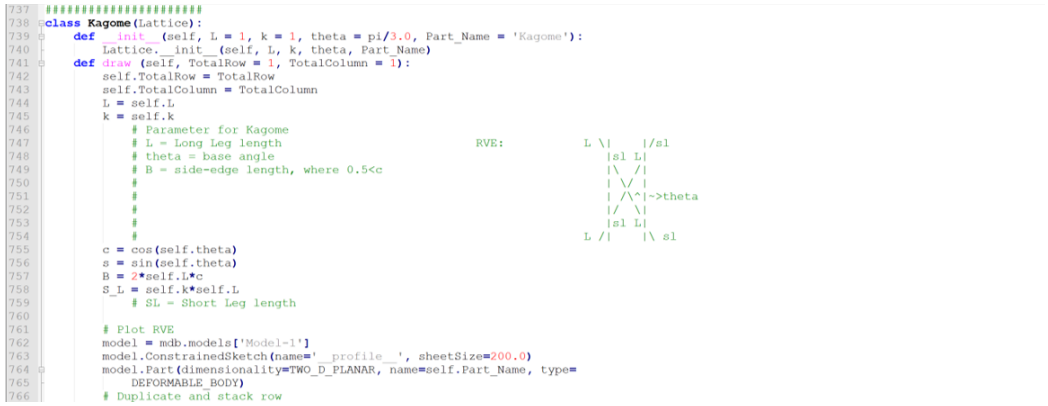
```
######################
class Diamond(Lattice):
    def __init__(self, L = 3, k = 1, theta = pi/6.0, Part_Name = 'Diamond'):
        Lattice.__init__(self, L, k, theta, Part_Name)
    def draw(self, TotalRow = 1, TotalColumn = 1):
        self.TotalRow = TotalRow
        self.TotalColumn = TotalColumn
            # Parameter for Square
            # L = edge length                                          RVE:      /\ L
            # theta = angle between bottom and x-axes                          /  \
            # phi = half angle between two edges                               \  /
            # phi+theta = pi/2                                                  \/_)_~>_theta
        c = cos(self.theta)
        s = sin(self.theta)

        # Duplicate and stack row
        model = mdb.models['Model-1']
        model.ConstrainedSketch(name='__profile__', sheetSize=200.0)
        for row in range(0, self.TotalColumn):
            for column in range(0, self.TotalRow):
                model.sketches['__profile__'].Line(point1=(0.0+column*2*self.L*c, 0.0+row*2*self.L*s), point2=(
                    self.L*c+column*2*self.L*c, self.L*s+row*2*self.L*s))
                model.sketches['__profile__'].Line(point1=(self.L*c+column*2*self.L*c, self.L*s+row*2*self.L*s), point2=(
                    2*self.L*c+column*2*self.L*c, 0+row*2*self.L*s))
                model.sketches['__profile__'].Line(point1=(2*self.L*c+column*2*self.L*c, 0+row*2*self.L*s), point2=(
                    self.L*c+column*2*self.L*c, -self.L*s+row*2*self.L*s))
                model.sketches['__profile__'].Line(point1=(0.0+column*2*self.L*c, 0.0+row*2*self.L*s), point2=(
                    self.L*c+column*2*self.L*c, -self.L*s+row*2*self.L*s))
                print '1'
            print '2'

        # Build profile in part
        model.Part(dimensionality=TWO_D_PLANAR, name=self.Part_Name, type=
            DEFORMABLE_BODY)
        model.parts[self.Part_Name].BaseWire(sketch=
            model.sketches['__profile__'])
        del model.sketches['__profile__']
        # Build top&bottom Vertices sets for BC's
```

(a) Diamond-1

```
        # Build top&bottom Vertices sets for BC's
        Part = mdb.models['Model-1'].parts[self.Part_Name]
        Part.Set(edges=Part.edges, name=self.Part_Name+'_AllEdges')
        Part.Set(vertices=Part.vertices, name=self.Part_Name+'_AllVertices')
        bottomVertices = []
        for i in Part.vertices:
            if i.pointOn[0][1] < 0:
                bottomVertices.append(Part.vertices.findAt(i.pointOn))
        Part.Set(vertices=bottomVertices, name=self.Part_Name+'_BottomVertices')
        topVertices = []
        for i in Part.vertices:
            if abs(i.pointOn[0][1] - (self.TotalRow*2-1)*self.L*s) < self.L*0.001:
                topVertices.append(Part.vertices.findAt(i.pointOn))
        Part.Set(vertices=topVertices, name=self.Part_Name+'_TopVertices')
        leftVertices = []
        for i in Part.vertices:
            if i.pointOn[0][0] == 0:
                leftVertices.append(Part.vertices.findAt(i.pointOn))
        Part.Set(vertices=leftVertices, name=self.Part_Name+'_LeftVertices')
    def getHight(self):
        # How tall the structure is
        model = mdb.models['Model-1']
        Part = model.parts[self.Part_Name]
        hight = Part.sets[self.Part_Name+'_TopVertices'].vertices[0].pointOn[0][1]-\
        Part.sets[self.Part_Name+'_BottomVertices'].vertices[0].pointOn[0][1]
        return hight # return a float
    def getWidth(self):
        # How wide the structure is
        model = mdb.models['Model-1']
        Part = model.parts[self.Part_Name]
        width = self.L*self.c*2*self.TotalColumn
        return width # return a float
    def getThickness(self, rho_bar = 1.0):
        # The corresponding thickness of lattice with specific relative density
        thickness = rho_bar*self.L*self.c*self.s
        return thickness
######################
```

(b) Diamond-2

Figure A.5: Diamond lattice-1



```
######################
class Kagome(Lattice):
    def __init__(self, L = 1, k = 1, theta = pi/3.0, Part_Name = 'Kagome'):
        Lattice.__init__(self, L, k, theta, Part_Name)
    def draw (self, TotalRow = 1, TotalColumn = 1):
        self.TotalRow = TotalRow
        self.TotalColumn = TotalColumn
        L = self.L
        k = self.k
            # Parameter for Kagome
            # L = Long Leg length                              RVE:          L \|    |/sl
            # theta = base angle                                                |sl L|
            # B = side-edge length, where 0.5<c                                 |\  /|
            #                                                                   | \/ |
            #                                                                   | /\^|~>theta
            #                                                                   |/  \|
            #                                                                   |sl L|
            #                                                                 L /|    |\ sl
        c = cos(self.theta)
        s = sin(self.theta)
        B = 2*self.L*c
        S_L = self.k*self.L
            # SL = Short Leg length

        # Plot RVE
        model = mdb.models['Model-1']
        model.ConstrainedSketch(name='__profile__', sheetSize=200.0)
        model.Part(dimensionality=TWO_D_PLANAR, name=self.Part_Name, type=
            DEFORMABLE_BODY)
        # Duplicate and stack row
```

(a) Kagome-1

```
        # Duplicate and stack row
        for column in range(0, self.TotalColumn):
            for row in range(0, self.TotalRow):
                # Right half
                model.sketches['__profile__'].Line(point1=(0.0+column*2*L*(1+k)*s, 0.0+row*2*L*(1+k)*c), point2=(
                    L*s+column*2*L*(1+k)*s, L*c+row*2*L*(1+k)*c))
                model.sketches['__profile__'].Line(point1=(0.0+column*2*L*(1+k)*s, 0.0+row*2*L*(1+k)*c), point2=(
                    L*s+column*2*L*(1+k)*s, -L*c+row*2*L*(1+k)*c))
                model.sketches['__profile__'].Line(point1=(L*s+column*2*L*(1+k)*s, -L*c+row*2*L*(1+k)*c), point2=(
                    L*s+column*2*L*(1+k)*s, L*c+row*2*L*(1+k)*c))
                    # Non-triangle part
                model.sketches['__profile__'].Line(point1=(L*s+column*2*L*(1+k)*s, L*c+row*2*L*(1+k)*c), point2=(
                    L*s+column*2*L*(1+k)*s, L*(k+1)*c+row*2*L*(1+k)*c))
                model.sketches['__profile__'].Line(point1=(L*s+column*2*L*(1+k)*s, L*c+row*2*L*(1+k)*c), point2=(
                    L*(k+1)*s+column*2*L*(1+k)*s, L*(k+1)*c+row*2*L*(1+k)*c))
                model.sketches['__profile__'].Line(point1=(L*s+column*2*L*(1+k)*s, -L*c+row*2*L*(1+k)*c), point2=(
                    L*s+column*2*L*(1+k)*s, -L*(k+1)*c+row*2*L*(1+k)*c))
                model.sketches['__profile__'].Line(point1=(L*s+column*2*L*(1+k)*s, -L*c+row*2*L*(1+k)*c), point2=(
                    L*(k+1)*s+column*2*L*(1+k)*s, -L*(k+1)*c+row*2*L*(1+k)*c))
                    # Left half
                model.sketches['__profile__'].Line(point1=(0.0+column*2*L*(1+k)*s, 0.0+row*2*L*(1+k)*c), point2=(
                    -k*L*s+column*2*L*(1+k)*s, k*L*c+row*2*L*(1+k)*c))
                model.sketches['__profile__'].Line(point1=(0.0+column*2*L*(1+k)*s, 0.0+row*2*L*(1+k)*c), point2=(
                    -k*L*s+column*2*L*(1+k)*s, -k*L*c+row*2*L*(1+k)*c))
                model.sketches['__profile__'].Line(point1=(-k*L*s+column*2*L*(1+k)*s, -k*L*c+row*2*L*(1+k)*c), point2=(
                    -k*L*s+column*2*L*(1+k)*s, k*L*c+row*2*L*(1+k)*c))
                    # Non-triangle part
                model.sketches['__profile__'].Line(point1=(-k*L*s+column*2*L*(1+k)*s, k*L*c+row*2*L*(1+k)*c), point2=(
                    -k*L*s+column*2*L*(1+k)*s, L*(k+1)*c+row*2*L*(1+k)*c))
                model.sketches['__profile__'].Line(point1=(-k*L*s+column*2*L*(1+k)*s, k*L*c+row*2*L*(1+k)*c), point2=(
                    -L*(k+1)*s+column*2*L*(1+k)*s, L*(k+1)*c+row*2*L*(1+k)*c))
                model.sketches['__profile__'].Line(point1=(-k*L*s+column*2*L*(1+k)*s, -k*L*c+row*2*L*(1+k)*c), point2=(
                    -k*L*s+column*2*L*(1+k)*s, -L*(k+1)*c+row*2*L*(1+k)*c))
                model.sketches['__profile__'].Line(point1=(-k*L*s+column*2*L*(1+k)*s, -k*L*c+row*2*L*(1+k)*c), point2=(
                    -L*(k+1)*s+column*2*L*(1+k)*s, -L*(k+1)*c+row*2*L*(1+k)*c))
                print '1'
            model.parts[self.Part_Name].clearGeometryCache()
            print '2'
```

(b) Kagome-2

Figure A.6: Kagome lattice-1

```
            print '2'

        model.parts[self.Part_Name].BaseWire(sketch=
            model.sketches['__profile__'])
        del model.sketches['__profile__']
        Part = mdb.models['Model-1'].parts[self.Part_Name]
        Part.Set(edges=Part.edges, name=self.Part_Name+'_AllEdges')
        Part.Set(vertices=Part.vertices, name=self.Part_Name+'_AllVertices')
        # Build vertices sets
        bottomVertices = []
        baseline = -L*(k+1)*c
        for i in Part.vertices:
            if abs(i.pointOn[0][1] - baseline) < L*0.001:
                bottomVertices.append(Part.vertices.findAt(i.pointOn))
        Part.Set(vertices=bottomVertices, name=self.Part_Name+'_BottomVertices')
        topVertices = []
        topline = (2*self.TotalRow-1)*L*(k+1)*c
        for i in Part.vertices:
            if abs(i.pointOn[0][1] - topline) < L*0.001:
                topVertices.append(Part.vertices.findAt(i.pointOn))
        Part.Set(vertices=topVertices, name=self.Part_Name+'_TopVertices')
        leftVertices = []
        for i in Part.vertices:
            if abs(i.pointOn[0][0] + L*(k+1)*s) < L*0.001:
                leftVertices.append(Part.vertices.findAt(i.pointOn))
        Part.Set(vertices=leftVertices, name=self.Part_Name+'_LeftVertices')
        print 'Build vertex sets'
    def getHight(self):
        # How tall the structure is
        model = mdb.models['Model-1']
        Part = model.parts[self.Part_Name]
        hight = Part.sets[self.Part_Name+'_TopVertices'].vertices[0].pointOn[0][1]-\
        Part.sets[self.Part_Name+'_BottomVertices'].vertices[0].pointOn[0][1]
        return hight # return a float
    def getWidth(self):
        # How wide the structure is
        model = mdb.models['Model-1']
        Part = model.parts[self.Part_Name]
        width = (self.L+self.k*self.L)*self.s*2*self.TotalColumn
        return width # return a float
    def getThickness(self, rho_bar = 1.0):
        # The corresponding thickness of lattice with specific relative density
        thickness = rho_bar*self.L*self.c*self.s*(1+self.k)/(1+self.c)
        return thickness
######################
```

(a) Kagome-3

Figure A.7: Kagome lattice-2

Appendix B

FORTRAN SCRIPT FOR COHESIVE LAW

```
      subroutine vumat(
C Read only (unmodifiable)variables -
     1  nblock, ndir, nshr, nstatev, nfieldv, nprops, lanneal,
     2  stepTime, totalTime, dt, cmname, coordMp, charLength,
     3  props, density, strainInc, relSpinInc,
     4  tempOld, stretchOld, defgradOld, fieldOld,
     5  stressOld, stateOld, enerInternOld, enerInelasOld,
     6  tempNew, stretchNew, defgradNew, fieldNew,
C Write only (modifiable) variables -
     7  stressNew, stateNew, enerInternNew, enerInelasNew )
C
      include 'vaba_param.inc'
C
      dimension props(nprops), density(nblock), coordMp(nblock,*),
     1  charLength(nblock), strainInc(nblock,ndir+nshr),
     2  relSpinInc(nblock,nshr), tempOld(nblock),
     3  stretchOld(nblock,ndir+nshr),
     4  defgradOld(nblock,ndir+nshr+nshr),
     5  fieldOld(nblock,nfieldv), stressOld(nblock,ndir+nshr),
     6  stateOld(nblock,nstatev), enerInternOld(nblock),
     7  enerInelasOld(nblock), tempNew(nblock),
     8  stretchNew(nblock,ndir+nshr),
     8  defgradNew(nblock,ndir+nshr+nshr),
     9  fieldNew(nblock,nfieldv),
     1  stressNew(nblock,ndir+nshr), stateNew(nblock,nstatev),
     2  enerInternNew(nblock), enerInelasNew(nblock)
C
C
      character*80 cmname

C! Decide which material to be used
```

(a) Declaration

```
C! Decide which material to be used
      if (cmname(1:4) .eq. 'MAT1') then
            call vumat_MAT1(
     1  nblock, ndir, nshr, nstatev, nfieldv, nprops, lanneal,
     2  stepTime, totalTime, dt, cmname, coordMp, charLength,
     3  props, density, strainInc, relSpinInc,
     4  tempOld, stretchOld, defgradOld, fieldOld,
     5  stressOld, stateOld, enerInternOld, enerInelasOld,
     6  tempNew,stretchNew,defgradNew,fieldNew,
     7  stressNew, stateNew, enerInternNew, enerInelasNew)

      else if (cmname(1:4) .eq. 'MAT2') then
            call vumat_MAT2(
     1  nblock, ndir, nshr, nstatev, nfieldv, nprops, lanneal,
     2  stepTime, totalTime, dt, cmname, coordMp, charLength,
     3  props, density, strainInc, relSpinInc,
     4  tempOld, stretchOld, defgradOld, fieldOld,
     5  stressOld, stateOld, enerInternOld, enerInelasOld,
     6  tempNew,stretchNew,defgradNew,fieldNew,
     7  stressNew, stateNew, enerInternNew, enerInelasNew)
      end if
      return
      end
C!  Vumat code end here ----------------------

C! Material 1 for reference
```

(b) Decision of material

Figure B.1: Fortran script for cohesive law implementation-1

```
C! Material 1 for reference
      subroutine vumat_MAT1(
C Read only (unmodifiable)variables -
     1  nblock, ndir, nshr, nstatev, nfieldv, nprops, lanneal,
     2  stepTime, totalTime, dt, cmname, coordMp, charLength,
     3  props, density, strainInc, relSpinInc,
     4  tempOld, stretchOld, defgradOld, fieldOld,
     5  stressOld, stateOld, enerInternOld, enerInelasOld,
     6  tempNew, stretchNew, defgradNew, fieldNew,
C Write only (modifiable) variables -
     7  stressNew, stateNew, enerInternNew, enerInelasNew )
C
      include 'vaba_param.inc'
C
      dimension props(nprops), density(nblock), coordMp(nblock,*),
     1  charLength(nblock), strainInc(nblock,ndir+nshr),
     2  relSpinInc(nblock,nshr), tempOld(nblock),
     3  stretchOld(nblock,ndir+nshr),
     4  defgradOld(nblock,ndir+nshr+nshr),
     5  fieldOld(nblock,nfieldv), stressOld(nblock,ndir+nshr),
     6  stateOld(nblock,nstatev), enerInternOld(nblock),
     7  enerInelasOld(nblock), tempNew(nblock),
     8  stretchNew(nblock,ndir+nshr),
     9  defgradNew(nblock,ndir+nshr+nshr),
     1  fieldNew(nblock,nfieldv),
     1  stressNew(nblock,ndir+nshr), stateNew(nblock,nstatev),
     2  enerInternNew(nblock), enerInelasNew(nblock)
C
C
      character*80 cmname
      E = props(1)*1.0d0
      SY = props(2)*1.0d0
      Gf = props(3)*1.0d0
      nu = props(4)*1.0d0
      epsy= SY/E
      G = E/(2.0d0*(1.0d0+nu))
```

(a) Definition of the first material

```
      do 100 km = 1,nblock
C! update ---------------------------------------------------------------
        dispt_max = stateOld(km,1)
        dispc_max = stateOld(km,2)
        disp_old = stateOld(km,3)
        i_step = stateOld(km,4)
        h_new = charLength(km)
        damt = stateOld(km,6)
        damc = stateOld(km,7)

        epst_max = stateOld(km,9)
        epsc_max = stateOld(km,10)
        eps_old = stateOld(km,11)

        if (i_step .le. 0.0d0) then
            h = h_new
        else
            h = stateOld(km,5)
        end if
        dispy = epsy*h
        eps = eps_old+strainInc(km,1)

C!  Stress calculation
```

(b) Variables updating

Figure B.2: Fortran script for cohesive law implementation-2

```
C!  Stress calculation
C       !Loading Case
        if (abs(eps) .ge. epst_max .AND. eps .ge. 0.0d0) then
                epst_max = abs(eps)
                dispt_max = h*(exp(epst_max)-1.0d0)
        end if

        if (abs(eps) .ge. epsc_max .AND. eps .lt. 0.0d0) then
                epsc_max = abs(eps)
                dispc_max = h*(exp(epsc_max)-1.0d0)
        end if
C       !Unloading Case
        disp = h*(exp(abs(eps))-1.0d0)

        if (abs(eps) .lt. 1.0d-14) then
            sig1 = 0.0d0
        end if
C       !Tension
        if (eps .gt. 0.0d0) then
            if (epst_max .le. epsy) then
            fteps = E*epst_max
            sig1 = fteps/epst_max*eps
            else
                fteps = max(0.0d0,
     1 SY*(dispt_max-2.0d0*Gf/SY)/(SY*h/E-2.0d0*Gf/SY))!max(0.0d0,SY+Es*(epst_max-epsy))

                sig1 = fteps/dispt_max*disp
            end if

            damt = 1.0d0 - (fteps*h/dispt_max)/E
        end if
C       !Compression
        if (eps .lt. 0.0d0) then
            if (epsc_max .le. epsy) then
            fceps = E*epsc_max
            sig1 = fceps/epsc_max*eps
            else
                fceps = max(0.0d0,
     2 SY*(dispc_max-2.0d0*Gf/SY)/(SY*h/E-2.0d0*Gf/SY)) !SY+Es*(epsc_max-epsy))

                sig1 = -fceps/dispc_max*disp
            end if

            damc = 1.0d0 - (fceps*h/dispc_max)/E
        end if


C! Save ----------------------------------------------------
```

(a) Stress calculation

```
C! Save ----------------------------------------------------
        stressNew(km,1) = sig1

        stateNew(km,1) = dispt_max
        stateNew(km,2) = dispc_max
        stateNew(km,3) = disp
        stateNew(km,5) = h
        stateNew(km,4) = i_step+1
        stateNew(km,6) = damt
        stateNew(km,7) = damc
        stateNew(km,8) = h_new

        stateNew(km,9) = epst_max
        stateNew(km,10) = epsc_max
        stateNew(km,11) = eps

!C Update the specific internal energy
        enerInternNew(km) = enerInternOld(km)
     1    + (sig1+stressOld(km,1))*strainInc(km,1)/(2.0d0*density(km))

100   continue

      return
      end

C! Material 2 for parametric study
```

(b) Variable saving

Figure B.3: Fortran script for cohesive law implementation-3; Material 2 is defined in the same argument as material 1 for parametric study in the future.

Appendix C

PYTHON SCRIPT FOR WRITING CONSTRAINT EQUATIONS INTO INPUT FILE

```python
import numpy as np
import math
import os
import shutil
import re

# Pre-process
for name in ['NOTriangle_Hexagon']:
    for scale in [1,2,4,8,16]:
        for elemNum in [10]:
            for id in ['Light4CSmall','Light4CNo']:
                dir= name+str(scale)+id+'_'\
                +'ratio1'+'_'+'elem'+str(elemNum)+'_nu00'
                inp_dir= dir
                tempinp_dir= inp_dir+".txt"
                gol_dir= dir.replace('NO','')+'.inp'
                shutil.copy(inp_dir+'.inp',tempinp_dir)
                #rewrite *.inp file
                w_str = ""
                w_set1 = ""
                w_set2 = ""
                w_eq = ""
                hex = []
                tri = []
                hvex = []
                tvex = []
                with open(tempinp_dir,'r') as r:
```

(a) Preparation for re-write a input file

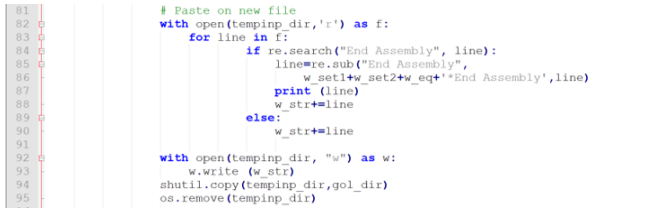
```python
                with open(tempinp_dir,'r') as r:
                    # Extract vertices
                    lines=r.readlines()
                    for i in range(len(lines)):
                        if re.search('Nset,
                            nset=Hexagon_AllVertices,
                            generate',lines[i]):
                            hvex = [int(num) for num in lines[i+1].split(',')]
                        if re.search('Nset,
                            nset=Triangle_AllVertices,
                            generate',lines[i]):
                            tvex = [int(num) for num in lines[i+1].split(',')]
                        if re.search('Part, name=Hexagon', lines[i]):
                            for line in lines[i+2:]:
                                if line.strip()!= "*Element, type=B21":
                                    data = [float(num) for num in line.split(',')]
                                    data[0]=int(data[0])
                                    hex.append(data)
                                else:
                                    break;
                        elif re.search('Part, name=Triangle', lines[i]):
                            for line in lines[i+2:]:
                                if line.strip()!= "*Element, type=B21":
                                    data = [float(num) for num in line.split(',')]
                                    data[0]=int(data[0])
                                    tri.append(data)
                                else:
                                    break;
                    # Make constraint equations
```

(b) Extract vertices

Figure C.1: Python script for writing constraint equations into input file-1

```
            # Make constraint equations
            n = 0
            dist_x = 2*cos(pi/6.0)
            dist_y = 2*sin(pi/6.0)
            for j in range(hvex[1]+1):
                for k in range(tvex[1]+1):
                    if sqrt((hex[j][1]+dist_x-tri[k][1])**2+\
                    (hex[j][2]+dist_y-tri[k][2])**2)<0.0001:
                        n = n+1
                        if n ==1:
                            w_set1 = w_set1+'Nset, nset=Hexagon_'+\
                            str(n)+', Instance=Hexagon\n'+' '+\
                            str(hex[j][0])+',\n'
                        else:
                            w_set1 = w_set1+'*Nset, nset=Hexagon_'+\
                            str(n)+', Instance=Hexagon\n'+' '+\
                            str(hex[j][0])+',\n'
                        w_set2 = w_set2+'*Nset, nset=Triangle_'+str(n)+\
                        ', Instance=Triangle\n'+' '+str(tri[k][0])+',\n'
                        w_eq = w_eq+'** Constraint: Const_'+str(n)+'-1\n'+\
                        '*Equation\n'+'2\n'+'Hexagon_'+str(n)+', 1, -1.\n'+\
                        'Triangle_'+str(n)+', 1, 1.\n'
                        w_eq = w_eq+'** Constraint: Const_'+str(n)+\
                        '-2\n'+'*Equation\n'+'2\n'+'Hexagon_'+str(n)+\
                        ', 2, -1.\n'+'Triangle_'+str(n)+', 2, 1.\n'
        r.close()
        # Paste on new file
```

(a) Make constraint equations



```
        # Paste on new file
        with open(tempinp_dir,'r') as f:
            for line in f:
                if re.search("End Assembly", line):
                    line=re.sub("End Assembly",
                        w_set1+w_set2+w_eq+'*End Assembly',line)
                    print (line)
                    w_str+=line
                else:
                    w_str+=line

        with open(tempinp_dir, "w") as w:
            w.write (w_str)
        shutil.copy(tempinp_dir,gol_dir)
        os.remove(tempinp_dir)
```

(b) Paste on new file

Figure C.2: Python script for writing constraint equations into input file-2

Appendix D

PYTHON SCRIPT FOR UCM GENERATION

```
# -*- coding: mbcs -*-
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
from abaqusConstants import *
import numpy as np

class Lattice:
class Triangle(Lattice):
 ########################
class Hexagon(Lattice):
 ########################
class Diamond(Lattice):
 ########################
class Kagome(Lattice):
 ########################
class Parallelogram(Lattice):
 ########################
def CreateSection (nu_eff,G,t,SectionName,Material,ProfileName):
 ########################
def Mesh(Specimen_Name, L, ele_Number =1, D =2):
 ########################
def MixLattice (BaseName, PlugName, D=2):
 ########################
def CutSpecimen (Specimen, x1, y1, x2, y2):
 ########################
def DoTensileTest (Specimen, mix, dir, strain):
 # Save by whli56 on 2018_10_12-15.40.36; build 2018 2017_11_07-09.21.41 127140
```

(a) Class and definition for lattices

```
# Save by whli56 on 2018_10_12-15.40.36; build 2018 2017_11_07-09.21.41 127140

# Main Codes
Para = 'Parallelogram'
Diam = 'Diamond'
Tri = 'Triangle'
Kago = 'Kagome'
Hex = 'Hexagon'
Bar = 'Bar'

# Set Parameter
E = 60000.0
e_y = 0.001
Sy = E*e_y
FEng = 1.0
nu = 0.3
G = E/(2*(1+nu))
rho_bar = 1.0/10.0  # relative density
mat1 = [E, Sy, FEng, nu]
ratio = 1 # ratio = Gf2:Gf1
mat2 = [E, Sy, FEng*ratio, nu]

scale  = 2
Base = Triangle(1,0,pi/3.0,Tri)
Base.draw2(20*scale,10*scale)
Plug = Hexagon (2,1,pi/6.0,Hex)
Plug.draw(6*scale,4*scale)

size_scale = 16
Base.scaleUpCellNumber (Base.getWidth(),Base.getHight(),size_scale)
Plug.scaleUpCellNumber (Plug.getWidth()-Plug.L*cos(Plug.theta),Plug.getHight()-Plug.L*sin(Plug.theta),size_scale)

# Cut
```

(b) Lattice selection and geometry generation

Figure D.1: Python script for UCM generation-1

```
# Cut
dist_x = Plug.L*cos(Plug.theta)
dist_y = Plug.L*sin(Plug.theta)
Base.cutSpecimen (dist_x, 0.0, Plug.getWidth(), Plug.getHight()-Plug.L*(sin(Plug.theta)+1)) # Base goes first as this might change origina
Plug.cutSpecimen (0.0, -dist_y, Plug.getWidth()-dist_x, -dist_y+Plug.getHight()-Plug.L*(sin(Plug.theta)+1))

# crack profile
crack = True
if crack == True:
else:

# Simplify commands
model = mdb.models['Model-1']
assembly = model.rootAssembly

# Make a patch
patch = False
if patch == True:
else:
print "Finished Drawing"

keep = True # Keep doing pre-process or not
mix = True  # Couple two lattice or not

if keep == True:

    # Create Materials
    model.Material(name='MAT1')
    model.materials['MAT1'].Density(table=((2.8e-07, ), ))
    model.materials['MAT1'].Depvar(deleteVar=12, n=12)
    model.materials['MAT1'].UserMaterial(mechanicalConstants=tuple(mat1))

    model.Material(name='MAT2')
    model.materials['MAT2'].Density(table=((2.8e-07, ), ))
    model.materials['MAT2'].Depvar(deleteVar=12, n=12)
    model.materials['MAT2'].UserMaterial(mechanicalConstants=tuple(mat2))

    # Create profile for Beam section
```

(a) Crack generation and material

```
    # Create profile for Beam section
    weight_factor = 4
    t_Base = Base.getThickness(rho_bar)
    t_Plug = Plug.getThickness(rho_bar)/(weight_factor*1.0)
    model.RectangularProfile(a=t_Base, b=t_Base, name='Profile-1-Base')
    model.RectangularProfile(a=t_Plug, b=t_Plug, name='Profile-1-Plug')
    model.CircularProfile(name='Profile-2-Base', r=t_Base)
    model.CircularProfile(name='Profile-2-Plug', r=t_Plug)

    # Create Section
    nu_eff = 0.0
    CreateSection (nu_eff,G,t = t_Base, SectionName = 'Section-Base',Material = 'MAT1',ProfileName = 'Profile-1-Base')
    CreateSection (nu_eff,G,t = t_Plug, SectionName = 'Section-Plug',Material = 'MAT2',ProfileName = 'Profile-1-Plug')

    # Assign Sections
    Base.assignSection('Section-Base')
    Plug.assignSection('Section-Plug')

    # Assign Orientation (if beam element)
    Base.assignOrientation()
    Plug.assignOrientation()

    # Generate Instances
    assembly.DatumCsysByDefault(CARTESIAN)
    Base.generateInst()
    Plug.generateInst()

    # Create step
    model.ExplicitDynamicsStep(improvedDtMethod=ON, massScaling=((SEMI_AUTOMATIC, MODEL, AT_BEGINNING, 5.0, 0.0, None, 0, 0,

    # Set output data-files
    model.fieldOutputRequests['F-Output-1'].setValues(numIntervals=200)
    model.fieldOutputRequests['F-Output-1'].setValues(variables=(
    # Set Amplitudes
    model.SmoothStepAmplitude(data=((0.0, 0.0), (0.1, 1.0)), name=

    # Set BC's
    DoTensileTest (Plug, mix, 'Y', 0.005)

    # Translate (if mix is true)
    dist_x = Plug.L*cos(Plug.theta)
    dist_y = Plug.L*sin(Plug.theta)
    assembly.instances[Plug.Part_Name].translate((dist_x,dist_y ,0))

    # Mesh by size (= strut length over elementnuber)
```

(b) Element sections and BC's

```
    # Mesh by size (= strut length over elementnuber)
    for mesh_num in [10]:
        Base.mesh(mesh_num, 2)
        Plug.mesh(mesh_num, 2)
        print "Meshed"
        totalscale = scale*size_scale
        # Submit Job
        if mix ==True:
        else:

        mdb.Job(activateLoadBalancing=False, atTime=None, contactPrint=OFF,
                description='', echoPrint=OFF, explicitPrecision=DOUBLE_PLUS_PACK,
                historyPrint=OFF, memory=60, memoryUnits=PERCENTAGE, model='Model-1',
                modelPrint=OFF, multiprocessingMode=MPI, name=na,
                nodalOutputPrecision=FULL, numCpus=16, numDomains=16,
                parallelizationMethodExplicit=DOMAIN, queue=None, resultsFormat=ODB,
                scratch='', type=ANALYSIS,
                userSubroutine='linearsoft5.f',
                waitHours=0, waitMinutes=0)

        mdb.jobs[na].writeInput(consistencyChecking=OFF)
```

(c) Mesh and input file generation

Figure D.2: Python script for UCM generation-2

Appendix E

DATA OF SIMULATION RESULTS IN CHAPTER 4

Relative rho ratio $\bar{\rho}_{tri}/\bar{\rho}_{Hex}$	W_{hex}/W_{total}	Total mass (kg)	Specific Peak load (N/kg)	Strength/ Sy	Specific toughness	Drop of Strength/Sy %	Drop of Specific peak load	Increase of specific toughness %
1	86.20%	5.57E-07	8.28E+06	1.37E-02	2.44E+05	59.96%	61%	2429.76%
3	41.00%	1.30E-07	1.93E+07	1.75E-02	1.87E+05	49.07%	10%	1840.07%
4	28.00%	1.07E-07	2.14E+07	1.91E-02	2.37E+05	44.34%	0%	2360.46%
5	20.00%	9.62E-08	2.10E+07	1.91E-02	1.84E+05	44.37%	2%	1812.47%
10	5.87%	8.18E-08	2.09E+07	2.22E-02	5.19E+04	35.18%	2%	438.63%
inf	0.00%	7.70E-08	2.14E+07	3.43E-02	9.64E+03	0.00%	0%	0.00%

Figure E.1: A sheet recording data collected from parametric study.

	Triangular					
Size	D (mm)	Mass (kg)	E (Mpa)	Specific Peak force	Specific toughness	Ultimate Strength/Sy
scale-1	13.86	1.83E-08	2.12E+03	4.64E+07	9.81E+03	3.54E-02
scale-2	27.71	7.70E-08	2.06E+03	2.14E+07	9.64E+03	3.43E-02
scale-4	55.43	3.15E-07	2.03E+03	1.03E+07	9.25E+03	3.38E-02
scale-8	110.85	1.27E-06	2.01E+03	5.06E+06	8.66E+03	3.36E-02
scale-16	221.70	5.13E-06	2.00E+03	2.50E+06	7.25E+03	3.35E-02

Figure E.2: A sheet recording data collected from the triangular lattice (NCP).

	Hexagonal						
Size	D (mm)	Mass (kg)	E (Mpa)	Specific Peak force	Specific toughness	Ultimate Strength/Sy	Yield strength/Sy
scale-1	13.86	7.46E-09	1.09E+00	3.78E+07	6.39E+06	7.82E-03	3.02E-04
scale-2	27.71	3.00E-08	1.25E+00	1.78E+07	5.82E+06	7.43E-03	3.15E-04
scale-4	55.43	1.21E-07	1.34E+00	8.25E+06	6.19E+06	6.90E-03	3.23E-04
scale-8	110.85	4.83E-07	1.38E+00	4.37E+06	6.40E+06	7.32E-03	3.27E-04
scale-16	221.70	1.93E-06	1.40E+00	2.25E+06	6.51E+06	7.56E-03	3.28E-04

Figure E.3: A sheet recording data collected from the hexagonal lattice (NCP).

	UCM					
Size	D (mm)	Mass (kg)	E (Mpa)	Specific Peak force	Specific toughness	Ultimate Strength/Sy
scale-1	13.86	2.57E-08	1.06E+03	4.38E+07	3.57E+05	1.88E-02
scale-2	27.71	1.07E-07	1.03E+03	2.14E+07	2.37E+05	1.91E-02
scale-4	55.43	4.36E-07	1.02E+03	1.05E+07	2.23E+05	1.90E-02
scale-8	110.85	1.76E-06	1.02E+03	5.16E+06	2.26E+05	1.89E-02
scale-16	221.70	7.07E-06	1.01E+03	2.57E+06	2.24E+05	1.89E-02

Figure E.4: A sheet recording data collected from the UCMs (NCP).

Size\Lattice	Tri	Hex	UCM
scale-1	0.0249	0.0054	0.0134
scale-2	0.0241	0.0041	0.0130
scale-4	0.0220	0.0037	0.0126
scale-8	0.0190	0.0031	0.0105
scale-16	0.0160	0.0029	0.0080

Figure E.5: A sheet recording data collected for the type II size effect (CCP).

Appendix F

COMPLIANT METHOD TO APPROXIMATE g_0 AND g_0'

In general, it is common to use the J-integral approach to find the dimensionless energy release rate g_0 [93]. This method is feasible for the continuum and homogeneous materials. However, the lattice structure is not the continuum at the finite size. Although the lattice can be approached as the continuum when the size of the entire structure is scaled to be large enough, still it will cost too much in memory resources and in time to compute the solution. Therefore, the compliant method is applied to calculate g_0 for the lattice structure. Recalling the relationship between the elastic potential Π^* and the energy release rate $\mathcal{G}$, the equation is

$$\mathcal{G} = \mathcal{G}(u, a) = \frac{1}{b}\left[\frac{\partial \Pi^*(P, a)}{\partial a}\right]_P \tag{F.1}$$

, where u is the equilibrium displacement, b is the thickness of the specimen, a is the crack length and P is the external load. Knowing that $\Pi^*(P, a) = \frac{1}{2}C(a)P$ with the compliant stiffness C, Eq. (A.1) can be written as

$$G = \frac{P^2}{2b}\left[\frac{dc}{da}\right]_P \tag{F.2}$$

By using the central difference approximation, Eq. (A.2) can be approximated as

$$\mathcal{G} = \frac{P^2}{2b}\left[\frac{C(a + \Delta a) - C(a - \Delta a)}{2\Delta a}\right] \tag{F.3}$$

This is the so-called compliant method or incremental stiffness method. To calculate $C(a)$, some simulations with different lengths of central crack are needed. Since a crack in the lattice is not continuous, it can be expressed as a function of the size of a unit cell ℓ, $a = f(\ell)$. Accordingly, the smallest incremental crack length is about the size of a unit cell as shown in Figure 4.12. It is noted that, the incremental crack length of UCM is governed by the lattice of the smallest length scale.

Now, assuming $a = a_0$, where a_0 is the original crack length, the relation between G_0 and the dimensionless energy release rate g_0 is

$$g_0 = g(\alpha_0) = \frac{EG}{D\sigma_N} \tag{F.4}$$

, where E is the stiffness, D is the width of entire structure, σ_N is the nominal stress and $\alpha_0 = a_0/D$. To calculate the derivative of g_0 with respect to a, the forward difference is used. This gives

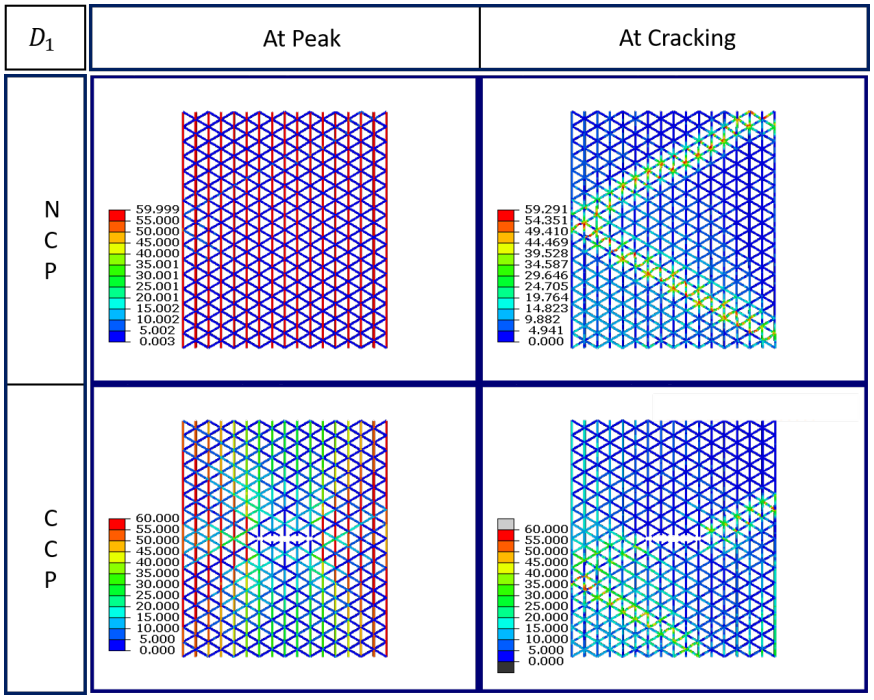
$$g_0' = g'(\alpha_0) = \frac{g(\alpha_0 + \Delta\alpha) - g(\alpha_0)}{\Delta\alpha_0} \tag{F.5}$$

Both the values of g_0 and g_0' are calculated. Additionally, since this method requires a certain number of cells in the lattice to be cut, the size of the entire lattice structure has to be large enough. As a consequence, the second small central-cracked plate is used. All lattices here are made of pure, leaner elastic material with Young's Modulus $E = 60,000$ (MPa) used to compute the elastic potential.

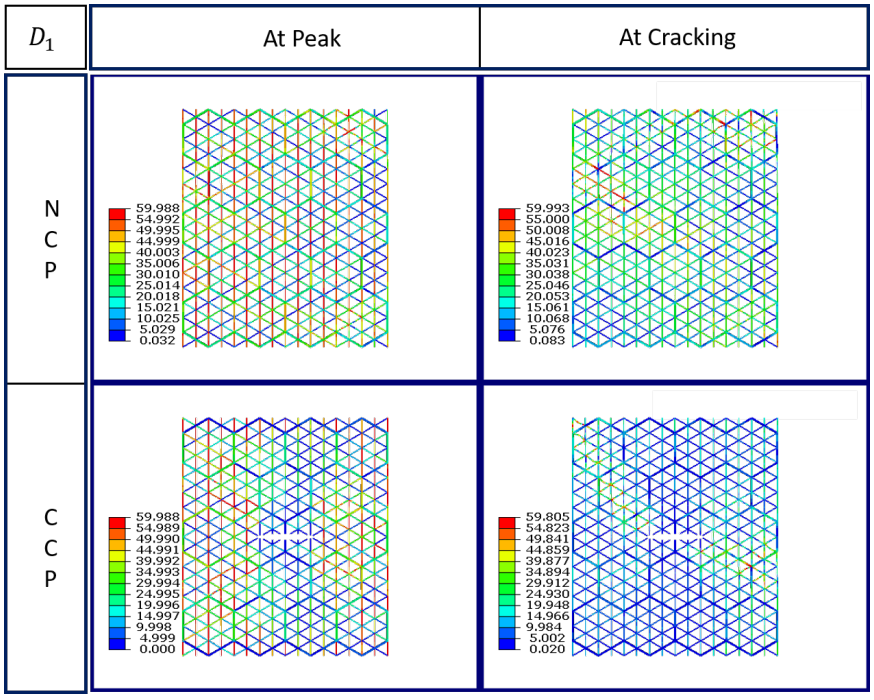
Appendix G

CONTOUR PLOTS OF SIMULATION RESULTS FROM CHAPTER 4

The provided FE simulations of non-creacked plate (NCP) and center-cracked plate (CCP) of the triangular lattice and the UCM display the crack propagation. All the simulation results are presented as contour plots using Von-Mises stress for crack visualization because this stress is related to deformation energy.

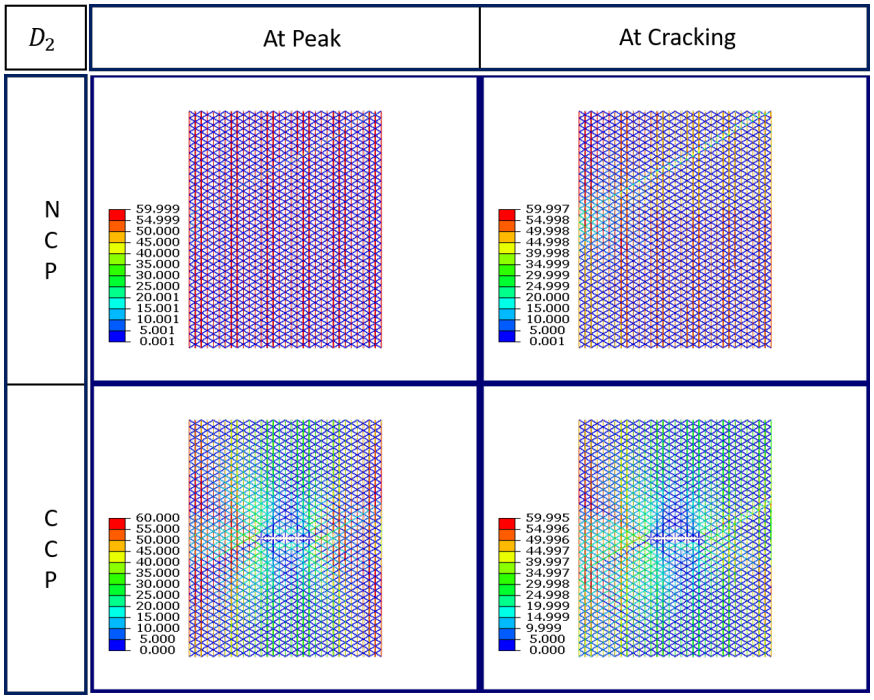


(a) Triangular lattice

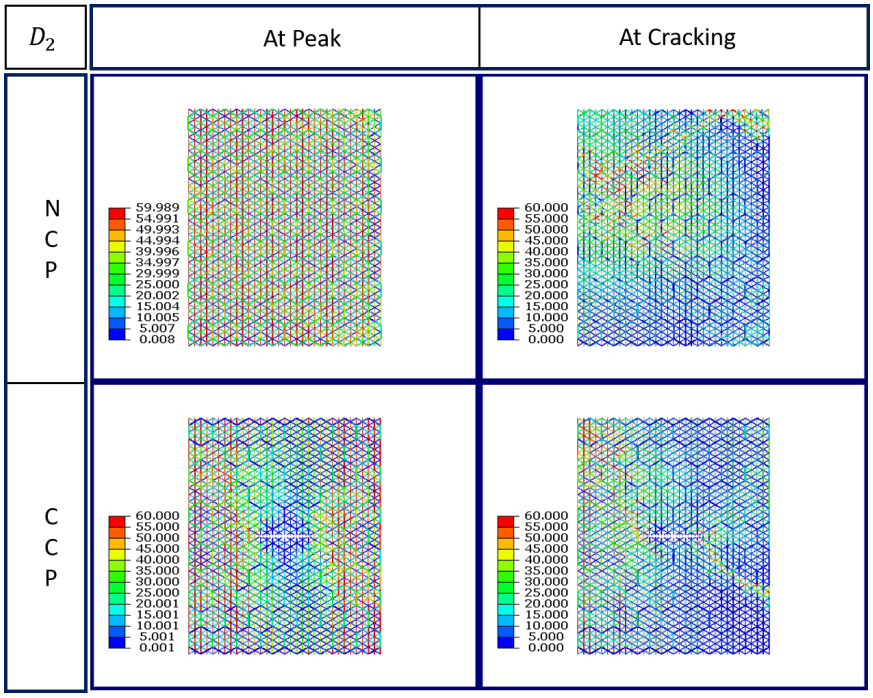


(b) UCM

Figure G.1: The contour plots for NCP and CCP at peak and at cracking-1

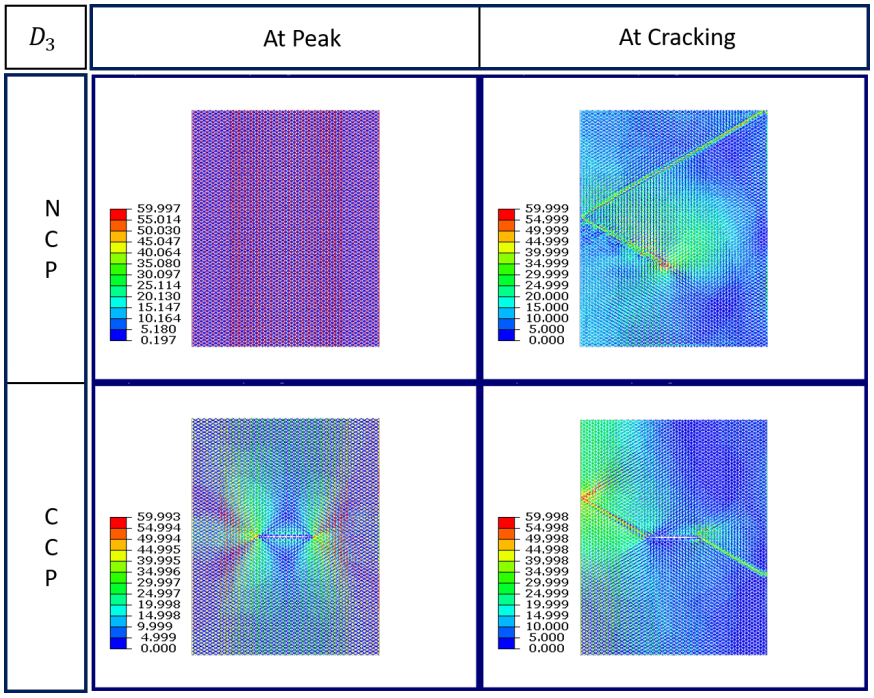


(a) Triangular lattice

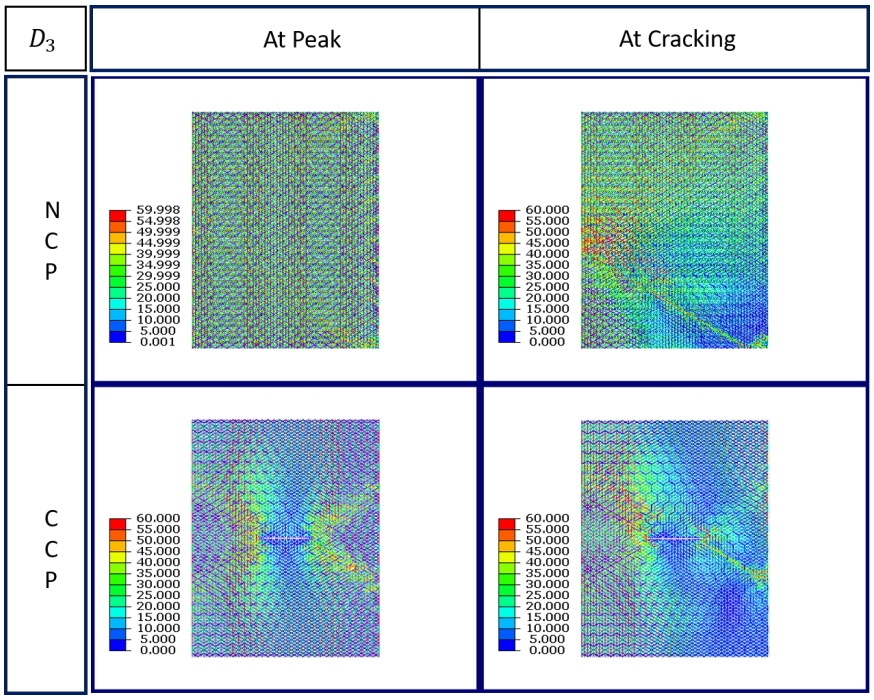


(b) UCM

Figure G.2: The contour plots for NCP and CCP at peak and at cracking-2

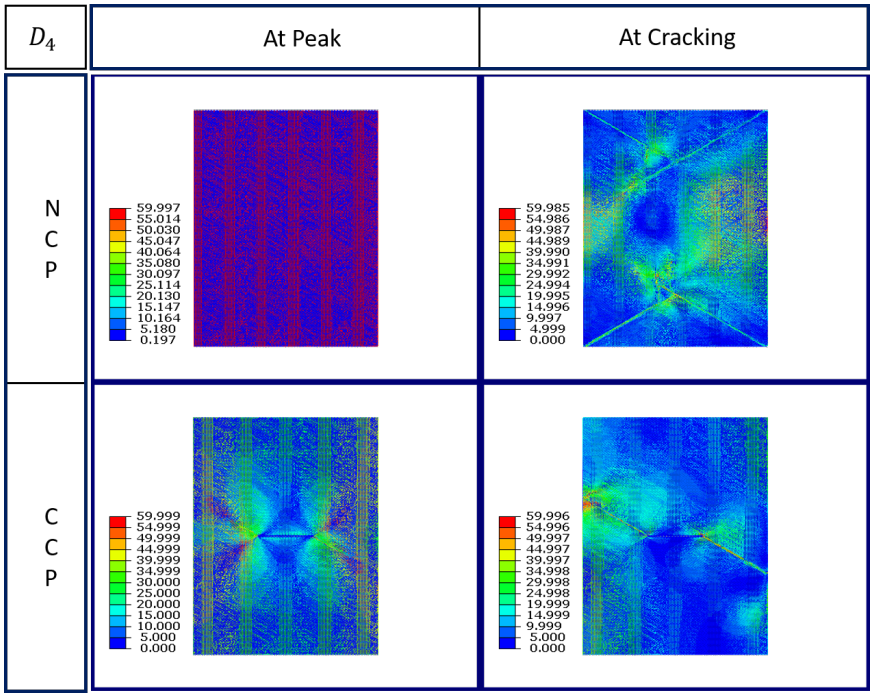


(a) Triangular lattice

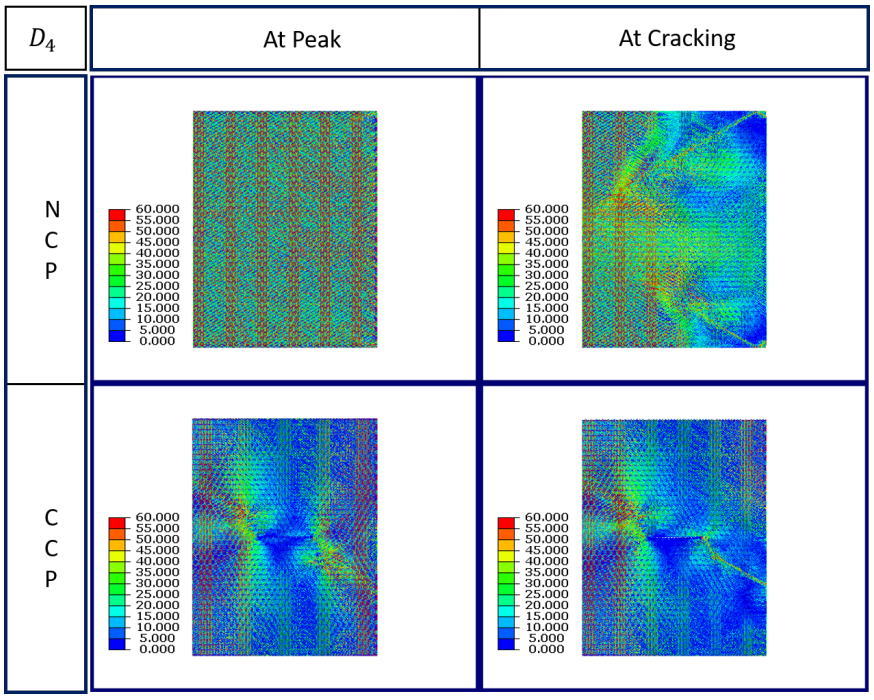


(b) UCM

Figure G.3: The contour plots for NCP and CCP at peak and at cracking-3



(a) Triangular lattice



(b) UCM

Figure G.4: The contour plots for NCP and CCP at peak and at cracking-4

Appendix H

CLOSED FORM OF THE FRACTURE ENERGY PREDICTION

Assume that the fracture energy of the lattice G_f can account for the fracture energy of the cell-wall material G_{fs}. This gives

$$AG_f = \beta n_f G_{fs} A_s \tag{H.1}$$

, where n_f is the number of broken strut, β is a constant depending on topology, and A is the cross-section. This can be expressed in terms of $\bar{\rho}$ as

$$G_f = \beta n_f G_{fs} \frac{A_s}{A} = \beta n_f G_{fs} \bar{\rho} \tag{H.2}$$

Given that $a = f(n_f, \ell)$, the n_f has a relation $n_f = f(a, \ell) = q\frac{a}{\ell}$ where ℓ is strut length. Plugging this into Eq. B.1, gives

$$G_f = HG_{fs}\bar{\rho}\frac{a}{\ell} \tag{H.3}$$

, where H is a constant depending on the topology of unit cell. Recalling Eq. 4.3, the high order terms can be neglected. This leads to a relation

$$g(\alpha) \approx g_0 + g_0' \approx \pi \frac{a + c_f}{D} \tag{H.4}$$

with $g_0 = \pi\alpha$ and $g_0' = \pi$. The definition of g is

$$g = \frac{EG}{D\sigma_f^2} \tag{H.5}$$

Let $G = G_f$ and upon making use of Eq. (2.5), Eq. (2.6) and Eq. (B.3), the Eq. (B.5) is rewritten as

$$g = \frac{B\bar{\rho}^b E_s}{DC^2\bar{\rho}^c\sigma_{fs}^2} HG_{fs}\frac{a}{\ell}\bar{\rho} \tag{H.6}$$

Substituting H in Eq. (B.3), the result is

$$G_f = \frac{C^2}{B}\bar{\rho}^{2c-b}\pi(a + c_f)\frac{\sigma_{fs}^2}{E_s} \tag{H.7}$$

The constants and c_f can be found in Table 2.1 and Table 4.2 respectively. Assuming $a = a_0 = 0.15D$ from analysis in Chapter 4, the analytical predictions for lattices of quasi-brittle material can be calculated.

BIBLIOGRAPHY

[1] Qiancheng Zhang, Xiaohu Yang, Peng Li, Guoyou Huang, Shangsheng Feng, Cheng Shen, Bin Han, Xiaohui Zhang, Feng Jin, Feng Xu, et al. Bioinspired engineering of honeycomb structure–using nature to inspire human innovation. *Progress in Materials Science*, 74:332–400, 2015.

[2] Granta design ltd., https://grantadesign.com/education/.

[3] Hrl.

[4] Lucas R Meza, Satyajit Das, and Julia R Greer. Strong, lightweight, and recoverable three-dimensional ceramic nanolattices. *Science*, 345(6202):1322–1326, 2014.

[5] Jens Bauer, Almut Schroer, Ruth Schwaiger, and Oliver Kraft. Approaching theoretical strength in glassy carbon nanolattices. *Nature materials*, 15(4):438, 2016.

[6] Chan Soo Ha, Michael E Plesha, and Roderic S Lakes. Chiral three-dimensional isotropic lattices with negative poisson's ratio. *physica status solidi (b)*, 253(7):1243–1251, 2016.

[7] Minh-Son Pham, Chen Liu, Iain Todd, and Jedsada Lertthanasarn. Damage-tolerant architected materials inspired by crystal microstructure. *Nature*, 565(7739):305, 2019.

[8] Davis recycling inc., http://davisconverters.com/.

[9] JE Shafizadeh, JC Seferis, EF Chesmar, and R Geyer. Evaluation of the in-service performance behavior of honeycomb composite sandwich structures. *Journal of Materials Engineering and Performance*, 8(6):661–668, 1999.

[10] Bbc news inc., url=https://www.bbc.com/news url=https://www.bbc.com/news.

[11] NA Fleck, VS Deshpande, and MF Ashby. Micro-architectured materials: past, present and future. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 466(2121):2495–2516, 2010.

[12] Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of computational physics*, 117(1):1–19, 1995.

[13] Textilelearner., https://textilelearner.blogspot.com/2011/03/definition-and-characteristics-of-plain_4390.html.

[14] Pksupplyonline., https://www.pksupplyonline.com.

[15] Selin Hanife Eryuruk and Fatma Kalaoğlu. The effect of weave construction on tear strength of woven fabrics. *Autex Research Journal*, 15(3):207–214, 2015.

[16] Zhao Qin and Markus J Buehler. Impact tolerance in mussel thread networks by heterogeneous material distribution. *Nature communications*, 4:2187, 2013.

[17] Akhlesh Lakhtakia and Raúl José Martín-Palma. *Engineered biomimicry*. Newnes, 2013.

[18] Horacio D Espinosa, Jee E Rim, Francois Barthelat, and Markus J Buehler. Merger of structure and material in nacre and bone–perspectives on de novo biomimetic materials. *Progress in Materials Science*, 54(8):1059–1100, 2009.

[19] HC Tankasala, VS Deshpande, and NA Fleck. Tensile response of elastoplastic lattices at finite strain. *Journal of the Mechanics and Physics of Solids*, 109:307–330, 2017.

[20] Winston O Soboyejo and TS Srivatsan. *Advanced structural materials: properties, design optimization, and applications*. CRC press, 2006.

[21] O Van der Sluis, PJG Schreurs, WAM Brekelmans, and HEH Meijer. Overall behaviour of heterogeneous elastoviscoplastic materials: effect of microstructural modelling. *Mechanics of Materials*, 32(8):449–462, 2000.

[22] Alfredo E Huespe and Javier Oliver. Crack models with embedded discontinuities. In *Numerical modeling of concrete cracking*, pages 99–159. Springer, 2011.

[23] Lorna J Gibson and Michael F Ashby. *Cellular solids: structure and properties*. Cambridge university press, 1999.

[24] Naomi ER Romijn and Norman A Fleck. The fracture toughness of planar lattices: Imperfection sensitivity. *Journal of the Mechanics and Physics of Solids*, 55(12):2538–2564, 2007.

[25] P Maimí, A Turon, and D Trias. Crack propagation in quasi-brittle two-dimensional isotropic lattices. *Engineering Fracture Mechanics*, 78(1):60–70, 2011.

[26] Ignacio Quintana-Alonso and Norman A Fleck. Fracture of brittle lattice materials: A review. In *Major accomplishments in composite materials and sandwich structures*, pages 799–816. Springer, 2009.

[27] Michael F Ashby, Tony Evans, Norman A Fleck, JW Hutchinson, HNG Wadley, and LJ Gibson. *Metal foams: a design guide*. Elsevier, 2000.

[28] LJ Gibson, MF Ashby, J Zhang, and TC Triantafillou. Failure surfaces for cellular materials under multiaxial loadsi. modelling. *International Journal of Mechanical Sciences*, 31(9):635–663, 1989.

[29] Rasto Brezny and David J Green. The effect of cell size on the mechanical behavior of cellular materials. *Acta metallurgica et materialia*, 38(12):2517–2526, 1990.

[30] VV Vasiliev, VA Barynin, and AF Rasin. Anisogrid lattice structures–survey of development and application. *Composite structures*, 54(2-3):361–370, 2001.

[31] Valery V Vasiliev, Vyacheslav A Barynin, and Alexander F Razin. Anisogrid composite lattice structures–development and aerospace applications. *Composite structures*, 94(3):1117–1127, 2012.

[32] MF Ashby. The properties of foams and lattices. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 364(1838):15–30, 2005.

[33] Lutz-Christian Gerhardt and Aldo R Boccaccini. Bioactive glass and glass-ceramic scaffolds for bone tissue engineering. *Materials*, 3(7):3867–3910, 2010.

[34] Hiroshi Itoh, Yu Aso, Masayasu Furuse, Yasuharu Noishiki, and Teruo Miyata. A honeycomb collagen carrier for cell culture as a tissue engineering scaffold. *Artificial Organs*, 25(3):213–217, 2001.

[35] I Quintana Alonso and NA Fleck. Damage tolerance of an elastic-brittle diamond-celled honeycomb. *Scripta Materialia*, 56(8):693–696, 2007.

[36] R Hooke. 1664. micrographia. london. *UK: Royal Society*.

[37] NJ Hoff. Structural problems of future aircraft. In *Proceedings of the Third Anglo-American Aeronautical Conference*, pages 77–114, 1951.

[38] D'Arcy Wentworth Thompson. *On Growth and Form by D'Arcy Wentworth Thompson*. At the University Press, 1968.

[39] Lorna J Gibson, Michael F Ashby, and Brendan A Harley. *Cellular materials in nature and medicine*. Cambridge University Press, 2010.

[40] S Vajjhala, AM Kraynik, and LJ Gibson. A cellular solid model for modulus reduction due to resorption of trabeculae in bone. *Journal of biomechanical engineering*, 122(5):511–515, 2000.

[41] Joanna Aizenberg, James C Weaver, Monica S Thanawala, Vikram C Sundar, Daniel E Morse, and Peter Fratzl. Skeleton of euplectella sp.: structural hierarchy from the nanoscale to the macroscale. *Science*, 309(5732):275–278, 2005.

[42] Tobias A Schaedler, Alan J Jacobsen, Anna Torrents, Adam E Sorensen, Jie Lian, Julia R Greer, Lorenzo Valdevit, and Wiliam B Carter. Ultralight metallic microlattices. *Science*, 334(6058):962–965, 2011.

[43] Jong-Shin Huang and LJ Gibson. Fracture toughness of brittle honeycombs. *Acta metallurgica et materialia*, 39(7):1617–1626, 1991.

[44] ST Gulati. Effects of cell geometry on thermal shock resistance of catalytic monoliths. Technical report, SAE Technical Paper, 1975.

[45] HEM Hunt. The mechanical strength of ceramic honeycomb monoliths as determined by simple experiments. *Chemical engineering research & design*, 71:257–266, 1993.

[46] Mudit Rastogi. Moisture ingress in honeycomb sandwich composite structures effects detection. *International Journal of Engineering Research And V5, no. 11 (August 2016).*

[47] Holm Altenbach and Andreas Öchsner. *Cellular and porous materials in structures and processes*, volume 521. Springer Science & Business Media, 2011.

[48] SK Maiti, MF Ashby, and LJ Gibson. Fracture toughness of brittle cellular solids. *Scripta Metallurgica*, 18(3):213–217, 1984.

[49] LJ Gibson, MF Ashby, J Zhang, and TC Triantafillou. Failure surfaces for cellular materials under multiaxial loadsi. modelling. *International Journal of Mechanical Sciences*, 31(9):635–663, 1989.

[50] Jong-Shin Huang and MS Chiang. Effects of microstructure, specimen and loading geometries on kic of brittle honeycombs. *Engineering fracture mechanics*, 54(6):813–821, 1996.

[51] John E Srawley. Wide range stress intensity factor expressions for astm e 399 standard fracture toughness specimens. *International Journal of Fracture*, 12(3):475–476, 1976.

[52] JY Chen, Y Huang, and M Ortiz. Fracture analysis of cellular materials: a strain gradient model. *Journal of the Mechanics and Physics of Solids*, 46(5):789–828, 1998.

[53] Norman A Fleck and XinMing Qiu. The damage tolerance of elastic–brittle, two-dimensional isotropic lattices. *Journal of the Mechanics and Physics of Solids*, 55(3):562–588, 2007.

[54] Fabian Lipperman, Michael Ryvkin, and Moshe B Fuchs. Fracture toughness of two-dimensional cellular material with periodic microstructure. *International Journal of Fracture*, 146(4):279–290, 2007.

[55] OB Olurin, Norman A Fleck, and Michael F Ashby. Deformation and fracture of aluminium foams. *Materials Science and Engineering: A*, 291(1-2):136–146, 2000.

[56] Huaiyuan Gu, Martyn Pavier, and Anton Shterenlikht. Experimental study of modulus, strength and toughness of 2d triangular lattices. *International Journal of Solids and Structures*, 152:207–216, 2018.

[57] Sukjoo Choi and Bhavani V Sankar. A micromechanical method to predict the fracture toughness of cellular materials. *International journal of solids and structures*, 42(5-6):1797–1817, 2005.

[58] I Christodoulou and PJ Tan. Crack initiation and fracture toughness of random voronoi honeycombs. *Engineering Fracture Mechanics*, 104:140–161, 2013.

[59] Michael Ryvkin and Raz Shraga. Fracture toughness of hierarchical self-similar honeycombs. *International Journal of Solids and Structures*, 152:151–160, 2018.

[60] Sidney Mindess. The fracture process zone in concrete. In *Toughening Mechanisms in Quasi-Brittle Materials*, pages 271–286. Springer, 1991.

[61] C Chen, Norman A Fleck, and TJ Lu. The mode i crack growth resistance of metallic foams. *Journal of the Mechanics and Physics of Solids*, 49(2):231–259, 2001.

[62] KR Mangipudi and PR Onck. Tensile failure of two-dimensional quasi-brittle foams. *International Journal of Solids and Structures*, 49(19-20):2823–2829, 2012.

[63] MF Ashby. Overview no. 92: materials and shape. *Acta metallurgica et materialia*, 39(6):1025–1039, 1991.

[64] Zenon D Konteatis, Anthony E Klon, Jinming Zou, and Siavash Meshkat. Computational approach to de novo discovery of fragment binding for novel protein states. In *Methods in enzymology*, volume 493, pages 357–380. Elsevier, 2011.

[65] M Mehdi S Mousavi, Elisa Paola Ambrosio, Silvia Appendino, F Chen Chen, Alain Favetto, Diego Manfredi, Francesco Pescarmona, and Aurelio Somà. Spacesuits and eva gloves evolution and future trends of extravehicular activity gloves. In *41th International Conference on Environmental Systems, Portland, Oregon, USA, AIAA*, volume 5147, 2011.

[66] Lia Addadi, Derk Joester, Fabio Nudelman, and Steve Weiner. Mollusk shell formation: a source of new concepts for understanding biomineralization processes. *Chemistry–A European Journal*, 12(4):980–987, 2006.

[67] AP Jackson, JFV Vincent, and RM Turner. Comparison of nacre with other ceramic composites. *Journal of Materials Science*, 25(7):3173–3178, 1990.

[68] AP Jackson, Julian FV Vincent, and RM Turner. The mechanical design of nacre. *Proceedings of the Royal society of London. Series B. Biological sciences*, 234(1277):415–440, 1988.

[69] DM Williamson and WG Proud. The conch shell as a model for tougher composites. *International Journal of Materials Engineering Innovation*, 2(2):149–164, 2011.

[70] A Srikantha Phani and Mahmoud I Hussein. *Dynamics of lattice materials*. Wiley Online Library, 2017.

[71] Henry Martyn Cundy and Arthur Percy Rollett. *Mathematical models*. Tarquin, 1981.

[72] Edward Harrington Lockwood and Robert Hugh Macmillan. *Geometric symmetry*. CUP Archive, 1978.

[73] Greg N Frederickson. *Dissections: plane and fancy*. Cambridge University Press, 2003.

[74] I Syozi. Phase transitions and critical phenomena: Transformation of ising models, 1972.

[75] Sangil Hyun and Salvatore Torquato. Optimal and manufacturable two-dimensional, kagome-like cellular solids. *Journal of Materials Research*, 17(1):137–144, 2002.

[76] Robert G Hutchinson and Norman A Fleck. The structural performance of the periodic truss. *Journal of the Mechanics and Physics of Solids*, 54(4):756–782, 2006.

[77] J Clerk Maxwell. L. on the calculation of the equilibrium and stiffness of frames. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 27(182):294–299, 1864.

[78] CR Calladine. Buckminster fuller's tensegrity structures and clerk maxwell's rules for the construction of stiff frames. *International journal of solids and structures*, 14(2):161–172, 1978.

[79] Sergio Pellegrino and Christopher Reuben Calladine. Matrix analysis of statically and kinematically indeterminate frameworks. *International Journal of Solids and Structures*, 22(4):409–428, 1986.

[80] VS Deshpande, MF Ashby, and NA Fleck. Foam topology: bending versus stretching dominated architectures. *Acta materialia*, 49(6):1035–1040, 2001.

[81] A Srikantha Phani and Norman A Fleck. Elastic boundary layers in two-dimensional isotropic lattices. *Journal of Applied Mechanics*, 75(2):021020, 2008.

[82] A-J Wang and DL McDowell. In-plane stiffness and yield strength of periodic metal honeycombs. *Journal of engineering materials and technology*, 126(2):137–156, 2004.

[83] François Cote, Vikram Deshpande, and Norman Fleck. The shear response of metallic square honeycombs. *Journal of Mechanics of Materials and Structures*, 1(7):1281–1299, 2006.

[84] Stefan Liebenstein, Stefan Sandfeld, and Michael Zaiser. Size and disorder effects in elasticity of cellular structures: from discrete models to continuum representations. *International Journal of Solids and Structures*, 146:97–116, 2018.

[85] Zdenek P Bazant. Mechanics of distributed cracking. *Appl. Mech. Rev*, 39(5):675–705, 1986.

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

[87] Wu Xu and Anthony M Waas. Modeling damage growth using the crack band model; effect of different strain measures. *Engineering Fracture Mechanics*, 152:126–138, 2016.

[88] Ronald Aylmer Fisher et al. The design of experiments. *The design of experiments.*, (7th Ed), 1960.

[89] Marcus Yoder, Lonny Thompson, and Joshua Summers. Size effects in lattice-structured cellular materials: edge softening effects. *Journal of materials science*, 54(5):3942–3959, 2019.

[90] PR Onck, EW Andrews, and LJ Gibson. Size effects in ductile cellular solids. part i: modeling. *International Journal of Mechanical Sciences*, 43(3):681–699, 2001.

[91] C Chen and NA Fleck. Size effects in the constrained deformation of metallic foams. *Journal of the Mechanics and Physics of Solids*, 50(5):955–977, 2002.

[92] PE Seiler, HC Tankasala, and NA Fleck. The role of defects in dictating the strength of brittle honeycombs made by rapid prototyping. *Acta Materialia*, 171:190–200, 2019.

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