

6-2018

Numerical Analysis of Fatigue Crack Growth of Low Porosity Auxetic Materials using the Contour J-integral

Garivalde S. Dominguez
Santa Clara University, gdominguez@scu.edu

Follow this and additional works at: https://scholarcommons.scu.edu/mech_mstr



Part of the [Mechanical Engineering Commons](#)

Recommended Citation

Dominguez, Garivalde S., "Numerical Analysis of Fatigue Crack Growth of Low Porosity Auxetic Materials using the Contour J-integral" (2018). *Mechanical Engineering Master's Theses*. 23.
https://scholarcommons.scu.edu/mech_mstr/23

This Thesis is brought to you for free and open access by the Engineering Master's Theses at Scholar Commons. It has been accepted for inclusion in Mechanical Engineering Master's Theses by an authorized administrator of Scholar Commons. For more information, please contact rscroggin@scu.edu.

Santa Clara University
DEPARTMENT OF MECHANICAL ENGINEERING

DATE: SEPTEMBER 5, 2018

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY
SUPERVISION BY

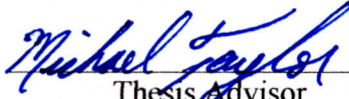
Garivalde Dominguez

ENTITLED

**Numerical Analysis of Fatigue Crack Growth of Low Porosity Auxetic Materials
using the Contour J-integral**

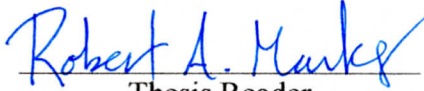
BE ACCEPTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

Master of Science in Mechanical Engineering



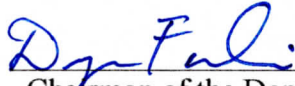
Thesis Advisor

Dr. Michael Taylor



Thesis Reader

Dr. Robert Marks



Chairman of the Department

Dr. Drazen Fabris

Numerical Analysis of Fatigue Crack Growth of Low Porosity Auxetic Materials using the Contour J-integral

by Garivalde S. Dominguez
B.S. Mechanical Engineering
Department of Mechanical Engineering
University of the Philippines

MASTER THESIS

Submitted in Partial Fulfillment of the Requirements
For the Degree of Master of Science
In Mechanical Engineering
In the School of Engineering

at

Santa Clara University

June 2018

Santa Clara, California

DEDICATION

*This work is dedicated to my wife, Justine,
for her wholehearted support and encouragement.*

ACKNOWLEDGEMENT

I would like to express my sincere appreciation to my thesis advisor, Dr. Michael Taylor, for his motivation and support throughout my research. Also, I am thankful for my thesis reader, Dr. Robert Marks, for his patience and advice for the improvement of my work. To the rest of computational solid mechanics group, Luca, Max, and Shawn, thank you for the intellectual discussion and camaraderie.

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENT	iv
TABLE OF FIGURES	vii
NOMENCLARURE	ix
ABSTRACT	xi
CHAPTER I: Introduction	1
CHAPTER II: Fundamentals of Fracture Mechanics	3
2.1 Background	3
2.2 Energy Release Rate	3
2.3 Stress Intensification Factor	3
2.4 Relationship between \mathcal{G} and K_I	6
2.5 Fatigue and Paris Law	6
2.6 J-integral Analytical and Numerical Solution	8
CHAPTER III: Extended Finite Element Method (XFEM)	10
3.1 Background	10
3.2 Partition of Unity	10
3.3 XFEM Enrichment	12
3.4 Solution for Discontinuity	12
3.5 Crack-tip Enrichment	14
3.6 XFEM Discretization	15
CHAPTER IV: Abaqus Implementation	20
4.1 Background	20
4.2 Fracture Criterion	20
4.3 Crack Initiation	21
4.4 Finite Element Solution for J-integral	24

CHAPTER V: Auxetic Structure	26
5.1 Background	26
5.2 Specific Test Sample.....	26
5.3 Periodic Structure.....	26
5.4 Axis Ratio	28
5.5 Porosity	28
CHAPTER VI: Numerical Analysis	29
6.1 Background	29
6.2 Numerical Methods on J-integral.....	29
6.3 Variation of Geometry	34
6.3.1 Constant Porosity.....	34
6.3.2 Constant Minimum Hole Distance.....	40
CHAPTER VII: Results and Discussion.....	46
7.1 Background	46
7.2 Experimental Data and Results.....	46
7.3 Comparison to the Numerical Data to the Experimental Data	48
CHAPTER VIII: Conclusion	56
REFERENCES	58
APPENDIX.....	62

TABLE OF FIGURES

Figure 1.	Single edge crack on an infinitely wide plate.....	4
Figure 2.	Log-log plot of change in crack length per change in cycle vs. change in stress intensity factor.....	7
Figure 3.	Contour combination forming a closed contour on a region A*	8
Figure 4.	Arbitrary crack line divided into two enriched regions.....	13
Figure 5.	A body in state of elastostatic equilibrium.....	16
Figure 6.	Abaqus simulation model: 40 mm by 40 mm plate single-edge notch tension test.....	22
Figure 7.	Crack propagation simulation using Abaqus: maximum principal stress within crack vicinity from initial rupture to final crack length.	23
Figure 8.	Numerical integration path to evaluate J-integral.....	24
Figure 9.	Whole test model of auxetic materials with their corresponding representative volume element (RVE).....	27
Figure 10.	Double notch initial crack of an RVE with circular void subjected into tensile test.....	30
Figure 11.	Evolution of normalized stress intensity factor along the normalized crack length. Comparison of the 5% porosity reference data model to the calculated model of RVEs under periodic boundary condition using Abaqus.....	32
Figure 12.	Evolution of normalized stress intensity factor along the normalized crack length. Comparison of the reference data model to the calculated model of RVE under finite boundary condition using Abaqus.....	33
Figure 13.	Evolution of normalized stress intensity factor along the normalized crack length. Variation of RVE with 5% porosity ellipse void by increasing AR_e in increments of 3.....	35
Figure 14.	Evolution of normalized stress intensity factor along the normalized crack length. Variation of RVE with 5% porosity stop-hole void by increasing AR_{sl} in increments of 3.....	36
Figure 15.	Evolution of normalized stress intensity factor along the normalized crack length. Variation of RVE with 5% porosity stop-hole void by increasing AR_{sl} in increments of 3.....	37
Figure 16.	Maximum principal stress distribution of RVE circle void model A and stop-hole void models B and C.....	39

Figure 17.	Evolution of normalized stress intensity factor along the normalized crack length. Variation of RVE ellipse void by increasing AR_e in increments of 5 in constant L_{min}	41
Figure 18.	Evolution of normalized stress intensity factor along the normalized crack length. Variation of RVE slot void by increasing AR_e in increments of 5 in constant L_{min}	42
Figure 19.	Evolution of normalized stress intensity factor along the normalized crack length. Variation of RVE stop-hole void by increasing AR_r in increments of 2 in constant L_{min}	43
Figure 20.	Contour maps of the Lagrangian strains from the DIC of the non-auxetic and non-auxetic samples.....	47
Figure 21.	Abaqus assembly diagram for fatigue test simulation of laminate with circular void pattern and laminate with stop-hole void pattern...	48
Figure 22.	Maximum principal stress contour map from Abaqus with crack growth fracture simulation at specified percentage of time step of enriched region of sample with circular void pattern.....	50
Figure 23.	Maximum principal stress contour map from Abaqus with crack growth fracture simulation at specified percentage of time step of enriched region of sample with stop-hole void pattern.....	51
Figure 24.	Porous areas of the whole test specimens specifying outer and inner crack regions.....	52
Figure 25.	Evolution of stress intensity factor along the normalized crack length for Region I and Region II of the whole test specimen comparing circular void model (non-auxetic) to stop-hole void model (auxetic). For Region II, periodic models of circle void and stop-hole voids are compared.....	54

NOMENCLARURE

α	: Paris' law constant
a	: crack length
a_i	: initial crack length
a_f	: final crack length
a_e	: major axis length of ellipse void
a_{sl}	: major axis length of slot void
a_{sh}	: major axis length of stop-hole void
\mathbf{a}_j	: enrichment additional degrees of freedom associated with crack-body
A^*	: contour integral area
AR_e	: axis ratio of ellipse void
AR_{sl}	: axis ratio of slot void
AR_{sh}	: axis ratio of stop-hole void
β	: Paris' law constant
b_e	: minor axis length of ellipse void
b_{sl}	: minor axis length of slot void
b_{sh}	: minor axis length of stop-hole void
\mathbf{b}	: body force
\mathbf{b}_k^l	: enrichment additional degrees of freedom associated with crack-tip
\mathbf{B}	: strain-displacement matrix
C^-	: lower crack contour
C^+	: upper crack contour
C_1	: outer contour near crack region
C_2	: inner contour near crack region
\mathbf{C}	: material modulus
δ_{ij}	: kronecker delta
ε_{ij}^m	: mechanical strain
E	: modulus of elasticity
f	: damage criterion
f_{tol}	: damage tolerance
\mathbf{f}_{ext}	: nodal external forces
F_k^l	: crack-tip enrichment function
Γ	: general boundary
Γ_{cr}	: crack region boundary
Γ_t	: traction boundary
Γ_u	: displacement boundary
\mathcal{G}	: strain energy release rate
H	: Heaviside function

J	: contour J-intergral
K_I	: tensile stress intensity factor
K_{II}	: in-plane shear stress intensity factor
K_{III}	: out-of-plane shear stress intensity factor
L_{\min}	: minimum hole spacing
L_0	: hole spacing along x_1 direction at $x_2 = 0$
n	: number of elements of standard finite element
\mathbf{m}	: normal unit vector to C_2
m	: number of elements of enriched crack-body
mf	: number of elements of enriched crack-tip
\mathbf{n}	: normal unit vector to C_1
$N_i^{\text{fe}}(\mathbf{x})$: shape function associated with the standard finite element
$N_i^{\text{enr1}}(\mathbf{x})$: shape function associated with the discontinuity function of crack body
$N_i^{\text{enr2}}(\mathbf{x})$: shape function associated with the enrichment function of the crack-tip
N_f	: number of fatigue cycles
Ω	: crack domain
φ	: partition of unity basis function
ψ	: porosity
ϕ	: partition of unity arbitrary field enrichment function
Π	: potential energy
q	: smoothing function under closed contour of the crack region
r	: near distance from the crack tip to an arbitrary point
r_{sh}	: stop-hole void radius
R	: circle void radius
σ	: applied stress
σ_{11}	: local normal stress along x_1 direction
σ_{12}	: local shear stress along x_1 direction
σ_{22}	: local normal stress along x_2 direction
σ_{max}	: cartesian component of stress
σ_{max}^0	: maximum allowable principal stress
θ	: direction of arbitrary point with respect to the crack-tip
\mathbf{t}	: traction
t	: time of fracture at an arbitrary crack length
t_f	: time at complete fracture
\mathbf{u}	: displacement vector
ν	: Poisson's ratio
W	: strain density
w_p	: Gaussian weight
ξ	: arbitrary point location associated with x
x_1	: cartesian horizontal direction
x_2	: cartesian vertical direction

ABSTRACT

Recent studies suggest that auxetic materials such as porous metals with orthogonal periodic void patterns have an increased fatigue life compared to non-auxetic materials. This study provides numerical solution to support the existing experiments with the use of contour J-integral as a parameter of stress intensity factor for computing the number of fatigue life cycle of the materials with auxetic structures. Representative volume elements (RVEs) were constructed to characterize the physical test specimens with void patterns such as ellipse, slot, and stop-hole. Extended finite element method (XFEM) was performed to verify the direction of crack propagation on auxetic materials. Sixty-five distinct RVEs were made for each void shape with increasing horizontal double notch to mimic the crack propagation. Using Abaqus, the contour J-integral was calculated automatically at the crack-tip region. Numerical computation showed that auxetics have lower rate of overall crack propagation compared to non-auxetics. Variation of geometric parameters were employed to the void patterns of the RVE which changed the porosity and the minimum hole distance of the auxetics. Computation on stress intensity factor for each crack increment showed that models with relatively larger negative Poisson's ratio have faster crack initiation. XFEM and J-integral simulations were performed on aluminum plates with circular and stop-hole void patterns and compared with experimental data. Results were in good agreement to the experiment where stop-hole void model had lower rate of crack evolution compared to the circular void model.

CHAPTER I

Introduction

Auxetics are materials that exhibit unusual behavior compared to typical engineering materials in that when they are stretched axially they expand transversely [1]. The concept behind this exceptional property is Poisson's ratio, ν – the ratio of the negative value of the lateral strain to the longitudinal strain of a material subjected in unidirectional load or displacement which ranges from -1.0 to 0.50 [2]. For a conventional engineering material (e.g. metal, wood, polymers), ν is greater than zero, but for auxetic materials, ν is less than zero. This includes but is not limited to metallic foams [2], polycrystalline ceramics [3], microporous polymer [4], metallic nanoplates [5], fiber reinforced composite [6] and laminates [7]. The physical behavior of these metamaterials comes from its internal structures which affect their deformation mechanism [8]. These structures allow a combination of flexure, hinging, and stretching of the material's unit cell [9] to achieve a negative Poisson's ratio. To tailor such structure, one of the physical features auxetics should have is high porosity [10], and auxetic behavior has been demonstrated on star-honeycomb [11], sinusoid ligament [12], and lozenge grid [13] structures. However, a recent study by Taylor et. al. paved the way on the investigation of low porosity auxetic material (2% to 5% porosity). In this study, an aluminum alloy sheet with symmetric, orthogonal elliptical voids subjected to tensile testing showed that increasing the aspect ratio of the elliptical voids reduces the Poisson's ratio to a more negative value [14]. Francesconi et. al. expanded the research of metallic sheets with two-dimensional, orthogonal void by studying the in-plane and out-of-plane eigenmodes of porous materials with more geometric variation of void patterns [15].

Javid et. al. demonstrated for stainless steel, that auxetic samples with novel orthogonal S-shaped void have longer fatigue life than non-auxetic samples with circular holes [16]. However, this research is limited to only one geometric feature of an auxetic material for a fatigue experiment, so to fill this literature gap, this paper employs an additional variation of geometries that will allow the reader to identify that: changing the shape parameter and porosity has an effect on the fatigue crack behavior of auxetics. This

paper was also motivated by the experimental results obtained by Francesconi et. al. in which the authors tested the fatigue life of auxetic materials with circular voids and stop-holes under tensile cyclic load [17]. Numerical analysis is used to model the observed behavior with the use of extended finite element method (XFEM), and the contour J-integral. XFEM was implemented to predict the crack initiation location and the propagation behavior while the contour J-integral was calculated to approximate the strain energy release rate. Then, we used the concept of Paris Law [18] to determine the number of cycles to failure of the auxetic materials. Sixty-five representative volume element models were created, each having a distinct representation of a horizontal double notched crack. The crack lengths were based on the minimum distance between holes, ranging from 10% to 90% of the minimum hole spacing. We have improved the procedure of Javid et. al. by employing a wider range of crack propagation path for the calculation of contour J-integral. In previous study, the crack length range makes it limited to observing the middle phase of crack propagation where the crack initiation and total rupture phase are excluded [18]. To enhance the simulation, we implemented 1% to 99% of minimum hole spacing to observe the crack initiation, crack evolution, and rupture. Aside from using periodic boundary condition, we also have applied finite boundary conditions on the actual plate specimen and demonstrated the comparison between the two methods.

The first part of the paper addresses the theory and numerical computation while the second part demonstrates the methodology and numerical results. Chapter II outlines the underlying concepts of linear elastic fracture theory, while Chapter III provides discussion of XFEM which is applied to simulate the crack behavior and predict its direction. The commercially available software package, Abaqus Simulia (by Dassault Systemes), was utilized to implement the finite element analysis and the procedure of the simulation is documented Chapter IV. Chapter V provides specification on the material and geometries that was used in the experiment and Chapter VI lists the methodology on obtaining the result of stress intensification factor at their respective crack length. Lastly, Chapter VII demonstrates a comparison of the experimental result to the numerical method that was described from the previous sections.

CHAPTER II

Fracture Mechanics Fundamentals

2.1 Background

In providing a quantitative interpretation of the fatigue crack growth of a linearly elastic auxetic material, it is important to understand the theoretical concepts governing the general behavior of crack propagation. This will be beneficial in the succeeding chapters since it will provide explanation on the relation of crack length extension to energy, stress and displacement. Furthermore, topics on fracture mechanics such as Paris Law and path-independent J-integral will be examined to provide analytical information on the numerical solution on the subsequent topics such as in Chapter III and Chapter V.

2.2 Energy Release Rate

Equivalent to Griffith energy balance on defining a crack extension [19], Irwin proposed an approach in which the energy release rate \mathcal{G} is in terms of the potential energy Π and the crack length a [20].

$$\mathcal{G} = -\frac{d\Pi}{da} \quad (2.1)$$

Equation 2.1 states that \mathcal{G} is a measure of the rate of change of the potential energy dissipation with the crack length.

2.3 Stress Intensification Factor

Consider three modes of loading that can be applied to an infinitely wide plate. As illustrated in Figure 1a, Mode I represents a tensile loading normal to the crack area that may result to a crack opening along x_1 direction. Mode II and Mode III demonstrate an

in-plane shear and out-of-plane shear respectively [21]. In this study, the research on the test specimen is subjected to cyclic tensile loading. Thus, the succeeding discussion is focused on Mode I type of loading.

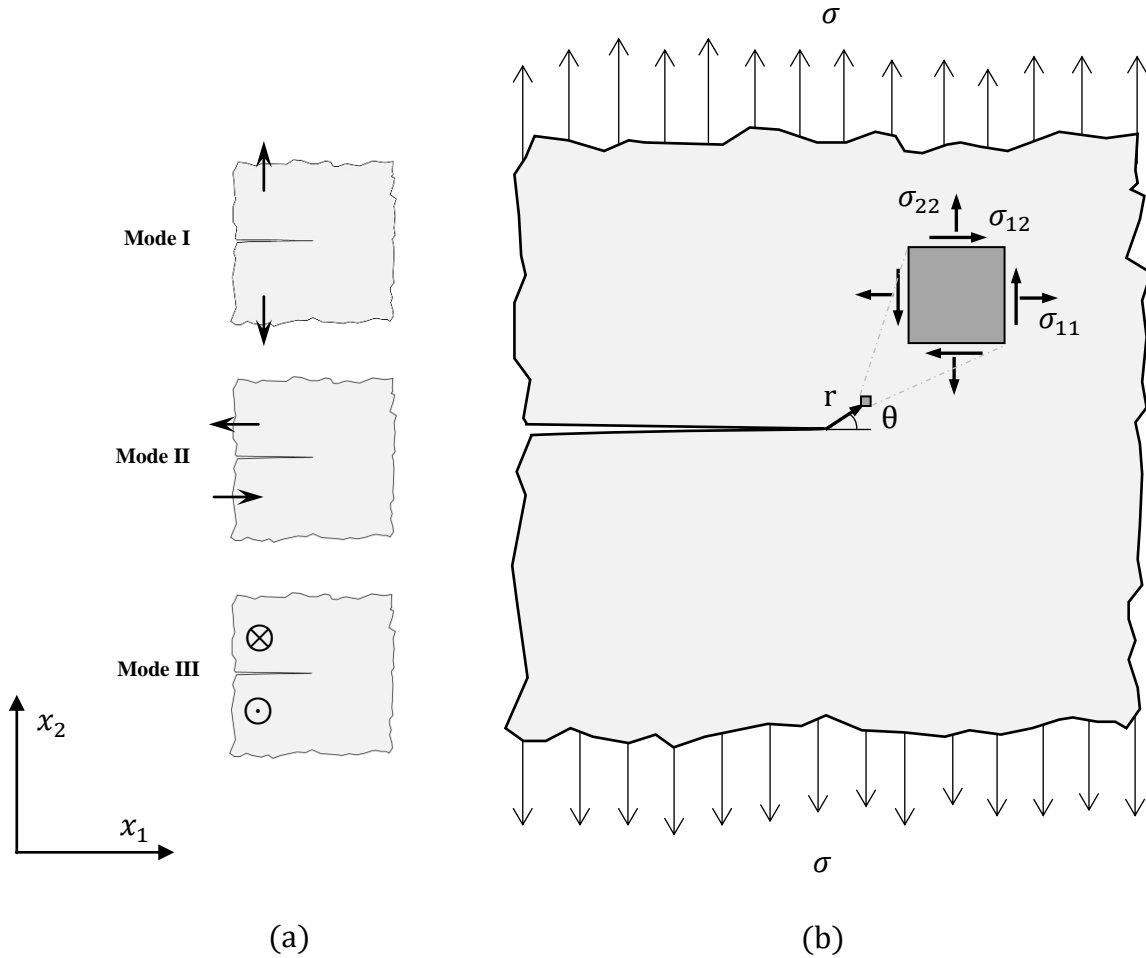


Figure 1. Single edge crack on an infinitely wide plate. (a) Three modes of loading applied to a crack (b) coordinate axis representation of local stress near the crack tip of a plate subjected to a remote tensile stress, σ .

Westergaard pioneered the solution for the local stresses near the crack tip [22] followed up by the works of Irwin, Sih and Sanford who formulated a generalized formula for the stress solution [23-25]. Given an initial crack length, a , and applied stress, σ , Equations 2.2 to 2.4 outline the local stresses located at a specific magnitude, r , and

direction, θ , at the very end of the crack tip described in Figure 1b. According to Westergaard's complex variable solution, the stresses near the crack tip of an isotropic linear elastic type of material with a Mode I type of loading can be derived as follows:

$$\sigma_{11} = \frac{\sigma \sqrt{a}}{\sqrt{2r}} \cos \frac{1}{2} \theta \left(1 - \sin \frac{1}{2} \theta \sin \frac{3}{2} \theta \right) \quad (2.2)$$

$$\sigma_{22} = \frac{\sigma \sqrt{a}}{\sqrt{2r}} \cos \frac{1}{2} \theta \left(1 + \sin \frac{1}{2} \theta \sin \frac{3}{2} \theta \right) \quad (2.3)$$

$$\sigma_{12} = \frac{\sigma \sqrt{a}}{\sqrt{2r}} \sin \frac{1}{2} \theta \cos \frac{1}{2} \theta \cos \frac{3}{2} \theta \quad (2.4)$$

Irwin modified the above equations [23] by introducing a constant called stress intensity factor, $K_I = \sigma \sqrt{\pi a}$ (Mode I). Referring to Equations 2.5 to 2.7, the use of K_I is convenient since the applied force on the plate and the crack length is combined to a single constant that can be considered as an amplitude of the local stress fields within a singularity, $1/\sqrt{r}$.

$$\sigma_{11} = \frac{K_I}{\sqrt{2\pi r}} \cos \frac{1}{2} \theta \left(1 - \sin \frac{1}{2} \theta \sin \frac{3}{2} \theta \right) \quad (2.5)$$

$$\sigma_{22} = \frac{K_I}{\sqrt{2\pi r}} \cos \frac{1}{2} \theta \left(1 + \sin \frac{1}{2} \theta \sin \frac{3}{2} \theta \right) \quad (2.6)$$

$$\sigma_{12} = \frac{K_I}{\sqrt{2\pi r}} \sin \frac{1}{2} \theta \cos \frac{1}{2} \theta \cos \frac{3}{2} \theta \quad (2.7)$$

For linear elastic fracture mechanics, the validity of stress intensity factor only applies to a singularity dominated zone where r approaches zero. Within that region, K_I can be defined as amplitude of the stress field at a given r and θ .

2.4 Relationship between \mathcal{G} and K_I .

Strain energy release rate and stress intensification factor play an important role in fracture mechanics. While \mathcal{G} describes crack propagation globally as the degradation of potential energy due to crack extension, K_I characterizes the magnitude of stress field locally, these two parameters are related to one another [26]. For a single notch crack with uniform tensile stress at an infinitely wide plate exhibiting a linear elastic behavior and plane stress condition, the relationship between \mathcal{G} and K_I is

$$\mathcal{G} = \frac{K_I^2}{E} \quad (2.8)$$

where E is the modulus of elasticity.

2.5 Fatigue and Paris Law

Given \mathcal{G} and E , one can manipulate Equation 2.8 to evaluate the stress intensity factor which will be used to identify the behavior of a crack growth. As illustrated in Figure 2, $\log \frac{da}{dN}$ vs. $\log \Delta K$ plot demonstrates a sigmoidal curve which can be observed as a fatigue crack behavior of metals. The curve is divided into three regions. Region I, at the lower end of the curve, is composed of a crack growth rate starting from a stress intensification factor threshold, K_{th} , then the change in crack length per cycle extends slowly to the boundary of Region II. Region III, at the upper portion, is represented by a relatively faster crack growth until rupture at critical stress intensity factor, K_C . Region II is where Paris and Erdogan described the section from which the crack propagation shows a linear behavior with slope β on logarithmic scale plot [18]. Equation 2.9 describes the plot within Region II.

$$\frac{da}{dN} = \alpha \Delta K^\beta \quad (2.9)$$

A power-law relationship for fatigue crack growth where change in crack length per cycles is proportional to a power of change in stress intensity factor. α and β are material constants which depend on material and environmental condition determined from experiments [21].

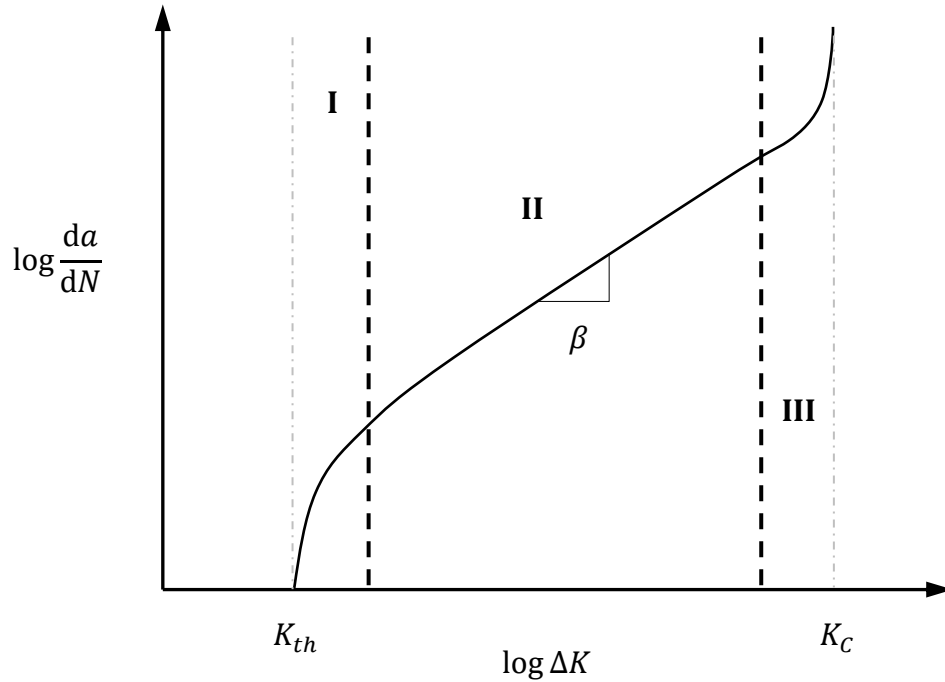


Figure 2. Log-log plot of change in crack length per change in cycle vs. change in stress intensity factor which represents the fatigue crack growth of metals (reproduced without permission) [21].

Given the change in stress intensity factor and the values of material constants, the number of fatigue life cycles can be obtained by integrating Equation 2.1 [18, 21]:

$$N_f = \int_{a_i}^{a_f} \frac{da}{\alpha \Delta K^\beta} \quad (2.10)$$

2.6 J-integral Analytical and Numerical Solution

For a common tensile test with simple geometry such as single edged notched specimen or center-crack specimen, the analytical solution for stress intensity factor is formulated based on the geometry of the test samples [21]. On the other hand, the J-integral is used for more complex geometries on the samples such as those of auxetic materials to approximate the value of the K_I [18].

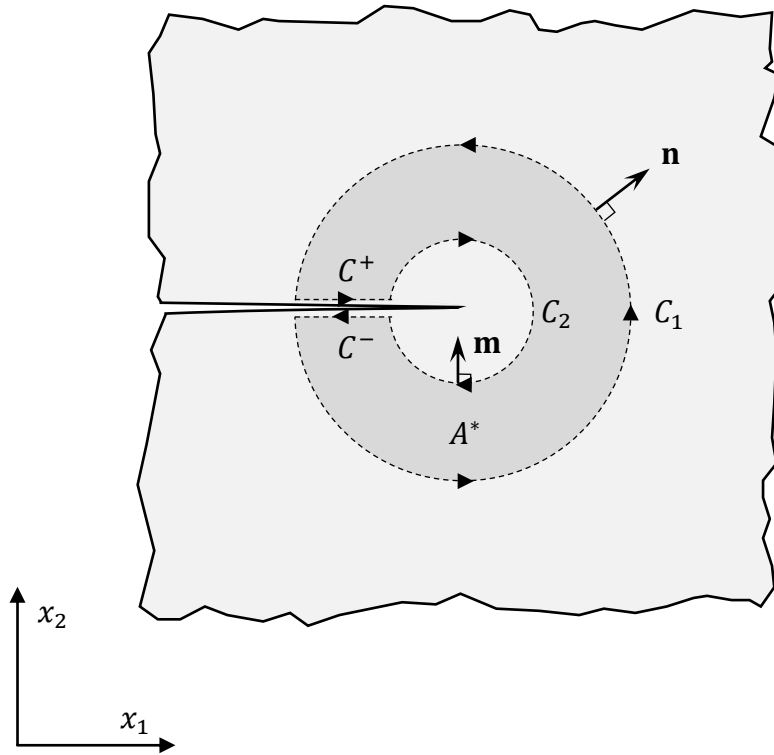


Figure 3. Contour combination forming a closed contour on a region A^* (reproduced without permission) [27].

Applying the concept of virtual crack extension [26], the J-integral can be interpreted as

$$J = -\frac{d\Pi}{da}, \quad (2.11)$$

which is equivalent to the energy release rate for linear elastic material.

$$J = \mathcal{G}. \quad (2.12)$$

Referring to Figure 3, a closed contour forming an area, A^* , can be written as follows:

$$C = C^+ + C^- + C_1 - C_2 \quad (2.13)$$

where C^+ and C^- are the contour in opposite direction facing the crack and C_1 and C_2 are the outer and inner contour surrounding the crack tip. It is also important to note that $m_i = -n_i$, where \mathbf{m} and \mathbf{n} are unit normal vectors of C_1 and C_2 respectively.

Shih et. al presented a generalized solution on J-integral [27], assuming a crack extension along x_1 direction at a certain crack tip region, C_2 , at quasi-static condition,

$$J = \lim_{C_2 \rightarrow 0} \int_C (W \delta_{1i} - \sigma_{ij} u_{j,1}) n_i dC \quad (2.14)$$

where W is the strain energy density given as:

$$W = \int_0^{\epsilon_{ij}} \sigma_{ij} d\epsilon_{ij} \quad (2.15)$$

where σ_{ij} is the cartesian components of stress and u_j and ϵ_{ij} are the displacement and mechanical strain respectively, n_i is the unit normal vector along C_2 [28].

Li derived Equation 2.14 by applying path-independence concept of the contour and by assuming that integrals along C^+ and C^- cancelled each other out and C_2 is at the very tip of the crack [27-29].

$$J = \int_{A^*} [(\sigma_{ij} u_{j,1} - W \delta_{1i}) q]_{,i} dA \quad (2.16)$$

where q is a smooth function enclosing the area A under the close contour C that is unity on C_2 and C_1 as C_2 approaches zero.

CHAPTER III

Extended Finite Element Method (XFEM)

3.1 Background

The numerical method that is implemented to predict the crack length and direction applying the concept of fracture mechanics to finite element method is called extended finite element method (XFEM). In the study described in the succeeding chapters (Chapter V and VI), the employment of XFEM is vital in verifying the path of the crack which will be used to support the assumption of the J-integral numerical analysis.

XFEM features an efficient method of numerical approximation where, instead of remeshing multiple times as crack propagates at a certain period to account for new boundaries, jump dislocation functions and enrichment functions are utilized to enable representation of a crack which may be located between mesh nodes [30-34]. Thus, crack can move through the finite elements. In this chapter, the fundamentals of XFEM are described. The discretization of the XFEM solution is also explained to unveil the underlying numerical concepts used in finite element analysis (FEA) software.

3.2 Partition of Unity

We continue the discussion by introducing the most basic mathematical framework of XFEM. Developed by Melenk and Babuska [35], the so-called partition of unity method (PUM) accounts for the structured composition of a global space to an approximation of a local behavior solution of a finite element space. Within a domain Ω , the partition of unity of the set of n functions $\varphi_i(\mathbf{x})$, is defined as

$$\sum_{i=1}^n \varphi_i(\mathbf{x}) = 1 \quad \forall \mathbf{x} \in \Omega \quad (3.1)$$

Proceeding from Equation 3.1, given an arbitrary field, $\phi(\mathbf{x})$ the following property should be satisfied,

$$\sum_{i=1}^n \varphi_i(\mathbf{x}) \phi_i(\mathbf{x}) = \phi(\mathbf{x}) \quad (3.2)$$

Equation 3.2 represents the concept of completeness of a solution in which the function $\varphi_i(\mathbf{x})$ is approximated by expressing in terms of the order of the function $\phi(\mathbf{x})$ [33].

A classical implementation of this concept is the n number of shape function of the set of an isoparametric finite elements given as,

$$\sum_{i=1}^n N_i(\mathbf{x}) = 1 \quad (3.3)$$

Similar to Equation 3.2, partition of unity can be applied to a displacement field \mathbf{u} :

$$\sum_{i=1}^n N_i(\mathbf{x}) \mathbf{u}_i(\mathbf{x}) = \mathbf{u}(\mathbf{x}) \quad (3.4)$$

where $\mathbf{u}(\mathbf{x})$ is the interpolant of $\mathbf{u}_i(\mathbf{x})$.

Completeness is necessary to achieve a desired accuracy from a given series of functions to approximate a particular smooth function. For example, in elasticity, \mathbf{u} can take on constant values to represent a rigid body motion and constant strain states. Also, completeness is important such that trial solutions and weight functions including their derivatives converge as the finite element size approach zero [36]. PUM ensures that finite element approximation is complete.

3.3 XFEM Enrichment

The concept of the PUM is employed in XFEM where the classical displacement solution in finite element function is composed of an additional set of m enrichment functions, $\phi(\mathbf{x})$ [33] (Equation 3.6)

$$\mathbf{u}(\mathbf{x}) = \mathbf{u}^{\text{fe}} + \mathbf{u}^{\text{enr}} \quad (3.5)$$

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^n N_i^{\text{fe}}(\mathbf{x}) \mathbf{u}_i(\mathbf{x}) + \sum_{i=1}^m N_i^{\text{enr}}(\mathbf{x}) \phi(\mathbf{x}) \mathbf{a}_i \quad (3.6)$$

where $N_i^{\text{fe}}(\mathbf{x})$ are the standard shape functions and $N_i^{\text{enr}}(\mathbf{x})$ is the shape function associated enrichment solution, while $\mathbf{u}_i(\mathbf{x})$ are the standard nodal degrees of freedom for finite element method and \mathbf{a}_i are the additional unknown degrees of freedom. Note that by PUM when $\mathbf{a}_i = \mathbf{1}$ and $\mathbf{u}_i = \mathbf{0}$, the enrichment function $\phi(\mathbf{x})$ represents exactly the approximation of $\mathbf{u}(\mathbf{x})$. Typically, both standard approximation and enrichment approximation use equal shape functions ($N_i^{\text{fe}}(\mathbf{x}) = N_i^{\text{enr}}(\mathbf{x})$) but in some case where the enrichment region uses different type of elements with respect to the standard finite element region (e.g. quadrilateral for standard region, and sub-triangles for enriched regions) $N_i^{\text{fe}}(\mathbf{x}) \neq N_i^{\text{enr}}(\mathbf{x})$ [30, 37].

Enrichment region for XFEM crack model has two parts as illustrated in Figure 4 and will be discussed in the succeeding sections. Region with circular nodes are the enriched elements of the discontinuous crack-body while the square nodes are applied for the enrichment of crack-tip.

3.4 Solution for Discontinuity

To model the discontinuity of the enriched crack region, a modified Heaviside function, $H(\xi)$, (signed function) is implemented as the enrichment function

$$\phi = H(\xi) = \begin{cases} -1, & \text{if } \xi < 0 \\ +1, & \text{if } \xi > 0 \end{cases} \quad (3.7)$$

where ξ is the arbitrary location point associated to x [35]. $H(\xi) = +1$ represents one side of the discontinuous element while $H(\xi) = -1$ represents the other [30].

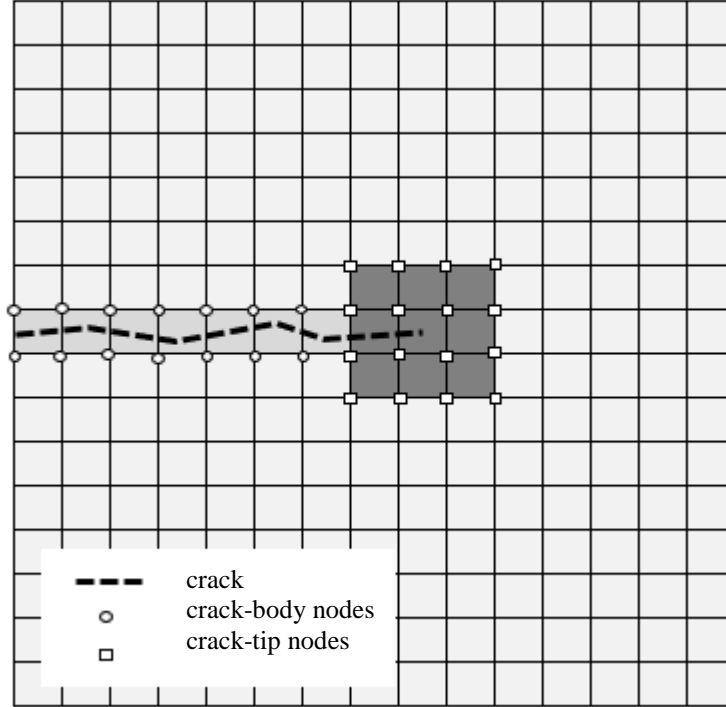


Figure 4. Arbitrary crack line divided into two enriched regions (reproduced without permission) [38].

With the application of (3.7), (3.6) can be written as

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^n N_i^{\text{fe}}(\mathbf{x})\mathbf{u}_i(\mathbf{x}) + \sum_{i=1}^m N_i^{\text{enr}}(\mathbf{x})H(\xi)\mathbf{a}_i. \quad (3.8)$$

However, if we verify the approximation of (3.8) the interpolation of value of the displacement field $\mathbf{u}(\mathbf{x})$ is derived as

$$\mathbf{u}(\mathbf{x}_i) = \mathbf{u}_i + H(\xi_i)\mathbf{a}_i \neq \mathbf{u}_i. \quad (3.9)$$

From (3.9) the field variable $\mathbf{u}(\mathbf{x})$ means that the displacement field is not an interpolation of nodal parameters \mathbf{u}_i . To account for interpolation error correction, $H(\xi)$ is shifted to a node of interest [30, 37]. Thus (3.8) is modified to

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^n N_i^{\text{fe}}(\mathbf{x})\mathbf{u}_i(\mathbf{x}) + \sum_{i=1}^m N_i^{\text{enr}}(\mathbf{x})(H(\xi) - H(\xi_i))\mathbf{a}_i. \quad (3.10)$$

3.5 Crack-tip Enrichment

Since (3.10) only applies for the representation of the discontinuity of the crack-body, additional functions to include the enrichment for the crack-tip is accounted in the XFEM solution,

$$\begin{aligned} \mathbf{u}(\mathbf{x}) = & \sum_{i=1}^n N_i^{\text{fe}}(\mathbf{x})\mathbf{u}_i(\mathbf{x}) + \sum_{i=1}^m N_i^{\text{enr1}}(\mathbf{x})(H(\xi(x)) - H(\xi_i))\mathbf{a}_i \\ & + \sum_{i=1}^{mf} N_i^{\text{enr2}}(\mathbf{x}) \left[\sum_{k=1}^{mp} F^k(x)\mathbf{b}_i^k \right], \end{aligned} \quad (3.11)$$

where \mathbf{b}_i^k are unknown values for the degrees of freedom associated to the crack-tip region [37].

As shown in Figure 4, multiple elements are enriched around the crack-tip region. This explains the summation on the function $F^k(x)$ where the generalized PUM is employed to represent mf number of domains [39].

Focusing on the function $F^k(x)$, the basis of this crack-tip enrichment function is the Westergaard field at the very near tip region which is redefined by Fleming [40]. Parallel to the formulation of stress intensification factor, $F^k(x)$ can also be derived through polar form as in (3.12) to (3.15).

$$F^1(r, \theta) = \sqrt{r} \sin\left(\frac{\theta}{2}\right) \quad (3.12)$$

$$F^2(r, \theta) = \sqrt{r} \cos\left(\frac{\theta}{2}\right) \quad (3.13)$$

$$F^3(r, \theta) = \sqrt{r} \sin\left(\frac{\theta}{2}\right) \sin \theta \quad (3.14)$$

$$F^4(r, \theta) = \sqrt{r} \cos\left(\frac{\theta}{2}\right) \sin \theta \quad (3.15)$$

Similar to the remedy in (3.10), $F^k(r, \theta)$ is shifted to guarantee the appropriate interpolation correction given in the generalized XFEM solution

$$\begin{aligned} \mathbf{u}(\mathbf{x}) = & \sum_{i=1}^n N_i^{\text{fe}}(\mathbf{x}) \mathbf{u}_i(\mathbf{x}) + \sum_{i=1}^m N_i^{\text{enr1}}(\mathbf{x}) (H(\xi(x)) - H(\xi_i)) \mathbf{a}_i \\ & + \sum_{i=1}^{mf} N_i^{\text{enr2}}(\mathbf{x}) \left[\sum_{k=1}^4 (F^k(r, \theta) - F^k(x_i)) \mathbf{b}_i^k \right] \end{aligned} \quad (3.16)$$

where $N_i^{\text{enr2}}(\mathbf{x})$ is the set of mf shape functions associated with the enrichment on the crack-tip region [37].

3.6 XFEM Discretization

As a preliminary before discussing the XFEM discretization, it is important to define the fundamental equations of a crack model in elastostatic equilibrium and this will be the foundation of the XFEM discrete solutions (Figure 5). Given Ω as the region bounded by the smooth curve Γ with displacement, \mathbf{u} , traction, \mathbf{t} and body force, \mathbf{b} , the strong form of the initial boundary value problem has the following equations [34, 30]:

$$\nabla \cdot \boldsymbol{\sigma} + \mathbf{b} = \mathbf{0} \quad \text{in } \Omega \quad (3.17)$$

$$\mathbf{u} = \bar{\mathbf{u}} \quad \text{in } \Gamma_u \quad (3.18)$$

$$\boldsymbol{\sigma} \cdot \mathbf{n} = \bar{\mathbf{t}} \quad \text{in } \Gamma_t \quad (3.19)$$

$$\boldsymbol{\sigma} \cdot \mathbf{n} = \mathbf{0} \quad \text{in } \Gamma_{cr} \quad (3.20)$$

where $\boldsymbol{\sigma}$ is the Cauchy stress tensor, $\bar{\mathbf{t}}$ and $\bar{\mathbf{u}}$ are the prescribed traction and displacement respectively, \mathbf{n} is the outward unit vector with respect to Γ .

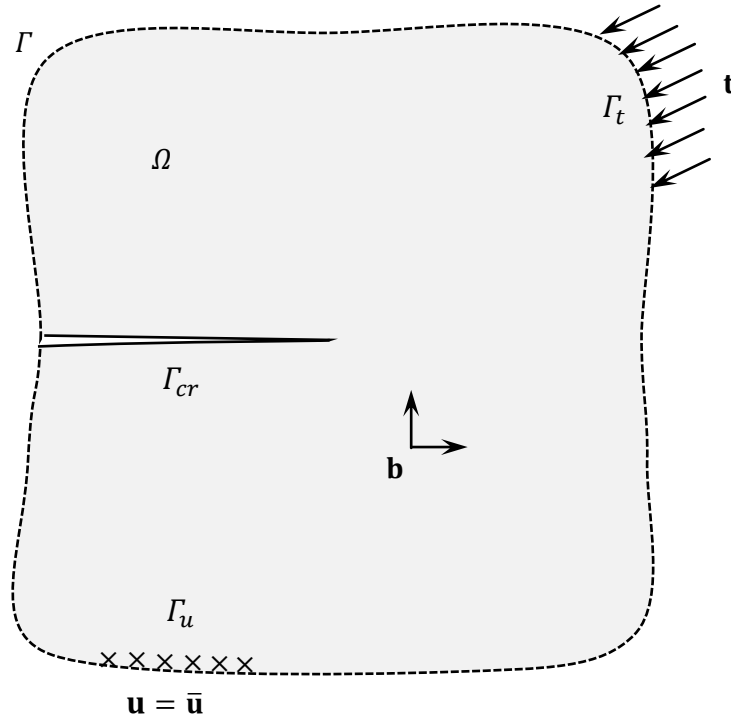


Figure 5. A body in state of elastostatic equilibrium.

On the other hand, the weak form of the initial boundary value problem is

$$\int_{\Omega} \boldsymbol{\sigma} \cdot \delta \boldsymbol{\varepsilon} = \int_{\Omega} \mathbf{b} \cdot \delta \mathbf{u} \, d\Omega + \int_{\Gamma} \mathbf{t} \cdot \delta \mathbf{u} \, d\Gamma \quad (3.21)$$

where $\boldsymbol{\varepsilon}$ is defined as the strain. The later equation will be used to formulate the standard discrete equation of XFEM [32].

While fracture models consist of a growing discontinuous region, the strong form is difficult to use because it complicates the required boundary conditions. Thus, we use weak form (3.21) since the continuity requirement is reduced for the finite element approximation and evaluation of element stiffness involves polynomial functions that are easy to interpolate by numerical methods such as Gauss Quadrature [39].

From (3.16), we can now define the strain solution by substituting the displacement approximation $\mathbf{u} = \bar{\mathbf{u}}$ to the strain expression

$$\boldsymbol{\varepsilon} = \bar{\mathbf{B}}\bar{\mathbf{u}} \quad (3.22)$$

where the strain-displacement matrix and displacement matrix are as follows

$$\bar{\mathbf{B}} = [\mathbf{B}_i^u \quad \mathbf{B}_i^a \quad \mathbf{B}_j^{b1} \quad \mathbf{B}_j^{b2} \quad \mathbf{B}_j^{b3} \quad \mathbf{B}_j^{b4}] \quad (3.23)$$

$$\bar{\mathbf{u}}^T = [\mathbf{u}_i \quad \mathbf{a}_i \quad \mathbf{b}_j^1 \quad \mathbf{b}_j^2 \quad \mathbf{b}_j^3 \quad \mathbf{b}_j^4] \quad (3.24)$$

The $\bar{\mathbf{B}}$ matrix specific components are as follows:

For standard finite element:

$$\mathbf{B}_i^u = \begin{bmatrix} N_{i,1}^{fe} & 0 \\ 0 & N_{i,2}^{fe} \\ N_{i,2}^{fe} & N_{i,1}^{fe} \end{bmatrix} \quad (3.25)$$

For the enriched region on the crack-body:

$$\mathbf{B}_i^a = \begin{bmatrix} N_i^{\text{enr1}} \left(H(\xi(x)) - H(\xi_j) \right)_{,1} & 0 \\ 0 & N_i^{\text{enr1}} \left(H(\xi(x)) - H(\xi_j) \right)_{,2} \\ N_i^{\text{enr1}} \left(H(\xi(x)) - H(\xi_j) \right)_{,2} & N_i^{\text{enr1}} \left(H(\xi(x)) - H(\xi_j) \right)_{,1} \end{bmatrix} \quad (3.26)$$

For the enriched region on the crack-tip:

$$\mathbf{B}_j^{\text{bk}} \Big|_{k=1,2,3,4} = \begin{bmatrix} N_i^{\text{enr2}}(F^k(r, \theta) - F^k(x_j))_{,1} & 0 \\ 0 & N_i^{\text{enr2}}(F^k(r, \theta) - F^k(x_j))_{,2} \\ N_i^{\text{enr2}}(F^k(r, \theta) - F^k(x_j))_{,2} & N_i^{\text{enr2}}(F^k(r, \theta) - F^k(x_j))_{,1} \end{bmatrix} \quad (3.27)$$

One can also obtain the standard discrete system of equations by substituting (3.16) to the following,

$$\mathbf{f}^{\text{ext}} = \mathbf{K}\bar{\mathbf{u}} \quad (3.28)$$

where \mathbf{f}^{ext} is the nodal external forces and are given as

$$\mathbf{f}^{\text{extT}} = [\mathbf{f}_i^{\text{u}} \quad \mathbf{f}_i^{\text{a}} \quad \mathbf{f}_j^{\text{b1}} \quad \mathbf{f}_j^{\text{b2}} \quad \mathbf{f}_j^{\text{b3}} \quad \mathbf{f}_j^{\text{b4}}] \quad (3.29)$$

The details of the values from the expression of (3.28) are the following

For standard finite element:

$$\mathbf{f}_i^{\text{u}} = \int_{\Gamma_t} N_i^{\text{fe}} \bar{\mathbf{t}} \, d\Gamma + \int_{\Omega} N_i^{\text{fe}} \mathbf{b} \, d\Omega \quad (3.30)$$

For the enriched region on the crack-body:

$$\mathbf{f}_i^{\text{a}} = \int_{\Gamma_t} N_i^{\text{enr1}} (H(\xi(x)) - H(\xi_j)) \bar{\mathbf{t}} \, d\Gamma + \int_{\Omega} N_i^{\text{enr1}} (H(\xi(x)) - H(\xi_j)) \mathbf{b} \, d\Omega \quad (3.31)$$

For the enriched region on the crack-tip:

$$\begin{aligned} \mathbf{f}_j^{\text{bk}} \Big|_{k=1,2,3,4} &= \int_{\Gamma_t} N_i^{\text{enr}2} \left(F^k(r, \theta) - F^k(x_j) \right) \bar{\mathbf{t}} \, d\Gamma \\ &+ \int_{\Omega} N_i^{\text{enr}2} \left(F^k(r, \theta) - F^k(x_j) \right) \mathbf{b} \, d\Omega \end{aligned} \quad (3.32)$$

In addition, the stiffness matrix, \mathbf{K} , from Equation 27 is formulated by the following expression:

$$\mathbf{K} = \int_{\Omega} \bar{\mathbf{B}}^T \mathbf{C} \bar{\mathbf{B}} \, d\Omega, \quad (3.33)$$

where \mathbf{C} is the material modulus matrix [30, 34].

For plane stress assumption, the isotropic material has the following matrix,

$$\mathbf{C} = \frac{E}{1 - \nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1 - \nu)/2 \end{bmatrix}, \quad (3.34)$$

where ν is the Poisson's ratio of the bulk material [36].

CHAPTER IV

Abaqus Implementation

4.1 Background

As stated earlier in the introduction, Abaqus was utilized to simulate crack propagation. Chapter III is connected to this sub-topic since Abaqus provides XFEM features that can implement enrichment function and discontinuity which allows simulation of crack propagation. Here, we will focus on the software implementation of fracture criterion, crack initiation, crack path, and damage evolution [41]. Additional information on how J-integral is discretized and implemented in Abaqus is also discussed in this Chapter.

4.2 Fracture Criterion

Traction-separation cohesive behavior was used to implement the simulation of crack propagation since it is more suitable for ductile materials, which are the focus of this work, compared to other methods [16, 41]. One of its damage initiation criteria, f , is based on the ratio of the maximum principal stress determined from finite element method, σ_{\max} and the allowable principal stress, σ_{\max}^0 ,

$$f = \left\{ \frac{\sigma_{\max}}{\sigma_{\max}^0} \right\}. \quad (4.1)$$

It is also important to note that σ_{\max} is assumed to be zero if its value is negative. This means that if the stress is purely compressive, the damage will not be initiated. Intuitively, damage occurs if f reaches the value of 1.0 or greater.

Abaqus requires an initial crack to be placed in the specimen because the basis of the model is linear elastic fracture mechanics by default. However, if initial crack is not specified, Abaqus will allow nucleation based on the area where maximum principal stress

exceeds the allowable value. In addition to the damage criterion, an input of damage tolerance, f_{tol} , such that the range for damage is

$$1.0 \leq f \leq 1.0 + f_{tol}. \quad (4.2)$$

At specific tolerance, if $f > 1.0 + f_{tol}$, the standard time increment is refined until the value of f is within the range of (4.2).

4.3 Crack Initiation

For the crack direction on two-dimensional model, when maximum allowable principal stress is specified, by default, the crack direction is always orthogonal to the direction of the maximum principal stress. However, there is an option in the software that applies the work Erdogan and Sih [42] to compute for the crack direction,

$$\theta = \arccos\left(\frac{2K_{II}^2 + \sqrt{K_I^4 + 8K_I^2 K_{II}^2}}{K_I^2 + 9K_{II}^2}\right), \quad (4.3)$$

where K_I , and K_{II} are stress intensity factors based on the modes of loading (see Section II). To specify the direction, Abaqus requires the user to input the modulus of elasticity, E , and strain energy release rates \mathcal{G} and use (2.8) to estimate the value of the stress intensification factor. However, in the case of unidirectional tensile loading (mode I), from (4.3), the direction θ will become zero.

To illustrate Abaqus' implementation, a 40 mm by 40 mm by 1 mm stainless steel plate with initial crack length of 2.5 mm was created (Figure 6). This provides a simple example of the input, procedure and result of Abaqus in running a traction-separation crack propagation simulation under plane-stress condition. The elastic properties for stainless steel are $E = 193$ GPa and $\nu = 0.33$. For the damage property, $\sigma_{max}^0 = 250$ MPa was included as the criterion for damage initiation. The strain energy release rate, $\mathcal{G} = 4$ J/mm²,

was also used for the initial direction of the crack extension. For the load and boundary conditions, a 500 N distributed load at the top edge and fix boundary at the bottom was inputted respectively. In order to apply the XFEM option the middle section of the plate (Figure 7) was selected as the enrichment region. We have implemented a 4-node bilinear plane-stress quadrilateral element (Abaqus Element Code: CPS4). Also, we used global seed mesh of 4 mm for the whole region except for the enrichment region where we used 1 mm.

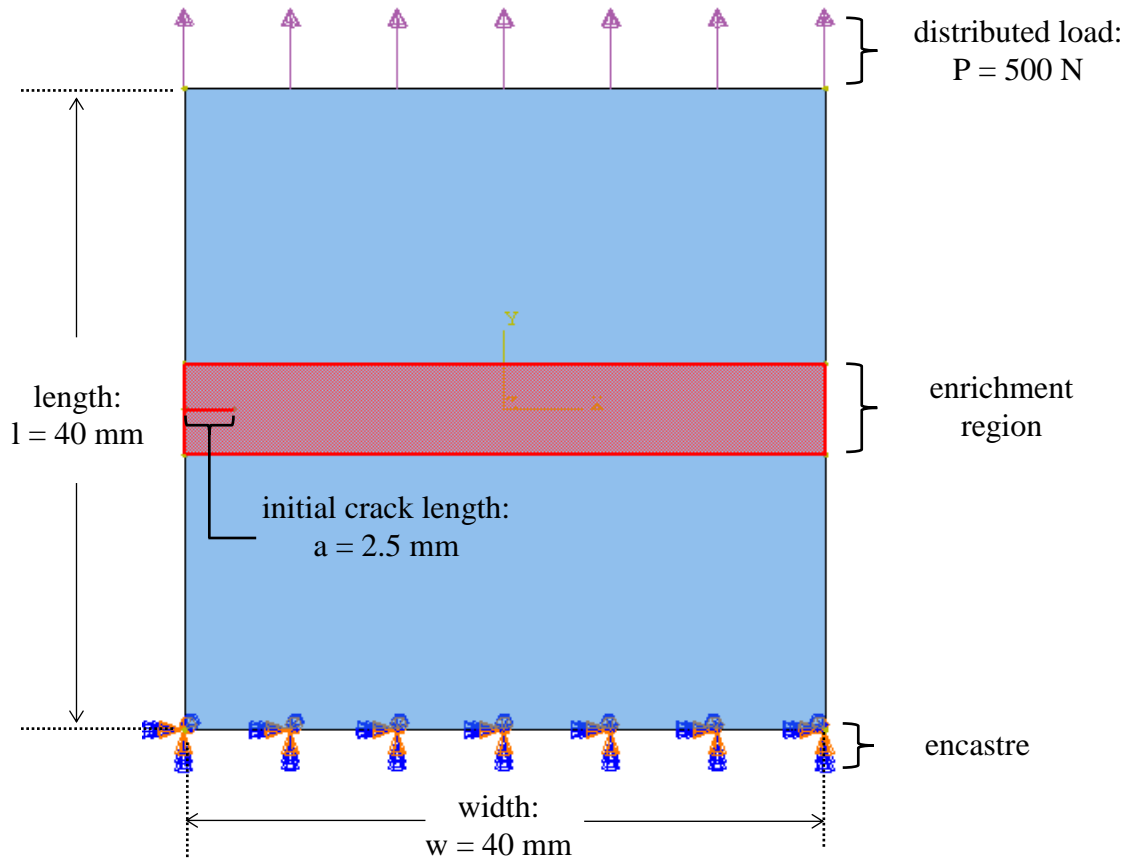
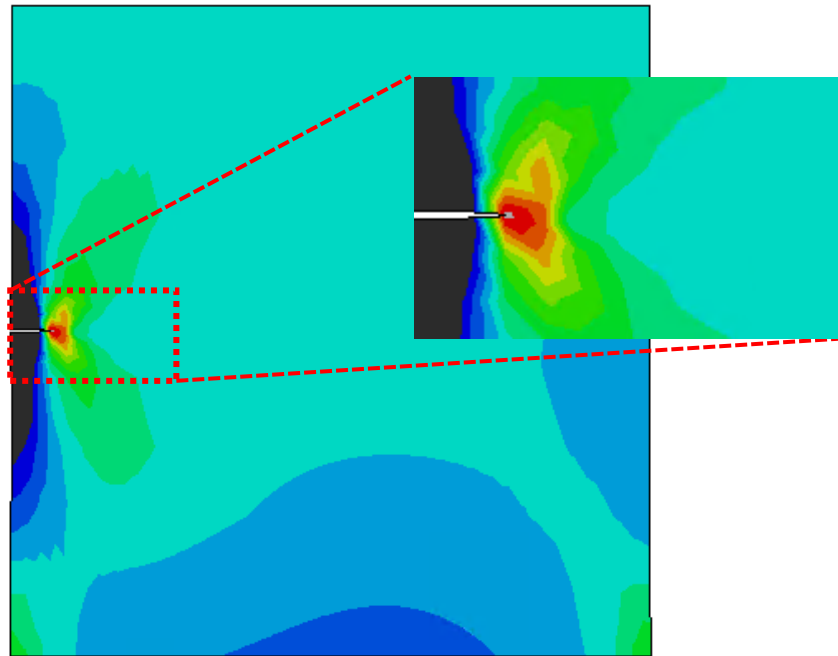
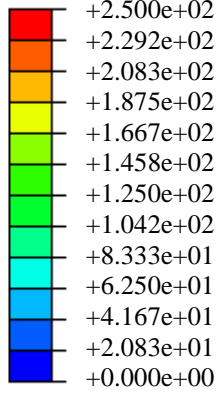


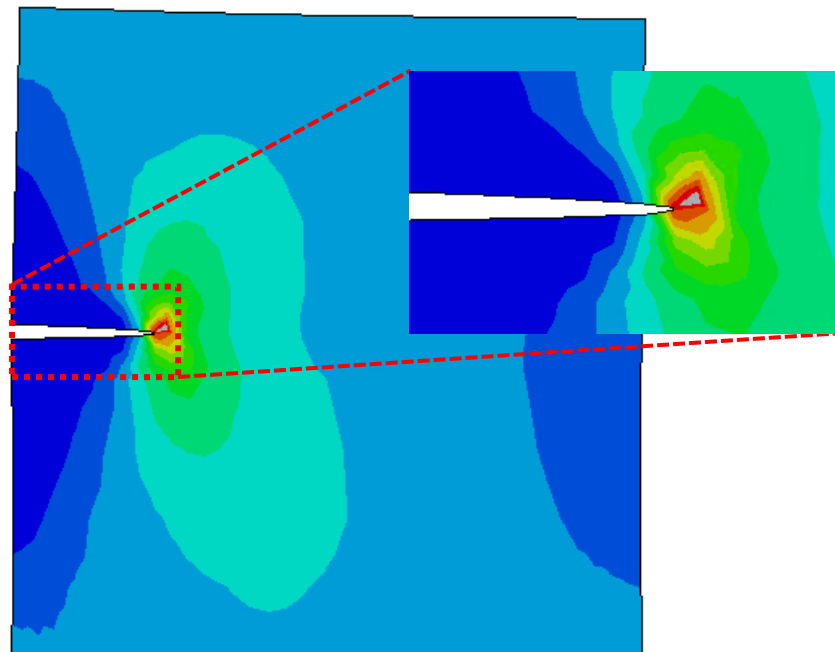
Figure 6. Abaqus simulation model: 40 mm by 40 mm plate single-edge notch tension test.

In Figure 7, we define t as the time of fracture at a specific crack length. The simulation shows that crack initiation occurred at the region near the crack tip where the local stress reached the maximum allowable principal stress at $t = 0.57$ s. The damage continued and repeated for a number of time increments until $t = 0.811$ s, where the crack length is 9.1 mm.

S, Max. Principal
(Discontinuities)



$t = 0.5719 \text{ s}$
 $a = 2.5 \text{ mm}$



$t = 0.8011 \text{ s}$
 $a = 9.1 \text{ mm}$

Figure 7. Crack propagation simulation using Abaqus: maximum principal stress within crack vicinity from initial rupture ($t = 0.57 \text{ s}$) to final crack length ($t = 0.80 \text{ s}$). Black region corresponds to stress less than 0 MPa while Gray region corresponds to stress greater than 250 MPa.

4.4 Finite Element Solution for J-integral

Here, we explore how the software discretizes the analytical solution of the contour J-integral from (2.16) which is beneficial in understanding how Abaqus implements numerical solution especially in Chapter VI and VII.

To discretize the domain form solution of energy release rate in (2.16), a 2×2 Gaussian integration is applied summing all the J-integral values for all elements, ne , on the region A^* [30].

$$J = \sum_{e=1}^{ne} \left\{ \sum_{g=1}^{ng} \left[\left[(\sigma_{ij} u_{j,1} - W \delta_{1i}) \frac{\partial q}{\partial x_i} \right] \det \left(\frac{\partial x_j}{\partial \xi_k} \right) \right]_g w_g \right\}_e \quad (4.4)$$

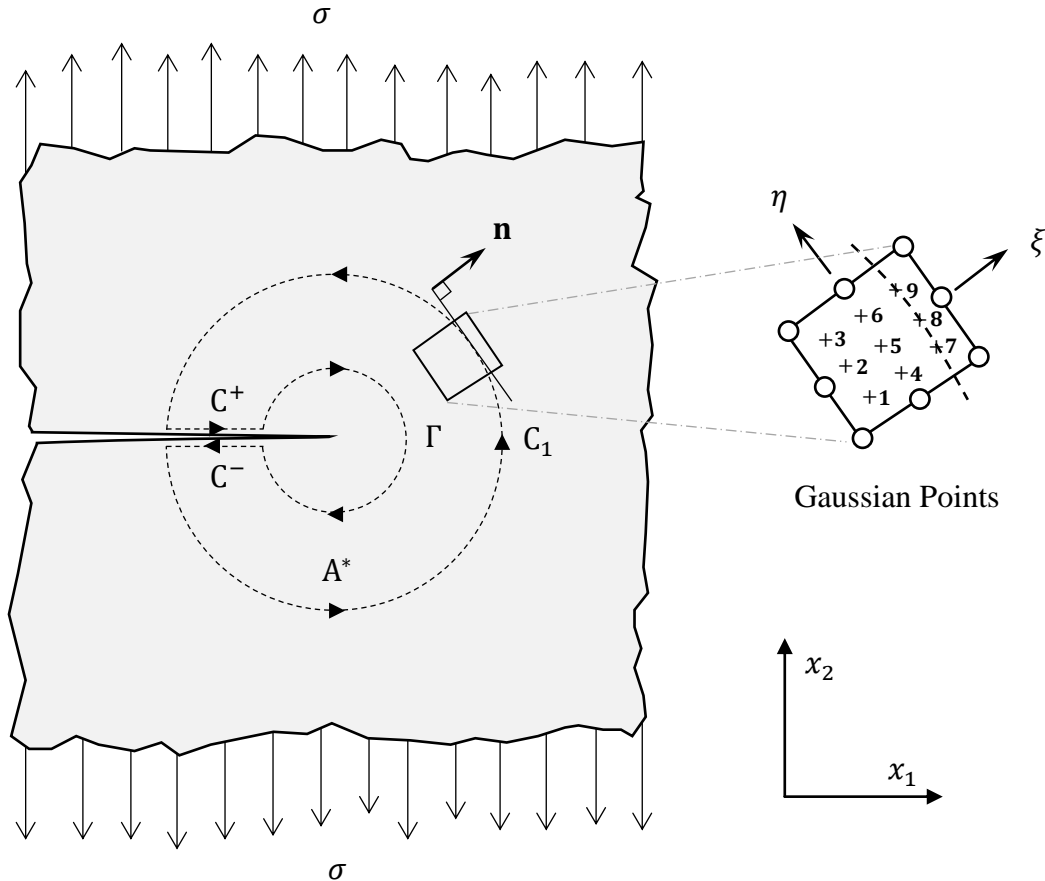


Figure 8. Numerical integration path to evaluate J-integral (reproduced without permission) [43].

The values within the $\{ \}_g$ are evaluated at Gauss points shown in Figure 8 and w_g is the Gaussian weight.

The spatial gradient of q and the nodal solution for strain energy, W from (4.4) are as follows [20, 27]

$$\frac{\partial q}{\partial x_i} = \sum_{i=1}^{N_{\text{nodes}}} \sum_{k=1}^2 \frac{\partial N_i}{\partial \xi_k} \frac{\partial \xi_k}{\partial x_j} q_i \quad (4.5)$$

$$W = \frac{1}{2} [\sigma_{11} u_{1,1} + \sigma_{12} (u_{1,2} + u_{2,1}) u_{1,1} + \sigma_{22} u_{2,2}] \quad (4.6)$$

Given that J-integral is calculated through finite element method, (2.8) and (2.12) is combined to form a solution for stress intensity factor [16] which leads to

$$K_I = \sqrt{J E}. \quad (4.7)$$

CHAPTER V

Auxetic Structure

5.1 Background

In this chapter, we describe the geometry of the auxetic structures analyzed in this work. We differentiate between auxetic test samples and a unit cell that represents the whole structure. We also define some geometric parameters that are used to change the characteristics of the auxetic material.

5.2 Specific Test Sample

We have examined the auxetics that have two-dimensional symmetric, orthogonal void pattern such as ellipse, slot, and stop-hole. We also included circle pattern as point of comparison to the other models (non-auxetic structure). As shown in Figure 9, the specimens are similar to the conventional dog-bone test material, the only difference is that they consist of pores that are purposefully located at the middle section of the sample. The blank specimens are 260 mm by 44 mm and 2 mm in thickness. Each grip section (top and bottom) has 50 mm distance from the end. The equivalent number of orthogonal void patterns is 20 and each has equal distance from one another.

5.3 Periodic Structure

We also analyzed representative volume elements (RVE) that are used to model a very large object with array of repeating structure. In Figure 9, each test model has its corresponding RVE and we based the structure of the unit cell by getting parameters at the very center of the plate. We modeled 10 mm by 10 mm RVE plates with vertical void at the center and corners of the cell; whole horizontal voids are found at the middle section of each edge.

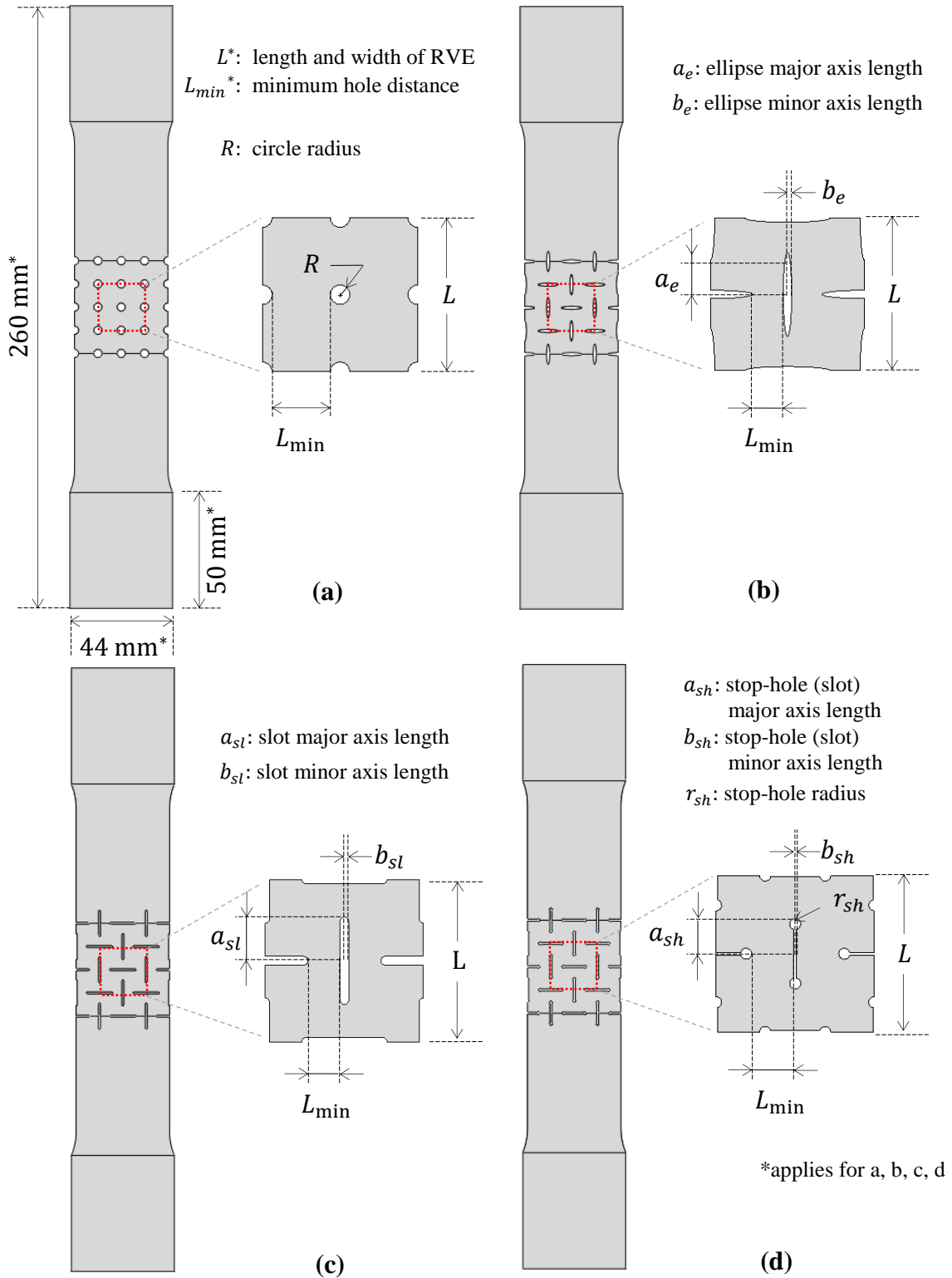


Figure 9. Whole test model of auxetic materials with their corresponding representative volume element (RVE), (a) Circle Void, (b) Elliptical Void, (c) Slot Void, (d) Stop-hole Void

5.4 Axis Ratio

In parallel to the previous studies [16], for example on Figure 9b, the major and minor axis length are specified and can be further relate the two dimensions to obtain the aspect ratio of the ellipse.

$$AR_e = \frac{a_e}{b_e} \quad (4.1)$$

AR_e was methodically altered, from previous investigation and in this study, to acquire the desired porosity of the RVE. In a similar manner, the ratio of the geometry of the other sample is also included. We have specified that the ratio of the slot length to the slot width as AR_{sl} and the ratio of the stop-hole void effective length (formed by combination of slot and circle voids) to the stop-hole slot as AR_{sh} .

$$AR_{sl} = \frac{a_{sl}}{b_{sl}} \quad (4.2)$$

$$AR_{sh} = \frac{a_{sh}}{b_{sh}} \quad (4.3)$$

5.5 Porosity

Porosity, ψ , is the fraction of the void area over the total area of the material (conventionally ranges from 0 to 100%). As an input parameter in the numerical model, ψ is considered as the initial blank area of the RVE divided by the total area of the void.

$$\psi = \frac{A_{\text{void}}}{A_{\text{total}}} \quad (4.4)$$

On the succeeding section, changing the porosity will be presented and its effect to the fatigue crack propagation parameters such as in stress intensity factor.

CHAPTER VI

Numerical Analysis

6.1 Background

In this chapter, the analysis for obtaining the value of stress intensity factor is examined from the work of Javid et. al. [16]. We briefly summarize the previous study on acquiring the J-integral with the use of Abaqus. We have both replicated some of Javid's main results, but also expanded on them to include parameter studies on porosity and minimum hole distance as well as XFEM analysis. We also introduced a new approach of using finite boundary condition in analyzing the model of actual test samples. This section is important since the methodology of numerical result of J-integral will be used in the calculation in Chapter VII.

6.2 Numerical Methods on J-integral

Since the samples that were tested are plates with 1 mm thickness, plane-stress 2D elements were used to simulate the crack propagation using XFEM. In particular, 4-node bilinear plane-stress quadrilateral elements were implemented to discretize the model (Abaqus Code: CPS4). For the materials, Javid et. al used stainless steel as subject with Modulus of elasticity of 193 GPa and Poisson's ratio of 0.33, which we also use here. There are two different steps in the procedure: first was to employ XFEM option on the test specimen to verify the direction of the crack through a uniaxial static analysis, second was to approximate K_I by gathering the J-integral results within a crack length increment. For the XFEM, methods from Chapter IV were implemented, the difference is that apart from actual geometry of the auxetic material, double notch initial crack was place on the middle left and right void of the RVE as shown in Figure 10. The traction separation was selected as a damage option and the damage criterion was based on the maximum principal stress, $\sigma_{max}^0 = 250$ MPa. In addition, the value of strain energy release rate was input as a

parameter, where $\mathcal{G} = 4 \text{ J/mm}^2$. In brief, the maximum allowable principal stress was used for damage initiation, while the strain energy release rate was used to apply a power law energy model for damage evolution.

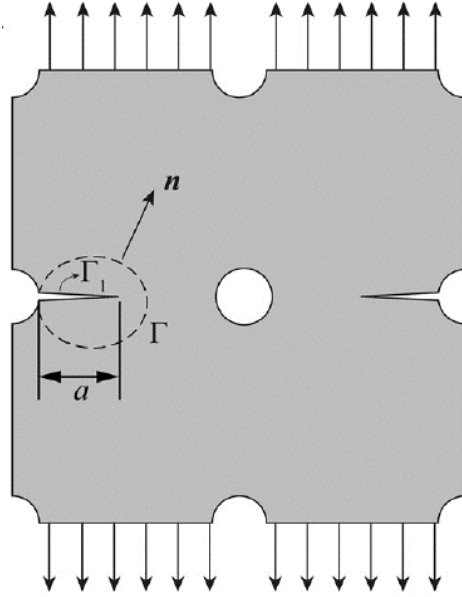


Figure 10. Double notch initial crack of an RVE with circular void subjected into tensile test (reproduced without permission) [16].

For the J-integral, the same feature of crack model was used (Figure 10) but instead of using enrichment functions, 65 distinct models of RVE were created each with increment of cracks between $0.1L_{\min}$ and $0.9L_{\min}$ formulated as follows [16]:

$$a = 0.1 \frac{a}{L_{\min}} + j0.8 \frac{a}{L_{\min}}, \quad j = 0,1,2, \dots 64 \quad (6.1)$$

The assumption on the models is to have a horizontal crack at each increment along x_1 direction where the maximum principal stress is located at the crack tip. The minimum hole spacing L_{\min} , was used to normalize the crack length in (6.1) since it is the maximum length of the crack between the two holes.

The RVE models are subjected to periodic boundary conditions with applied uniaxial tensile strain of 0.002. An interaction option was selected to perform calculations of the J-integral at the crack-tip section of the RVE. After gathering the result of the contour, (2.20) was used to evaluate the K_I , and Paris Law was used (Equation 2.10) to approximate the value of N_f .

The result showed from the reference paper that the crack evolution for the circular void model have higher values of stress intensity factor compared to the ellipse void model. From Figure 11, having 5% porosity applied for all models, it is illustrated that the behavior of the circular void model has positive slope which means that as the crack propagates the stress intensity factor increases. On the other hand, the stress intensity factor decreases with crack length for elliptical voids. Based on Paris Law from (2.10), ΔK is inversely proportional to the number of cycles, N_f . Additionally, ΔK is equal to the difference between the stress intensity factor at a specific crack length, K_p , and initial stress intensity factor, K_0 . K_0 is assumed to be equal to zero, therefore ΔK is equal to K_p . Thus, the elliptical void model has a higher value of N_f compared to the circular void model which is in agreement with the experimental results of the reference study [16]. The procedure of Javid et. al was also followed for the normalization of the stress intensity factor. The computed value of the stress intensity factor from (4.7), also considered as the maximum stress intensity factor the tip of the crack (K_{\max}), is divided by the stress intensity of the bulk material which is equal to G_0/\sqrt{L} , where G_0 is the strain energy release rate of the bulk material. This was implemented so that the RVEs, having different geometries, were transformed into unit form for ease of comparison [16].

We have replicated the aforementioned procedure as shown where the multiple points fit the plot of reference models by applying periodic boundary condition (PBC). An additional two models were included on the plot. 5% porosity was used for slot void and stop-hole void models based on their corresponding geometry. Figure 11 also shows that these behave almost identically with the elliptical void model in which the normalized K decreases as $\frac{a}{L_{\min}}$ increases. Note that the method of obtaining the contour J-integral was

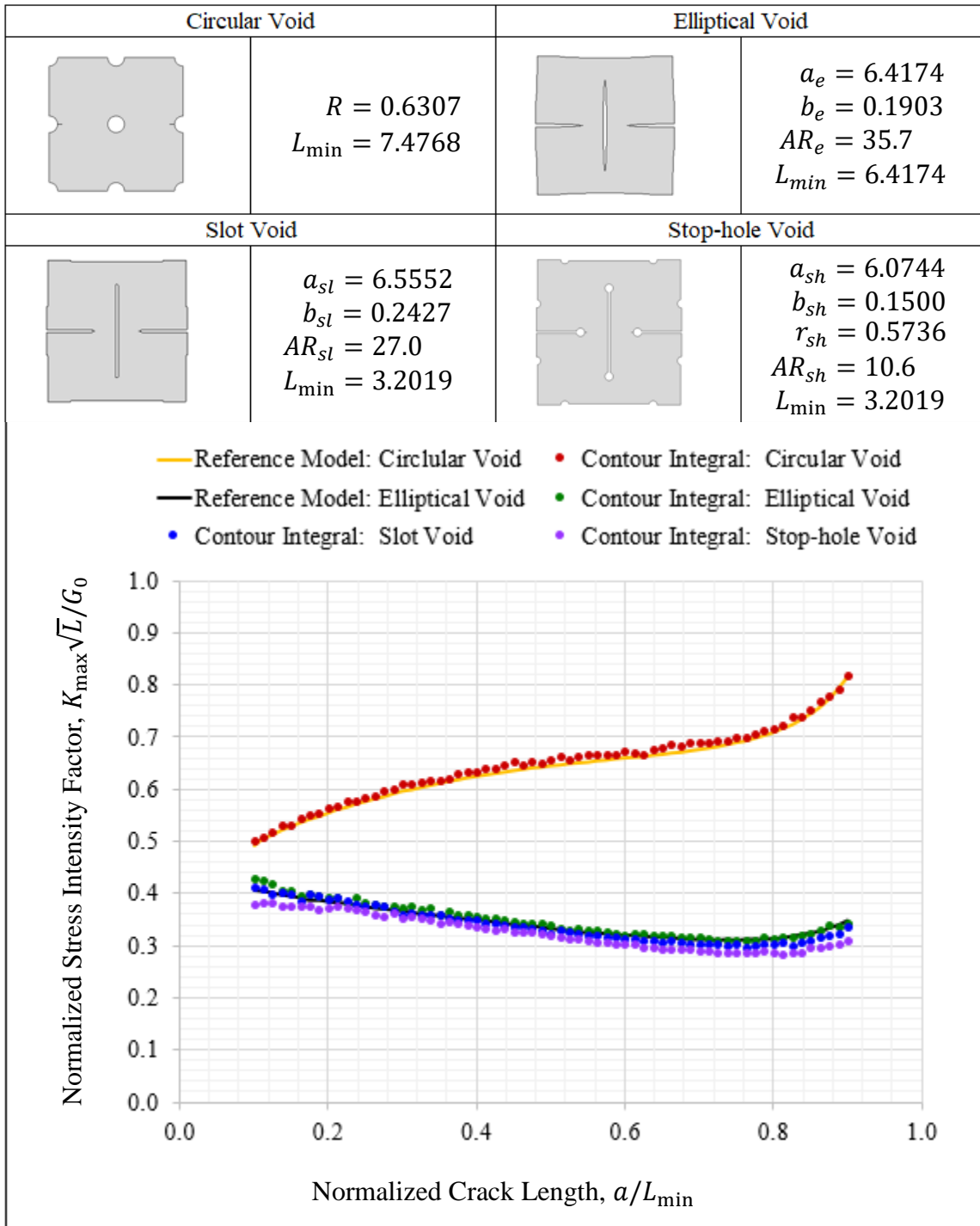


Figure 11. Evolution of normalized stress intensity factor along the normalized crack length. Comparison of the 5% porosity reference data model [16] to the calculated model of RVEs under **periodic boundary condition** using Abaqus.

implemented on Abaqus and we have created a Python script (Appendix A.1) to automatically generate the PBC to each model. Apart from the existing method, another approximation was implemented by using finite boundary conditions (FBC). In Chapter VII, we compare finite element samples to their corresponding RVEs models and the following analysis verifies that Javid's procedure works with FBC. In this procedure, a displacement of 0.01 mm was applied on the top and bottom edges of the RVEs which is computed by multiplying the center to center distance, L , with half of applied uniaxial tensile strain load. A similar procedure was applied to the RVEs, where J-integral results were calculated based on 65 models with increasing crack length based on the crack increment in (6.1). Figure 12 shows that this method also approximates the reference model. Compared to the reference model, the circular void model has greater values of stress intensity factor until the point of inflection at $\frac{a}{L_{\min}} = 0.66$. For the elliptical, slot, and stop-hole void models, although the decline of stress intensity was observed similar to the

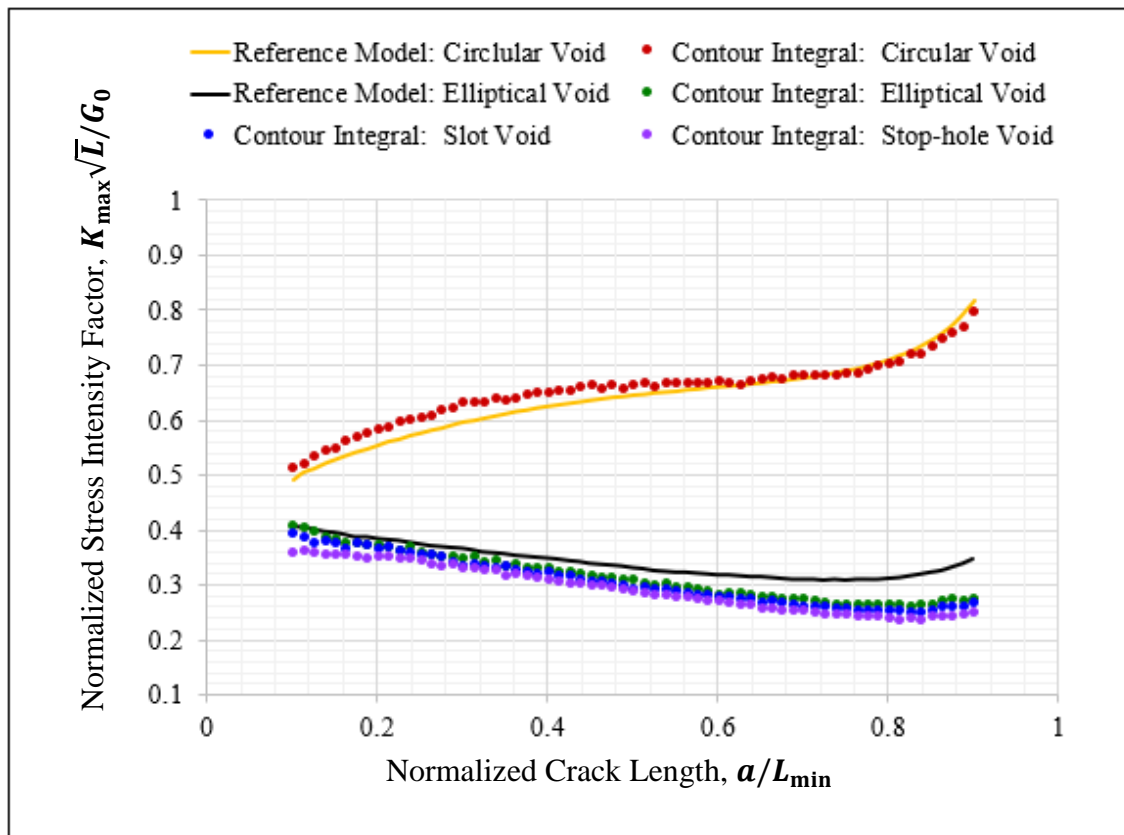


Figure 12. Evolution of normalized stress intensity factor along the normalized crack length. Comparison of the reference data model [16] to the calculated model of RVE under **finite boundary condition** using Abaqus.

ellipse void reference model, the points deviates from the reference as the crack is extended.

6.3 Variation of Geometry

To observe further the behavior of the stress intensity factor vs. crack length, AR were varied while holding either the porosity or minimum separation constant. The variations were divided into parts based on the constants that were fixed, porosity, and minimum hole spacing. Furthermore, we have also computed the effective Poisson's ratio of each model to see its relation to the stress intensity factor during the crack evolution. We also changed the range of the crack evolution by using 1% to 99% of L_{\min} . Using this method, the crack initiations and crack propagations before total failure are observed.

6.3.1 Constant Porosity

For the ellipse void model with constant porosity of 5% in Figure 13, we have altered the model by increasing AR_e in increments of 3 from Model B to Model J. The circular void model, $AR_e = 1$, was also included on the plot as reference of comparison to the other models. The models with negative Poisson's ratio was also highlighted to distinguish them from the other models.

A slow decrease in normalized K between 0.1 and 0.8 $\frac{a}{L_{\min}}$ was observed on models E to J and sudden increase in K after $\frac{a}{L_{\min}} = 0.8$. The plot also showed that at initial point, $\frac{a}{L_{\min}} = 0.01$, the model with circle void has the lowest value of normalized stress intensity factor compared to the other models but increases rapidly as the crack length grows. Recalling Paris Law, this implies that crack initiation and the initial crack growth stage is relatively slow, but ultimately becomes faster than that in other geometries, ultimately giving the circular void configuration a shorter lifetime. It is depicted that after $\frac{a}{L_{\min}} = 0.044$, Model J, having the largest value of negative Poisson's ratio, was observed to have the lowest values of stress intensity factor followed by models I to E. Based on that, for ellipse void model, we have observed that, as the negative Poisson's ratio increases, K magnitudes at each point from model decreases.

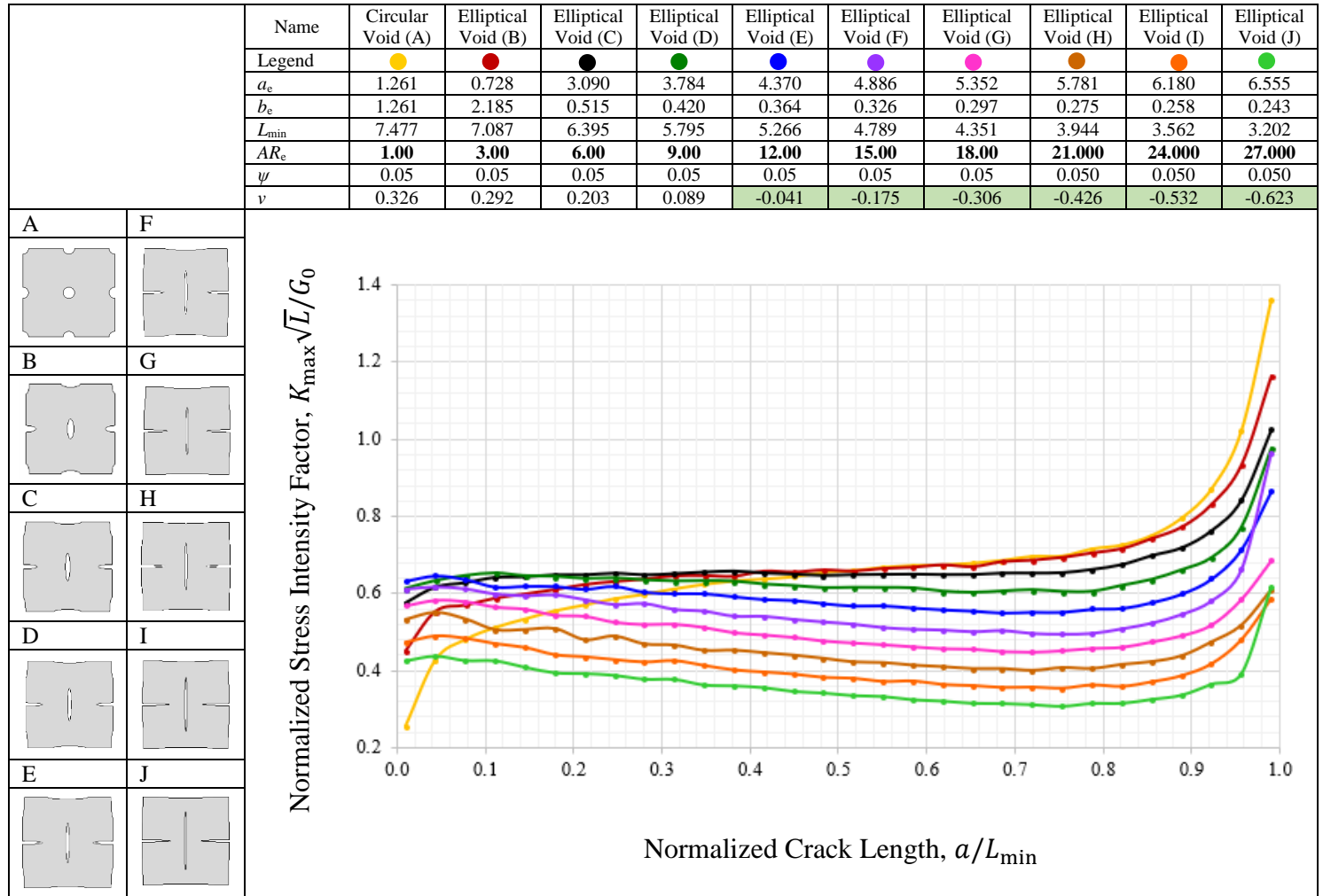


Figure 13. Evolution of normalized stress intensity factor along the normalized crack length. Variation of RVE with 5% porosity **elliptical void** by increasing AR_e in increments of 3 (from Model B to Model J). Center-to-center length of the RVE, $L = 10$ mm.

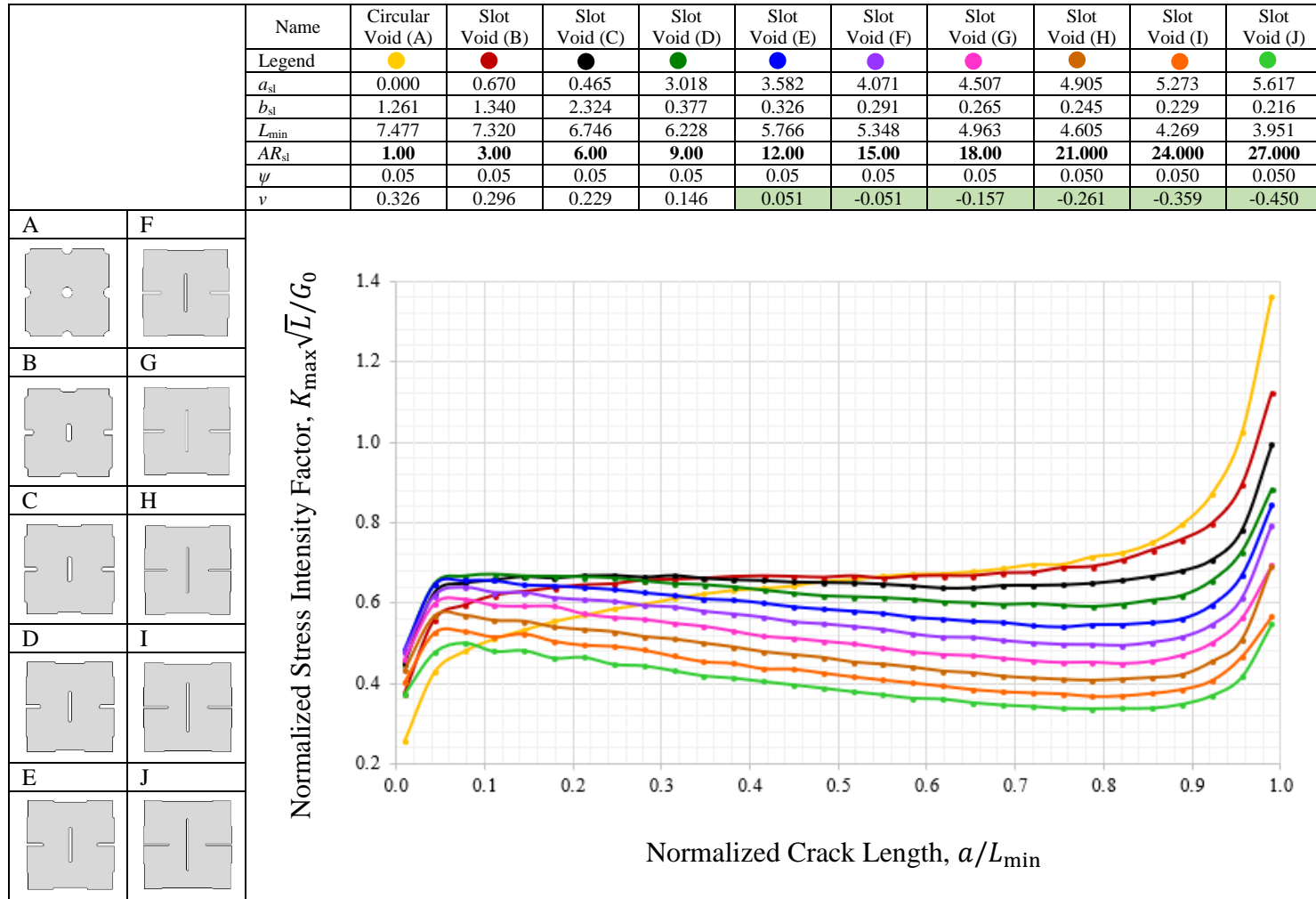


Figure 14. Evolution of normalized stress intensity factor along the normalized crack length. Variation of RVE with 5% porosity **slot void** by increasing AR_{sl} in increments of 3 (from Model B to Model J). Center-to-center length of the RVE, $L = 10$ mm.

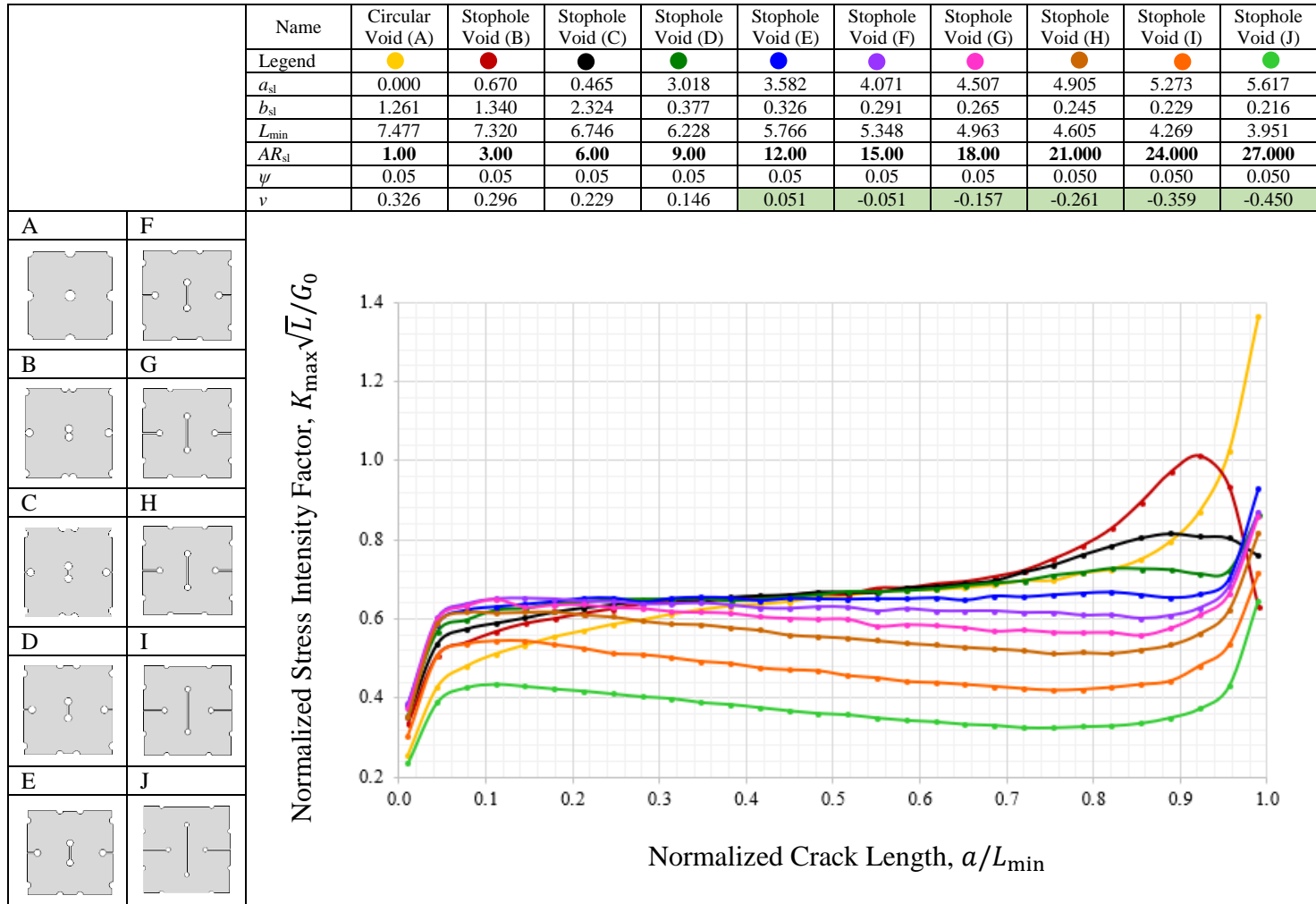


Figure 15. Evolution of normalized stress intensity factor along the normalized crack length. Variation of RVE with 5% porosity **stop-hole void** by increasing AR_{sl} in increments of 3 (from Model B to Model H) and increments of 6 (from Model H to Model J).

For the slot void models with constant porosity of 5% in Figure 14, AR_{sl} was modified in increments of 3 from Model B to J. The starting points of the model is relatively lower than the models with elliptical void. These models have higher values of N_f compared from the previous models which means that elliptical void model crack propagate faster than the slot void model. On the other hand, parallel to the behavior of the elliptical void, the slot void has decreasing value of normalized K from $\frac{a}{L_{min}} = 0.078$ to 0.889. Model J which has the largest value of negative Poisson's ratio is found to have the lowest values of normalized K , then followed by models I to F. It is also shown that circle void model, starts at the lowest portion of the graph but evolves rapidly until $\frac{a}{L_{min}} = 0.990$.

For the stop-hole void model variation in Figure 15, the RVE were modified through changing AR_{sh} with increments of 3 from models B to G and increments of 6 from models G to J. Like the slot void model, the first points of the stop-hole RVEs starts with lower values of normalized K compared to the elliptical void model. The last three models (H-J), with negative Poisson's ratio, are depicted to have the lower values of normalized K . Applying Paris Law, Models H to J implies that they have higher values of N_f in boundaries between $\frac{a}{L_{min}} = 0.078$ to 0.821. This also means that they propagate slower than other models.

Most of the stress intensity factor trend in the stop-hole void model, starts at the lowest point then exhibits an increasing trend until it climbs to its highest point at $\frac{a}{L_{min}} = 0.990$. However, for models B and C, their peaks are found at $\frac{a}{L_{min}} = 0.922$ and 0.899 respectively. A more detailed picture is shown in Figure 16, which shows the maximum principal stress distribution (ranging from 0 to 250 MPa) of B and C in comparison to model A. It is identified that the vertical void at the center of the RVEs B and C has high stress concentration on their stop-holes. Since the maximum principal stress of stop-hole models do not lie at $x_2 = 0$ (reference: (0,0) center of the RVE), the assumption of the crack direction is violated.

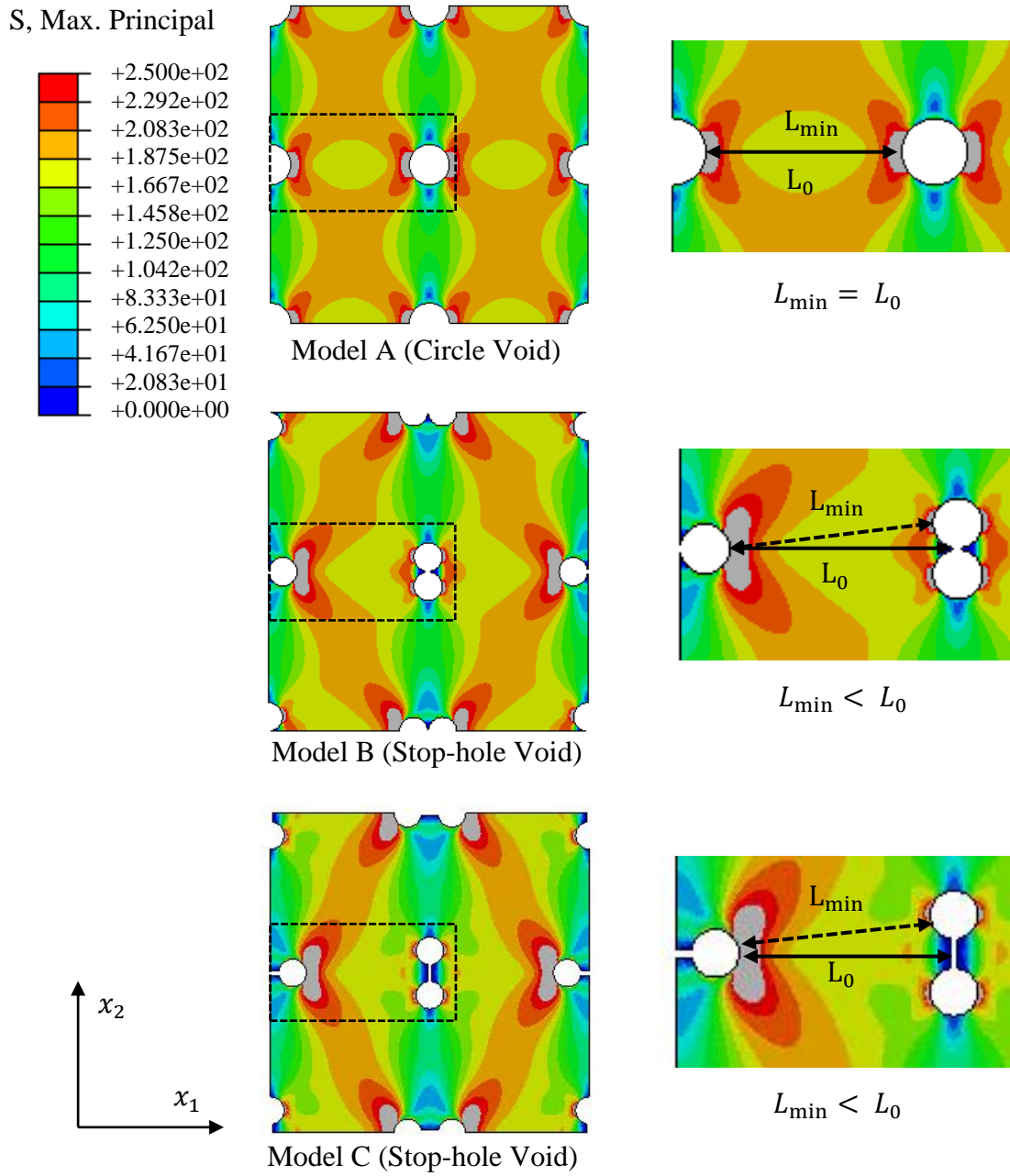


Figure 16. Maximum principal stress distribution of RVE circle void model A and stop-hole void models B and C. Gray colors show the areas of stress above maximum principal stress.

From Figure 16, we define L_0 as the horizontal hole-to-hole distance at $x_2 = 0$. We have observed that for circle void model, which satisfies the assumption of horizontal direction of crack, $L_0 = L_{\min}$. On the other hand, for models B and C, $L_0 > L_{\min}$, which violates the assumption of horizontal crack propagation.

6.3.2 Constant Minimum Hole Spacing

In this subsection, we have maintained the minimum hole spacing but change the porosity for every model by altering the AR . This also allows parameters a , b , and r to change. As a reference, we have selected one model with negative Poisson's ratio per void shape from Figures 13 to 15. The criteria of selection were based on the range of L_{\min} between 3 mm to 4 mm since we do not want L_{\min} to be too small that the distances of the holes are closer or too large that the range models that are computed has positive Poisson's ratio. From the models on the previous subsection, several satisfies these criteria, but we only selected just one reference. We produce variation by subtracting and adding increments of constant number from the reference. We denote the selected reference model based on its previous name and add a superscript 0 to it (e.g. J to J^0). In general, the models J^g , were denoted such that if $g = 0$ it represents the reference model and if $g = -3, -2, -1$, the models are associated with a decrease in the parameter of interest with respect to the reference while if $g = +3, +2, +1$, the models are associated with an increase in the parameter of interest with respect to the reference.

For the elliptical void model in constant minimum hole spacing of 3.943 mm (Figure 17), model J was selected from Figure 13 and set as reference then changed the AR_e by increasing (blue) and decreasing (red) the parameter in multiples of 5 with respect to the reference. In the plot, Model J^{-3} , having the largest porosity of 0.160 and smallest value of negative Poisson's ratio, is found to start at the lowest point in comparison to the other models but has the highest overall normalized stress intensity factor as the crack propagates. This is followed by models J^{-2} and J^{-1} , considering that they have higher porosity but relatively small values of negative Poisson's ratio, their normalized

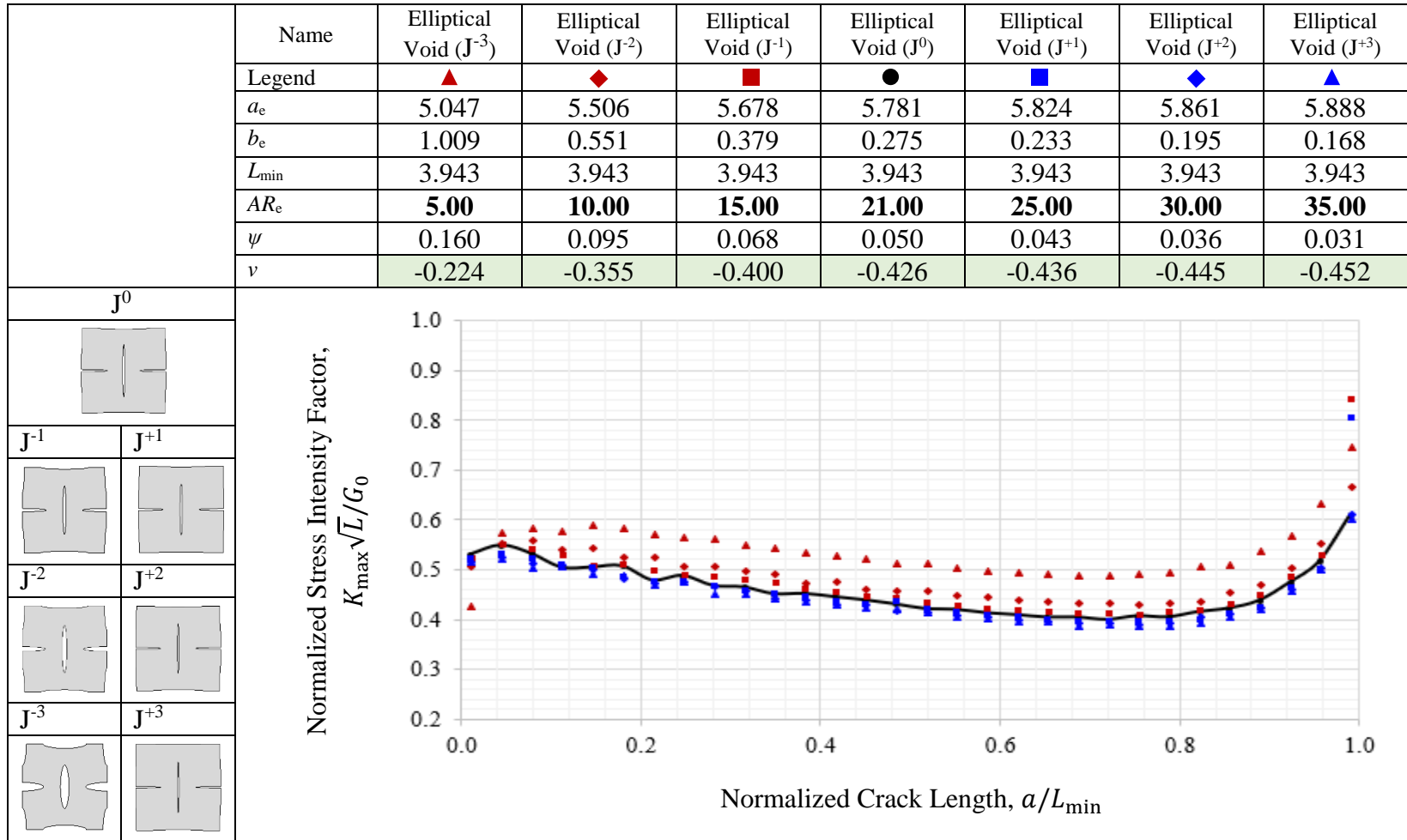


Figure 17. Evolution of normalized stress intensity factor along the normalized crack length. Variation of RVE **elliptical void** by increasing AR_e in increments of 5 (from Model J^{-3} to Model J^{+3}) in constant L_{\min} (3.943 mm). Center-to-center length of the RVE, $L = 10$ mm. The solid black line represents the reference model H^0 .

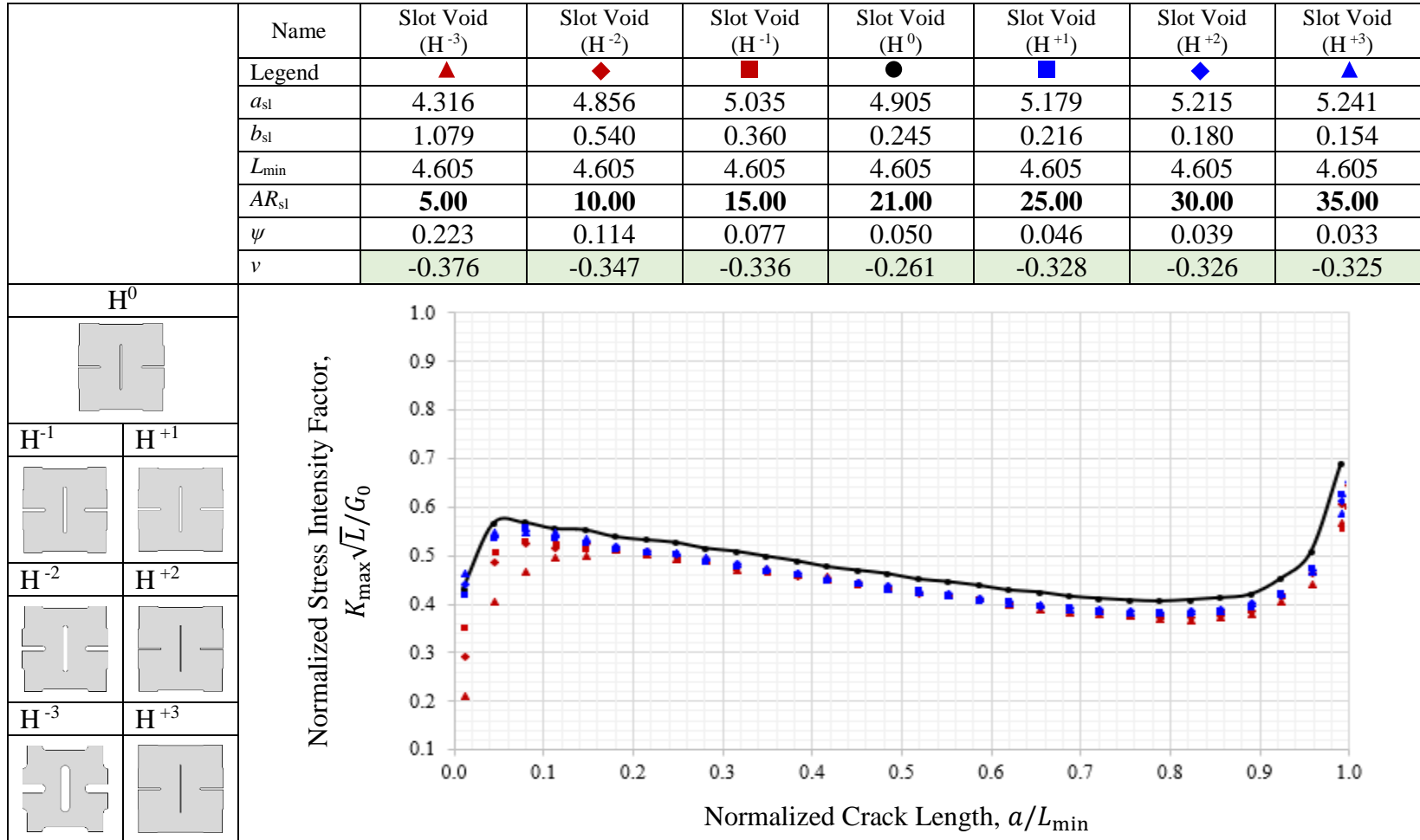


Figure 18. Evolution of normalized stress intensity factor along the normalized crack length. Variation of RVE slot void by increasing AR_{sl} in increments of 5 (from Model H^{-3} to Model H^{+3}) in constant L_{min} (4.605 mm). Center-to-center length of the RVE, $L = 10$ mm. The solid black line represents the reference model H^0 .

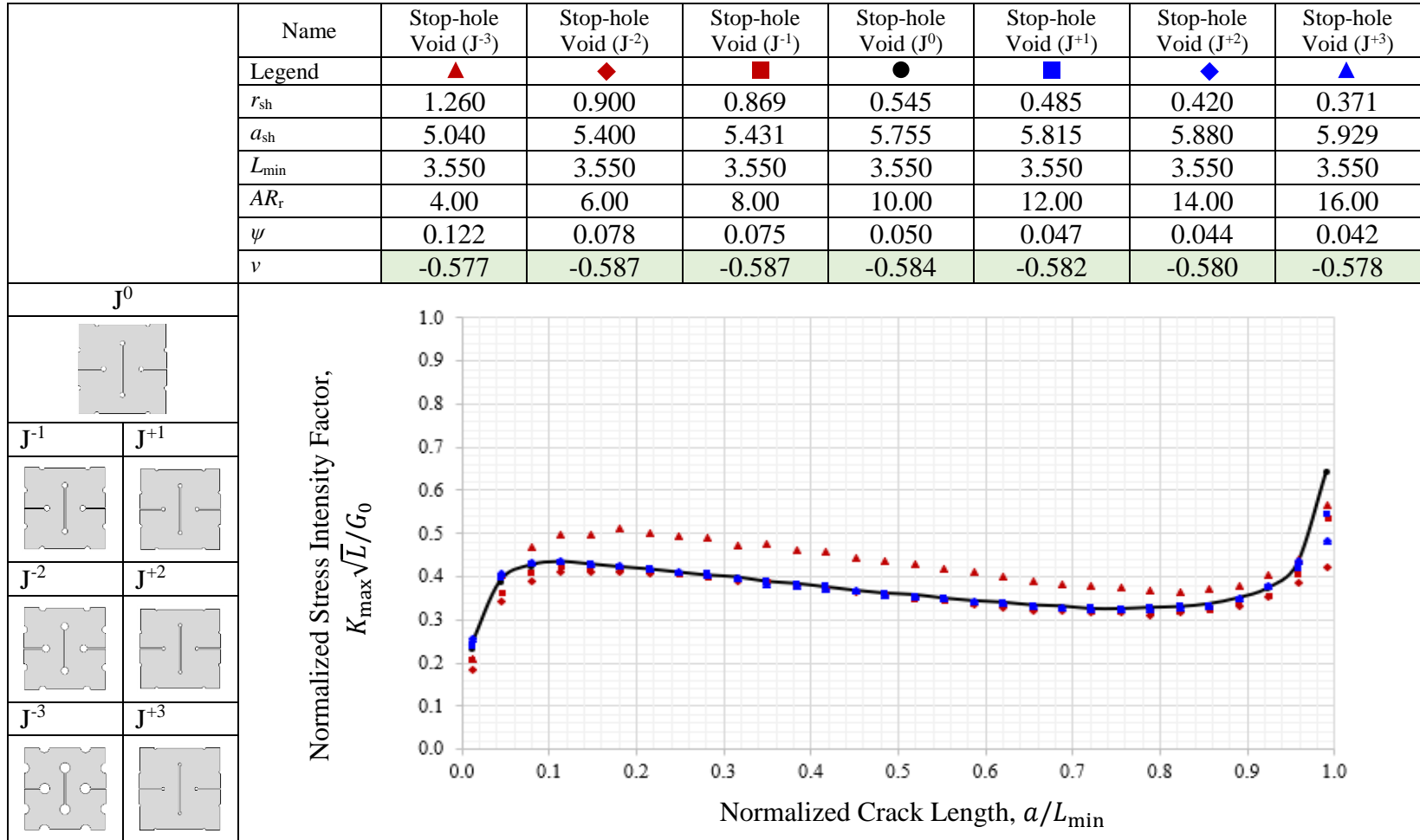


Figure 19. Evolution of normalized stress intensity factor along the normalized crack length. Variation of RVE **stop-hole void** by increasing AR_r in increments of 2 (from Model J^{-3} to Model J^{+3}) in constant L_{min} (3.550 mm). Center-to-center length of the RVE, $L = 10$ mm. The solid black line represents the reference model H^0 .

stress intensity factor is greater compared to J^0 . Models J^3 , J^2 , and J^1 have found to be at the lower level of J^0 . This implies that these models, having the relatively lower amount of porosity but larger negative Poisson's ratio, have the lowest amount of normalized stress intensity factor hence their crack propagates slowly compared to the other models.

For the slot void model in Figure 18, the constant minimum hole spacing of 4.605 mm was selected from model H of Figure 14 and set as reference model H^0 . From H^0 we have changed the value of AR_{sl} in multiples of 5 on models that have increased parameter and models with decreased parameter, similar in the RVE elliptical void. It is depicted on Figure 18 that the reference model, H^0 , with smallest value of Poisson's ratio have the highest values of stress intensity factors to the rest of the plot. Model H^{-3} , having the largest value of negative Poisson's ratio of -0.376, begins at the lowest point of the plot then had an increasing value of K until $\frac{a}{L_{min}} = 0.179$, decrease gradually as the crack length evolve to $\frac{a}{L_{min}} = 0.821$ where K increase until it reaches the highest point. Similar behavior was observed to the other models. The only differences are the points between $\frac{a}{L_{min}} = 0.010$ to 0.821, where Model H^{-3} is followed by Models H^{-2} and H^{-1} (with negative Poisson's ratio of -0.347 and -0.336 respectively). Models such as H^1 , H^2 and H^3 were identified to have higher starting point than H^{-3} , H^{-2} and H^{-1} models. Based on Paris Law, this proves that models have increased values of AR_{sl} with respect to H^0 models (red) tends to have initial crack in contrast to the models with decreased values of AR_{sl} (blue).

For the stop-hole void model in Figure 19, Model J, with constant minimum hole spacing of 3.550 mm and slot width of 0.15 mm, was selected as the reference model from Figure 15. A different approach was implemented to vary the geometry of the stop-holes, instead of using $AR_{sh} = \frac{a_{sh}}{b_{sh}}$ (Figure 8d) as a changing parameter we defined AR_r as the ratio of the slot length, b_{sh} and the stop-hole radius, r_{sh} . Then, we changed AR_r by increasing (red) and decreasing (blue) the values in multiples of 2 from the reference model where $AR_r = 10$, hence altered the stop-hole radius but maintaining the values of L_{min} and b_{sh} . It is shown in Figure 19 that models with lower AR_r from the reference J^0 such as J^{-3} ,

J^{-2} and J^{-1} have the lowest starting point of stress intensity factor. However, Model J^{-3} with the smallest value of negative Poisson's ratio and highest porosity, is observed to have the highest overall amount of stress intensity factor as the crack evolves. The models J^{+1} , J^{+2} and J^{+3} have values of negative effective Poisson's ratio near to the reference J^0 and their plots demonstrates to be approximately equivalent to J^0 .

CHAPTER VII

Results and Discussion: Comparison to the Experimental Data

7.1 Background

In this chapter, we apply the numerical modeling procedure developed in previous chapters to analyze the fatigue experiments on low porosity metallic structures done by Francesconi et. al. [17]. In particular, the investigation was a comparison of thin plates with circular and stop-hole voids, which have non-auxetic and auxetic behavior, respectively. These specimens were subjected to tensile sinusoidal cyclic load and their fatigue fracture behavior was observed. The author captured strain contour maps using optical digital image correlation (DIC). It was determined that the test subject with stop-hole void has higher fatigue life compared to the specimen with circular void. From crack initiation, propagation to rupture, the experimental result showed that non-auxetic structure had a faster rate of crack evolution in comparison to the auxetic. While this test examination result gives a favorable result to a material with auxetic pattern, numerical analysis is useful to support such conclusion and try to explain the phenomenon. Therefore, in this chapter we have applied methods of finite element analysis, both XFEM and contour J-integral analysis, to simulate the actual crack evolution of the test specimen as well as to compare with the experimental results of the fatigue behavior.

7.2 Experimental Data and Results

In this section, we provide a brief summary of the experiment and results. The material that was used for the plate samples were 260 mm by 40 mm Aluminum 6060 -T6 with 2 mm thickness. The Young's modulus of the material is 65.4 GPa and its Poisson's ratio is 0.32. Also, based on the stress strain curve of the specimen, the yield strength is 195 MPa and the ultimate tensile strength is 216 MPa. The experimental samples have circular voids with radius, R , of 1.784 mm while the stop-holes have the following

dimensions: $a_{sh} = 4.625$ mm, $a_{sh} = 450$ mm, and $r_{sh} = 0.625$ mm. Both were specifically fabricated to acquire a 10% porosity.

For the fatigue test, a mode I load-controlled, sinusoid cyclic type of loading was applied to each specimen. Since the two different whole patterns lead to different effective material properties, the loads that were applied to each sample were calculated based on several factors affecting the fatigue test such as geometrical features, material, fabrication and stress concentration [17]. This was done to make fatigue comparison as “fair” as possible. Based on the author’s computation, the applied load for the laminate with circular void is 6050 N while the applied load for the laminate with stop-hole void is 3505 N. In addition, the total number of fatigue cycles were tuned to have 50,000 cycles and 68,000 cycles. To compare the behavior of the crack, the controlled final cycles were used to normalize the number of cycles at each phase of the crack propagation.

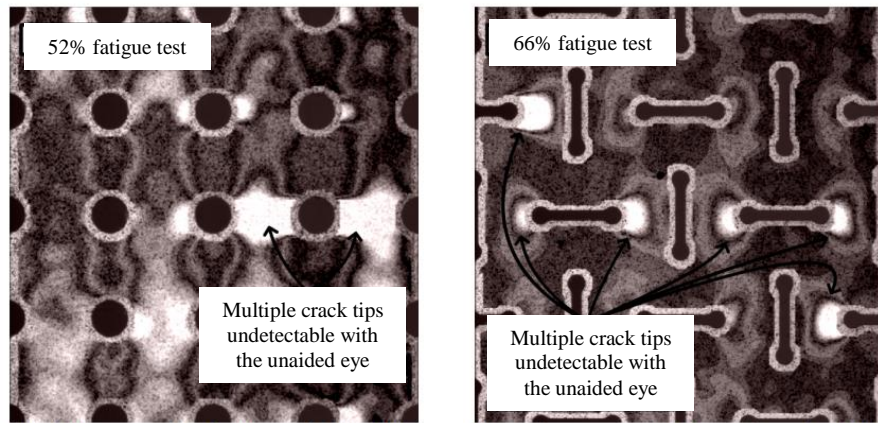


Figure 20. Contour maps of the Lagrangian strains from the DIC of the non-auxetic (left) and non-auxetic samples (right) [17].

The result showed from the DIC that significant strain concentrations were observed between 25% and 30% of each cycle. The crack initiated at 52% of the total cycles for the specimen with circular void while at 66% of the total life cycles for the specimen with stop-hole void. A contour map on Figure 20 illustrates the crack initiation of the samples.

7.3 Comparison to the Numerical Data to the Experimental Data

Using the actual material properties and the loading conditions from the experiment, we implemented the XFEM procedure to verify the path of the crack which was used to support the assumption for the computation of the J-integral. We performed a static analysis with dimensions identical to the plates that were tested.

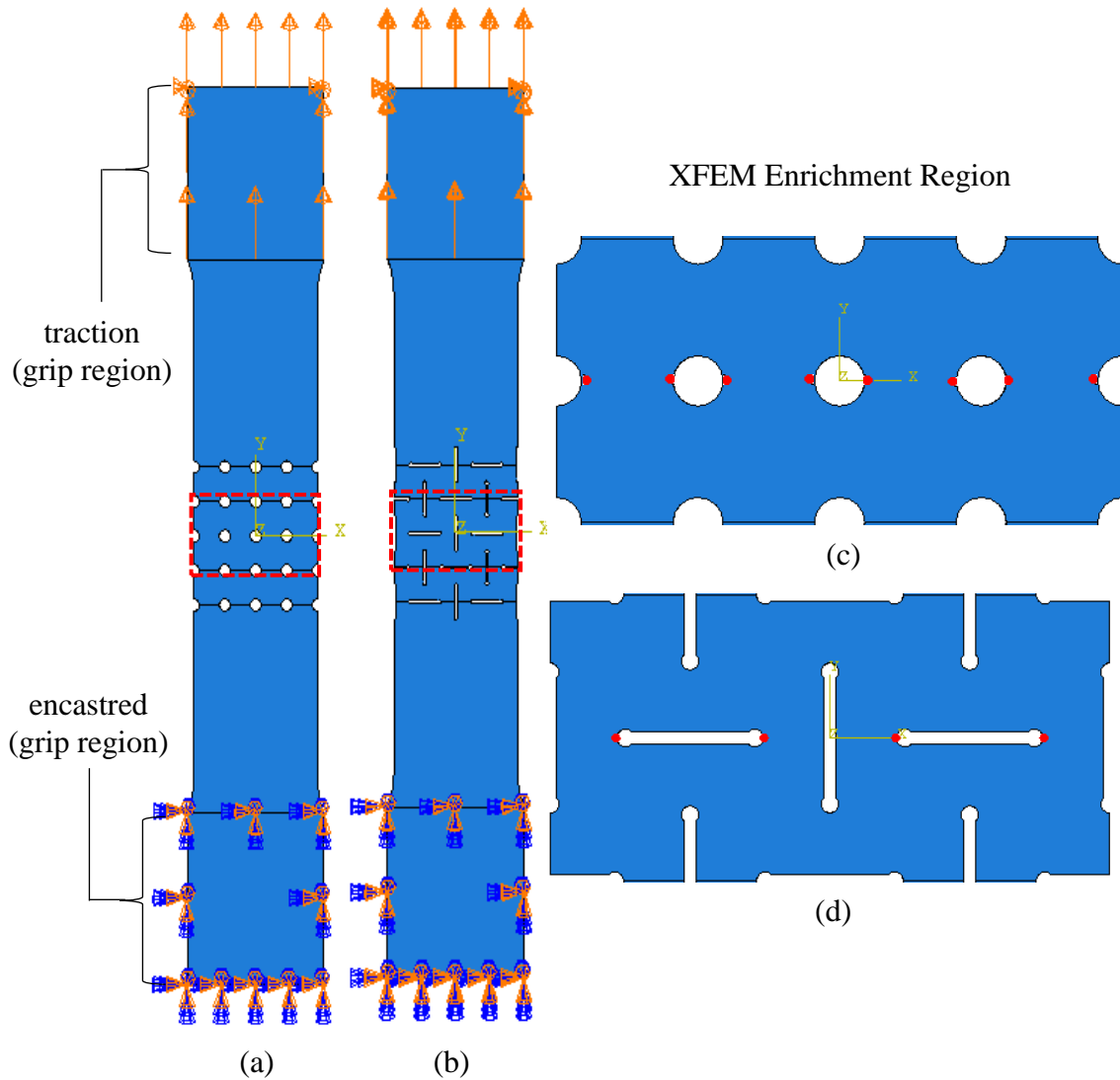


Figure 21. Abaqus assembly diagram for fatigue test simulation of plate with circular void pattern (a) and plate with stop-hole void pattern (b). Magnified section (XFEM enriched region) of specimen with circle void (c) and with stop-hole void (d).

We modeled two-dimensional plate under plane-stress condition with applied tensile displacement of 6.4 mm computed from the stress-strain curve of the material. To ensure an accurate result, we implemented 8-node biquadratic plane-stress quadrilaterals (Abaqus Code: CPS8) with 0.1 mm seed mesh for the enrichment region while we used 1 mm seed mesh for the rest of the parts (Figure 21). As shown in the reference configuration, we also included the initial cracks with 1% size of L_{\min} (see red highlights in Figure 21c and Figure 21d). The basis of the locations of initial crack were the maximum stress is located when the specimens were simulated in static analysis.

In the simulation, we denote t which indicates the time for an arbitrary crack length, and it ranges from 0 to t_f , ($0 \leq t \leq t_f$), where t_f is the time when the crack has completely propagated through the specimen.

The maximum principal stress contour map from Figure 22 shows magnified section of the whole specimen with circular void pattern. These magnified sections were the region in which XFEM enrichment was implemented. With initial crack deliberately placed at feasible location, the crack initiated at $t = 0.32t_f$ before total rupture where the maximum principal stress was located. In the simulation, the crack continuously grew horizontally while the stress surrounding the crack increases its area from $t = 0.87t_f$ to $t = 0.97t_f$.

Equivalent to the contour map of the circle model, the simulation for the stop-hole model shows that the crack also grew at the location where the maximum principal stresses were concentrated. The XFEM simulation showed that, in the enriched region the crack evolved at $t = 0.24t_f$. Then it propagated with increasing stress concentration around the crack region, and this was observed between $t = 0.52t_f$ and $t = 0.94t_f$ (Figure 23).

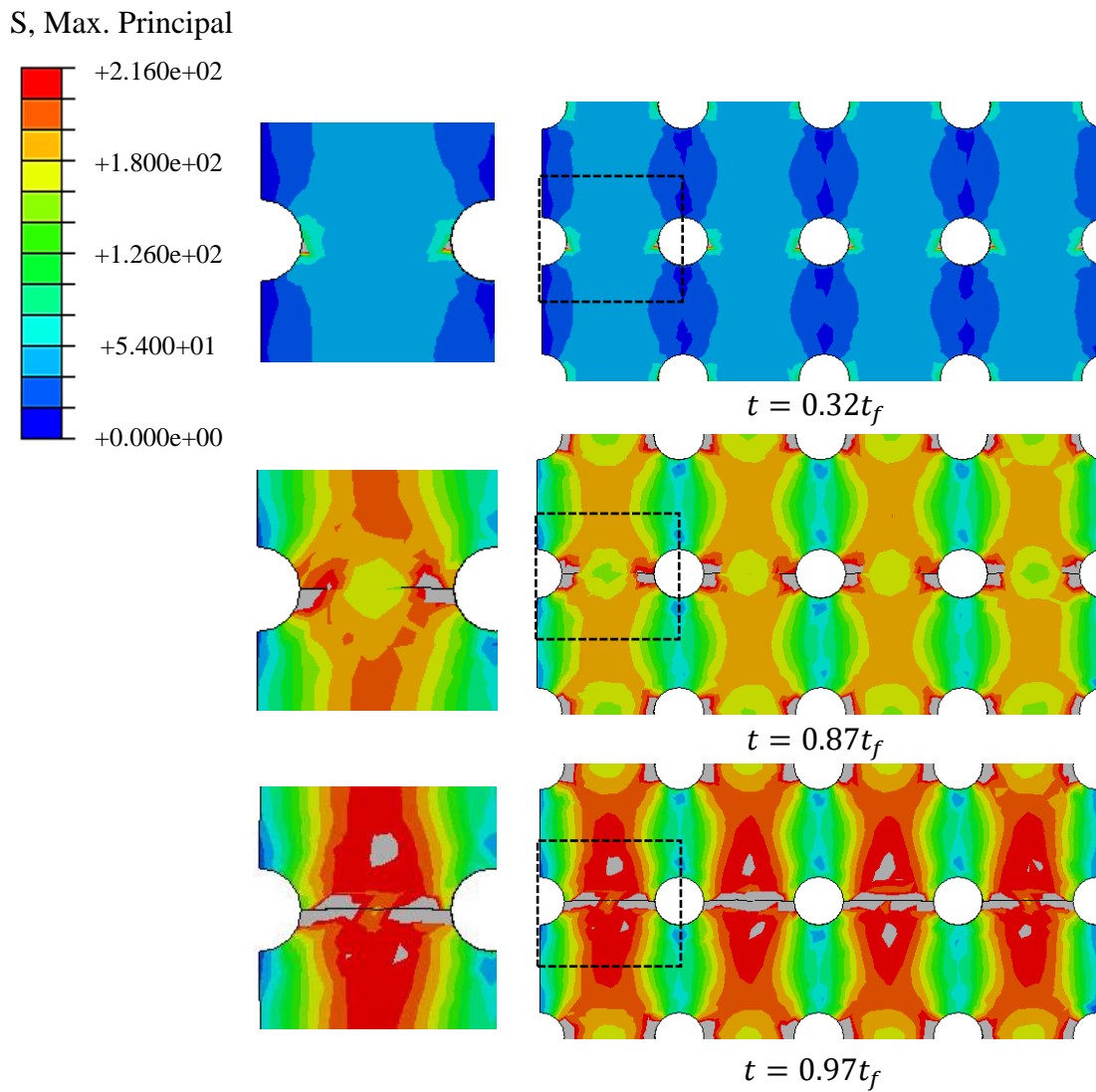


Figure 22. Maximum principal stress contour map from Abaqus with crack growth fracture simulation at specified percentage of time step of enriched region of sample with circular void pattern. Gray regions indicate stress above maximum allowable principal stress (216 MPa).

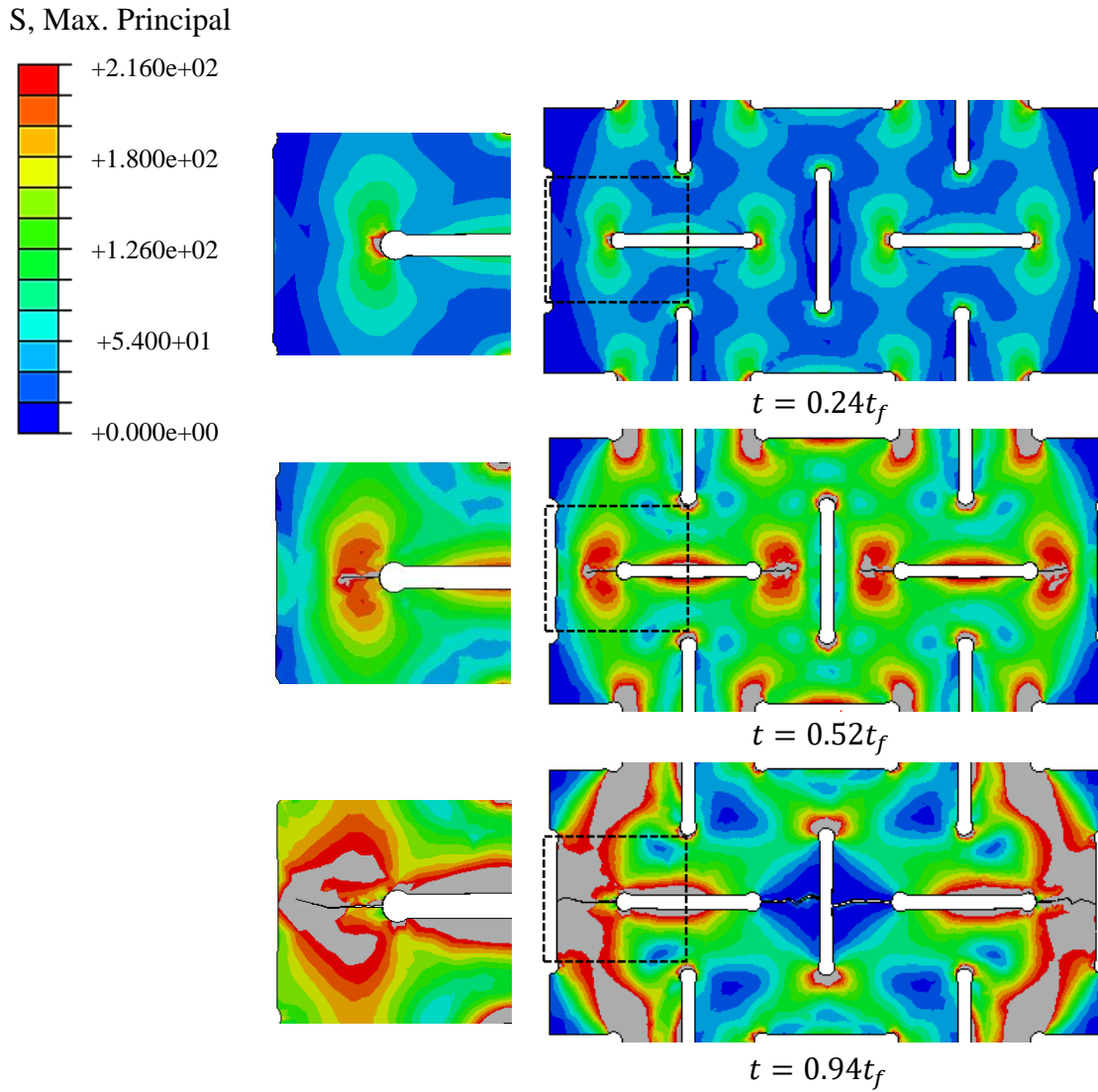


Figure 23. Maximum principal stress contour map from Abaqus with crack growth fracture simulation at specified percentage of time step of enriched region of sample with stop-hole void pattern. Gray regions indicate stress above maximum allowable principal stress (216 MPa).

The XFEM results for the two models demonstrated that the crack evolved horizontally based on the damage criteria of maximum allowable principal stress. Hence,

we have employed these results to have a logical assumption in the contour J-integral models.

With this assumption, we proceed by creating a finite element model of the actual test specimen exactly the same as Figure 21 and introduced artificial cracks on the models. They are similar to what was implemented in the RVEs in Chapter VI but applied for the whole test specimen models. Unlike the RVEs, the whole test specimens have 4 crack locations. Considering that there are no imperfections involved, the contour J-integral on the left crack must be equal to the right crack. Thus, we obtain two results of the contour J-integral for the whole test specimen: one is from the outer cracks (relative to the center) and the other is at the inside cracks, and we denote these regions as Region I and Region II, respectively (Figure 24). Then we created 130 distinct models (65 models per region) of the

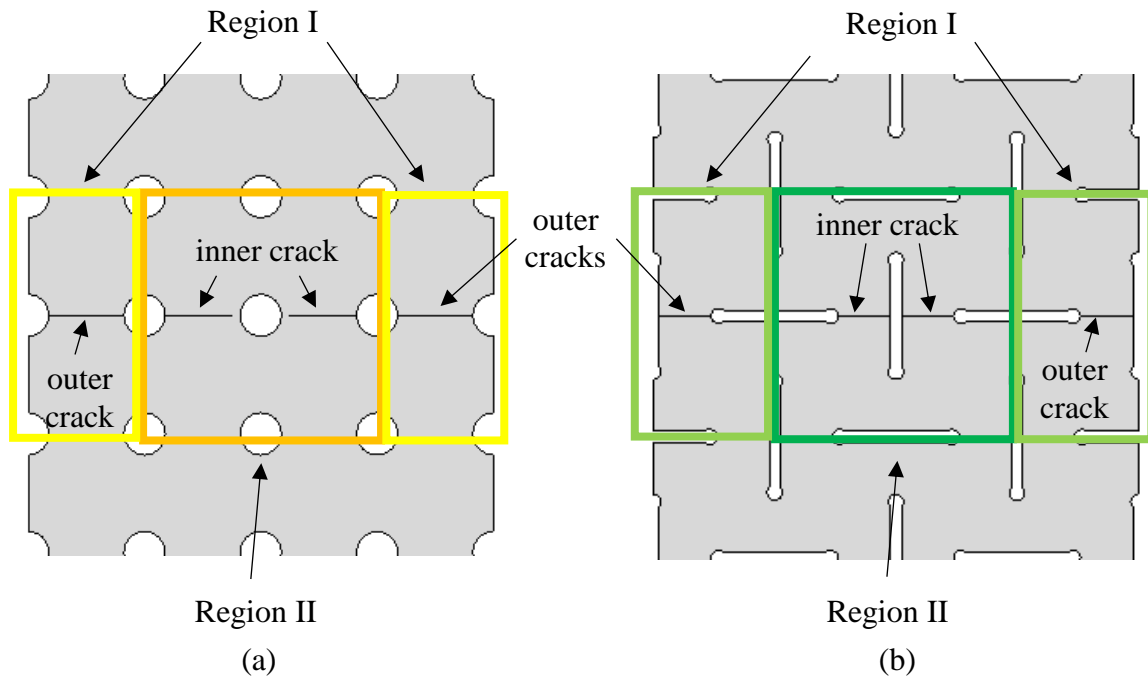


Figure 24. Porous areas of the whole test specimens specifying outer and inner crack regions.
 (a) Circular void model, (b) Stop-hole void model.

test specimen, with increasing horizontal cracks based on L_{\min} (6.1) but with crack range of $0.01L_{\min}$ to $0.99L_{\min}$.

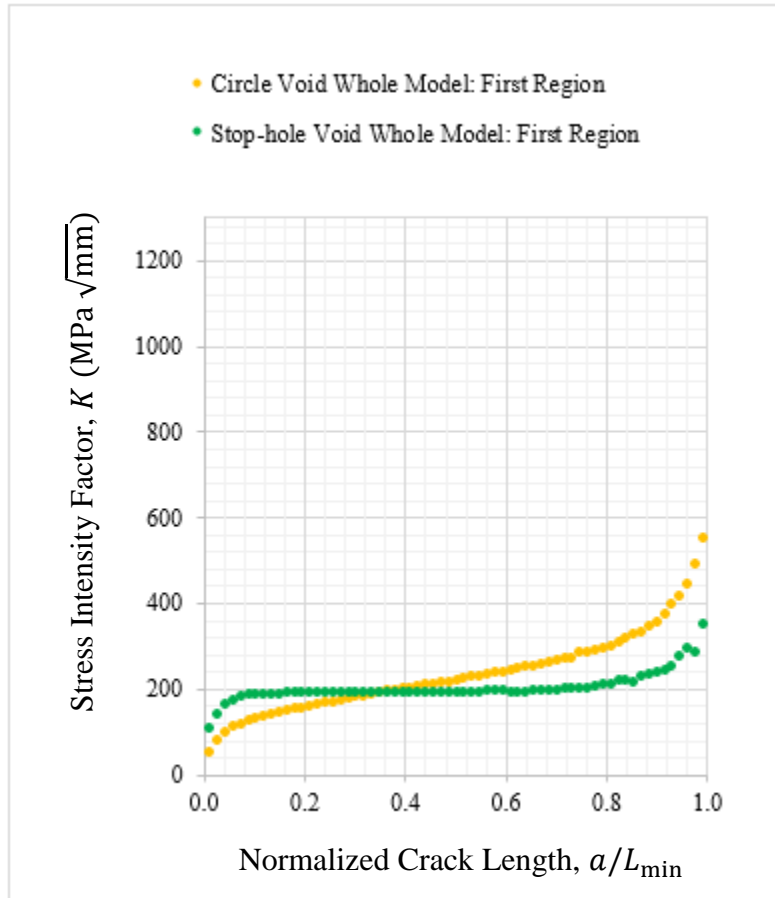
In the simulation, we used the material properties of the actual test specimen. For each test specimen, we modeled 8-node biquadratic plane stress quadrilaterals (Abaqus code: CPS8) with seed mesh size of 0.1 mm. We also applied 6050 N and 3505 N of force for circular void model and stop-hole void model, respectively. We simulated each model by creating a Python script that allows to create an increasing crack length at region I and II (Appendix A.2).

We compared the finite test sample models to the RVEs (infinite periodic models). We created two-dimensional RVEs from the dimensions and material properties of each test specimen. Also, applied on each are 8-node biquadratic plane stress quadrilaterals (Abaqus code: CPS8) with seed mesh size of 0.1 mm.

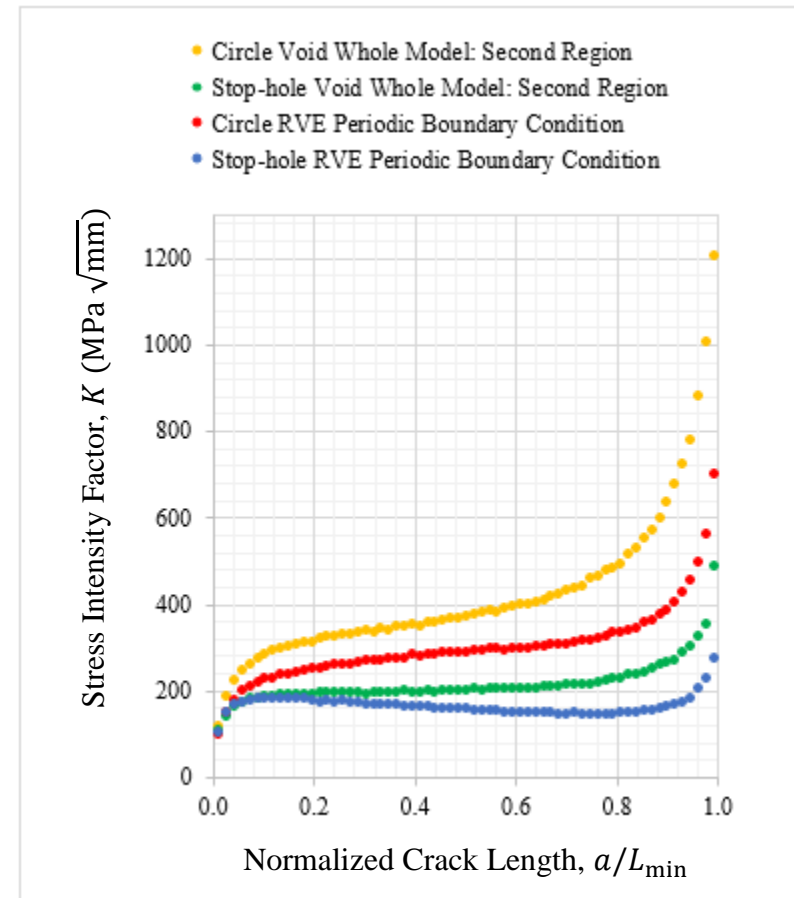
For the contour J-integral result, we created 65 models with increasing constant value based on the range of 1% to 99% of L_{\min} . Like the previous simulation in Chapter VI, we utilized periodic boundary conditions to each model. Since different loads were applied to the specimen, we have computed first for the equivalent applied stress based on the maximum applied load given from the experimental result. Then, we used the material stress-strain curve to interpolate the corresponding strain (Table 1).

	Circular Void Model	Stop-hole Void Model
Applied Force (N)	6050	3505
Applied Stress (MPa)	75.6	43.8
Applied Strain (mm/mm)	0.001049	0.000773

Table 1. Computed load applied to the J-integral models of circular void and stop-hole void.



Region I



Region II

Figure 25. Evolution of stress intensity factor along the normalized crack length for Region I and Region II of the whole test specimen comparing circular void model (non-auxetic) to stop-hole void model (auxetic). For Region II, periodic models of circle void and stop-hole voids are compared.

For Region I, the results show that the circular void model for the test specimen starts at the lower stress intensity factor compared to the stop-hole void model. After $\frac{a}{L_{\min}} = 0.35$, the stress intensity factor of the stop-hole void model is lower than the circular void model. Based on Paris Law, this means that in outer crack region, crack initiation for stop-hole model is faster than the circle model. However, at a certain time in propagation phase, the rate of crack extension of circular void model becomes faster than the stop-hole void model.

For Region II, the K vs $\frac{a}{L_{\min}}$ shows that both whole test specimen started at approximately same level of stress intensity factor which means that the cracks initiate at the same time for both of the test specimens. However, the rate of circular void model is much faster as the crack propagates, while the stop-hole void model has relatively slower rate. It is also observed that in crack propagation, there is only slight difference of stress intensity factor between Region I and Region II for stop-hole void model. For the circular void model, the stress intensity factors are much higher in Region II compared to Region I.

We also graphed the K vs $\frac{a}{L_{\min}}$ of the RVE models and compare it to the whole test specimen models in Region II. The results showed that the test specimen models have higher set of stress intensity factors compared to the infinite models. However, it is consistent for both finite and infinite models that the stress intensity of stop-hole void model is lower compared to the circular void model. Applying Paris law, in which K is inversely proportional to the number of fatigue cycles, the numerical data showed that the crack propagation was faster for the circular void model compared to stop-hole void model. In other words, these data support the experimental data where the crack initiated first for circular void model in comparison to the other and the number of fatigue cycles for stop-hole void model is much higher than that of the model with circular void.

CHAPTER VIII

Conclusion

The purposes of this study were: first, to provide a numerical analysis of the fatigue fracture behavior of auxetic structures with various geometric parameters through the use of the contour J-integral, and second, to apply a similar procedure of numerical methods to the existing experimental data. XFEM was also applied in both analyses in order to simulate the actual crack propagation of the models and to support the assumption underlying the J-integral calculation.

For the variation of parameter, the dimensions of models with symmetric orthogonal patterns, such as ellipse, slot, and stop-hole, were changed by altering the axis-ratio of the void shapes. The analysis was divided into two, where the one parameter that was maintained to be constant, ψ and L_{\min} . For analysis where 5% porosity was fixed the following result showed that:

- (1) when axis-ratio were changed with increasing amount, hence L_{\min} decreased for every increment of the axis ratio in which also the RVE approached a more negative Poisson's ratio, the values of stress intensity factor along the crack evolution were decreasing,
- (2) both slot void and stop-hole void models have lower starting point of stress intensity factor compared to the elliptical void model in which we conclude based on Paris' Law, that elliptical void pattern of auxetic structure had faster crack initiation compared to the other two models.

For the analysis where L_{\min} defined as constant for each void shape pattern the following result showed that:

- (3) when axis ratio was shifted into multiples of 5 on models above (J^{+1} , J^{+2} and J^{+3}) and below (J^{-3} , J^{-2} and J^{-1}) the reference model of elliptical void pattern, J^0 , higher stress intensity factors were computed for the models J^{-3} , J^{-2} and J^{-1}

compared to the models J^{+1} , J^{+2} and J^{+3} . We also conclude, based on Paris' Law, that for the RVE with ellipse pattern, higher porosity (which also have the lowest value of negative Poisson's ratio), tend to have shorter fatigue life,

- (4) when axis ratio was shifted into multiples of 5 on models above (H^{+1} , H^{+2} and H^{+3}) and below (H^{-3} , H^{-2} and H^{-1}) the reference model of slot void pattern, H^0 , lower stress intensity factors at crack initiation were found for the models H^{-3} , H^{-2} and H^{-1} compared to the models H^{+1} , H^{+2} and H^{+3} . We conclude that for a constant L_{\min} , models H^{+1} , H^{+2} and H^{+3} have faster crack initiation compared to the models H^{-3} , H^{-2} and H^{-1} ,
- (5) for the stop-hole void model, when axis ratio was fixed but stop-hole radius was changed in order to simulate specific AR_r , lower stress intensity factors at crack initiation were found for the models below (J^{-3} , J^{-2} and J^{-1}) the reference model, J^0 . For stop-hole void, models J^{-3} , J^{-2} and J^{-1} (at constant L_{\min}), having lower stress intensity factor, tend to have slower crack initiation in comparison to the models (J^{+1} , J^{+2} and J^{+3}) above the reference model, J^0 .

We also have provided numerical comparison to the actual fatigue fracture experiments. Here, plates with circular void (non-auxetic) pattern were compared to plates with stop-hole void pattern (auxetic) (both have constant porosity of 10%) in terms of their behavior along their life cycle. The contour J-integral computation results showed a good agreement to the experimental data where stop-hole void models showed lower values of K on both crack initiation and crack propagation compared to the model with circular void pattern. Based on Paris' Law, we conclude that material with auxetic structure have the higher over-all fatigue life than the non-auxetic.

Future research should consider a mix mode of loading for the test samples. It will be important to investigate the crack behavior of auxetic materials when biaxial or triaxial loads are applied. It will also help if larger specimens will be examined. This will be beneficial in comparing the large specimen with auxetic patterns to infinite models (periodic).

REFERENCES

- [1] Lakes, R.S., *Design considerations for negative Poisson's ratio materials*, ASME J Mech. Design, **115**, 1993, pp. 696-700.
- [2] Lakes, R.S., *Foam structures with a negative Poisson's ratio*, Science, **235**, 1987, pp. 1038-1040.
- [3] Tan, X., Aulbach, W., Granzow, T., Kling., M., Marsilius, Kleebe, H.J., Rodel, J., *Effect of uniaxial stress on ferroelectric behavior of [Bi(1/2)Na(1/2)] TiO(3)-based lead-free piezoelectric ceramics*. Journal of Applied Physics, **106**, 2009, 044107.
- [4] Caddock, B.D., Evans, K.E., *Microporous materials with negative Poisson's ratios: I. Microstructure and mechanical properties*. Journal of Physics D: Applied Physics **22** (12), 1989, pp. 1877–1882.
- [5] Ho, D.T., Park, S.D., Kwon, S.Y., Park, K., Kim, S.Y., *Negative Poisson's ratios in metal nanoplates*. Nature Communications. **5**, 2014, p. 3255.
- [6] Alderson, K.L., Simkins, V.R., Coenen, V.L., Davies, P.J., Alderson, A., Evans, K.E., *How to make auxetic fibre reinforced composites*. Physica Status Solidi B 242 (3), 2005, pp. 509–518.
- [7] Lim, T. C., *Out-of-plane modulus of semi-auxetic laminates*. European Journal of Mechanics A/Solids **28**, 2009, pp. 752–756.
- [8] Gaspar, N., Smith, C.W., Alderson, A., Grima, J.N., Evans, K.E., *A Generalised Three-Dimensional Tethered-Nodule Model for Auxetic Materials*, J. Mater. Sci., **46**, 2011, pp. 372-384.
- [9] Attard, D., Grima, J.N., *Modelling of hexagonal honeycombs exhibiting zero Poisson's ratio*. Physica Status Solidi., **248**, 2011, pp. 52-59.
- [10] Liu, Y., Hu, H., *A review on auxetic structures and polymeric materials*, Scientific Research and Essays, **5**, 2010, pp. 1052-1063.
- [11] Lakes, R.S., *Deformation mechanisms in negative Poisson's ratio materials: structural aspects*. J. Mater. Sci., **26**, 1991, pp. 2287-2292.
- [12] Dolla, W.J.S., Fricke, B.A., Becker, B.R., *Structural and Drug Diffusion Models of Conventional and Auxetic Drug-eluting Stents*. J. Medical Devices, **1**, 2007, pp. 47-55.

- [13] Smith, C.W., Grima, J.N., Evans, K.E., *A novel mechanism for generating auxetic behaviour in reticulated foams: missing rib foam model*. Acta. Mater., **48**, 2000, pp. 4349-4356.
- [14] Taylor, M., Francesconi, L., Gerendas, M., Shanian, A., Carson, C., Bertoldi, K., *Low Porosity Metallic Periodic Structures with Negative Poisson's Ratio*, **26**, 2014, pp. 1-6.
- [15] Francesconi, L., Taylor, M., Bertoldi, K., Baldi, A., *A numerical-experimental investigation of thin auxetic metallic structures*. **3**, 2016, pp. 335-341.
- [16] Javid F., Liu J., Rafsanjani, A., Schaenzer, Pham M.Q., Backman, D., Yandt, S., Innes, M.C., Booth-Morrison, C., Gerendas, M., Scarinci, T., Shanian, A., Bertoldi, K., *On the design of porous structures with enhanced fatigue life*. Extreme Mechanics Letters. **16**, 2017, pp.13-17.
- [17] Francesconi, L., Taylor, M., Baldi, A., *An investigation of stress concentration, crack nucleation, and fatigue life of thin low porosity metallic auxetic structures*, Proceedings of the 2018 SEM Annual Conference and Exposition on Experimental and Applied Mechanics, (submitted).
- [18] Paris, P.C. and Erdogan, F., *A Critical Analysis of Crack Propagation Laws*. Journal of Basic Engineering, **85**, 1960, pp. 528–534.
- [19] Griffith, A.A., *The Phenomena of Rupture and Flow in Solids*. Philosophical Transactions, Series A, **221**, 1920, pp. 163–198.
- [20] Irwin, G.R., *On Set of Crack Propagation in High Strength Steel and Aluminum Alloys*. Sagamore Research Conference Proceedings, **2**, 1956, pp 289-350.
- [21] Anderson, T.L., *Fracture Mechanics: Fundamentals and Applications*. 3rd ed. CRC Press LLC. 2005, pp. 43, 52, 59.
- [22] Westergaard, H.M., *Bearing Pressures and Cracks*. Journal of Applied Mechanics, **6**, 1939, pp. 49–53.
- [23] Irwin, G.R., *Analysis of Stresses and Strains near the End of a Crack Traversing a Plate*. Journal of Applied Mechanics, **24**, 1957, pp. 361–364.
- [24] Sih, G.C., *On the Westergaard Method of Crack Analysis*. International Journal of Fracture Mechanics, **2**, 1966, pp. 628–631.
- [25] Sanford, R.J., *A Critical Re-Examination of the Westergaard Method for Solving Opening Mode Crack Problems*. Mechanics Research Communications, **6**, 1979, pp. 289–294.

- [26] Helen, T.K., *On the method of virtual crack extension*. Int. J. Numer. Meth. Engineering. **9**, 1975, pp. 187-207.
- [27] Shih, C.F., Moran, B., Nakamura, T., *Energy release rate along a three-dimensional crack front in a thermally stressed body*. International Journal of Fracture. **30**, 1986 pp. 79-102.
- [28] Li, F.Z., Shih, C.F., Needleman, A., *A comparison of methods for calculating energy release rates*. Engineering Fracture Mechanics. **21**, No. 2, 1985 pp. 405-421.
- [29] Rice, J. R. *A path independent integral and the approximate analysis of strain concentration by notches and cracks*. Journal of Applied Mechanics **35**, 1968 pp. 368–375.
- [30] Mohammadi, S., *Extended Finite Element Method for Fracture Analysis of Structures*. Blackwell Publishing Ltd. 2008, pp. 59, 69-96.
- [31] Cox, J.V., *An extended finite element method with analytical enrichment for cohesive crack modeling*, International Journal for Numerical Methods in Engineering, **78**, 2009, pp. 48-83.
- [32] Belytschko, T., Black T., *Elastic crack growth in finite elements with minimal remeshing*. Int. J. Numer. Meth. Engng. **45**, 1999, pp. 601-620.
- [33] Fries, T.P., Baydoun, M., *Crack propagation with the extended finite element method and a hybrid explicit–implicit crack description*, Int. J. Numer. Meth. Engng. **89**, 2012; pp. 1527–1558.
- [34] Stazi, F.L., Budyn, E., Chessa, J., Belytschko, T., *An extended finite element method with higher-order elements for curved cracks*. Computational Mechanics. **31**, 2003, pp. 38-48.
- [35] Melenk J.M., Babuska, I., *The partition of unity finite element method: basic theory and applications*. Comput. Methods Appl. Mech. Eng. **39**, 1996, pp. 289–314
- [36] Belytschko, T., and Fish, J., *A first course in finite elements*. John Wiley and Sons Ltd., England, 2007, pp. 77-79, 223.
- [37] Belytschko T., Gracie, R., Ventura G., *A review of extended/generalized finite element methods for material modeling*, Modelling Simul. Mater. Sci. Eng., **17**, 2009, 043001, pp. 1-24.
- [38] Bordas, S., Nguyen, P.V., Dunant, C., Guidoum, Hung Nguyen-Dang, H., *An extended finite element library*. Int. J. Numer. Meth. Engng, **71**, 2007, pp. 703–732.
- [39] Belytschko T, Lu YY, Gu L. *Element-free Galerkin methods*. International Journal of Numerical Methods in Engineering. **37**, 1994, pp. 229-256.

- [40] Fleming, M., Chu Y.A., Moran, B., Belytschko, T., *Enriched element-free galerkin methods for crack tip fields*. Int. J. Numer. Meth. Eng. **40**, 1997, pp. 1483–1504.
- [41] Abaqus 6.14 Simulia, Abaqus Analysis User's Guide, 2016.
source:<<http://130.149.89.49:2080/v2016/books/usb/default.htm?startat=pt04ch10s07at36.html>>
- [42] Erdogan, F., and Sih, G.C., *On the Crack Extension in Plates under Plane Loading and Transverse Shear*, Journal of Basic Engineering, **85**, 1963, pp. 519–527.
- [43] Oller, S., *Fractura Mecánica. Un Enfoque Global*, Centro Internacional de Métodos Numéricos EnIngeniería, 2001 Numerical Simulation of Mechanical, p. 2001.

APPENDIX

A.1 Python Script for generating 65 models with increasing crack length using periodic boundary conditions: (Stop-hole void model). Same codes were applied to the circular void, ellipse void, and slot void models but with different part geometries.

Note: For more information in the Python Script of other void geometries, contact Dr. Michael Taylor (mjtaylor@scu.edu).

```
# Garivalde Dominguez
# Reference: Michael Taylor
# 03222018

pathName
="C:/Users/gdomingu/Python_Abaqus_Script/RVE_StopholeVoid__PBC_constPorosity_0p1/"
os.chdir(pathName)

# LIBRARY
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 math
import os

session.journalOptions.setValue(replayGeometry=COORDINATE, recoverGeometry=COORDINATE)
# MODEL -----
# Model Name
modelName = 'Unit_Stophole_Void'
mdb.models.changeKey(fromName='Model-1', toName=modelName)

# Material Properties -----
materialName = 'stainless_steel'
Youngs_Modulus = 65.4e3      # Young's modulus (in MPa)
Poissons_Ratio = 0.32      # Poisson's Ratio

# Geometric Properties -----
center_to_center = 10      # center to center distance for the holes (in mm)
porosity = 0.10            # porosity
thickness = 0.0            # thickness of the plates
seed_mesh = 0.10          # seed-mesh (in mm)

width_plate = 2.0*center_to_center; # width of plate
height_plate = 2.0*center_to_center; # height of plate
```

```

stop_hole_radius = 0.6250          # stop hole radius (in mm)
minor_axis_hole = 0.4500          # major axis of each void
major_axis_hole = 4.6255          # major axis of each void

# ratio between major and minor axis for the holes
axes_ratio = 11.6677;

# Crack Geometries -----
# minimum hole distance (in mm)
minimum_hole_distance = center_to_center - minor_axis_hole - major_axis_hole
    - stop_hole_radius;
# number of increments
num_increments = 50;
# crack length increment
crack_length_increment = (0.99*minimum_hole_distance - 0.01*minimum_hole_distance)
    / (num_increments-1);

print 'minimum_hole_distance: ' + str(minimum_hole_distance)

# Displacement Load -----
strain_load = 0.000773          # strain load (in mm/mm)
displacement_load = strain_load*center_to_center;    # displacement (in mm)

for crack_counter in range(0, num_increments):
    # crack length (in mm)
    crack_length = 0.01*minimum_hole_distance + crack_counter*crack_length_increment;

    print 'counter: ' + str(crack_counter +1 )
    print 'crack length: ' + str(crack_length)
    print 'number of increments: ' + str(num_increments)
    print 'crack length increment: ' + str(crack_length_increment)

    # round crack length for naming
    crack_length_5deci = math.ceil(crack_length*1000000)/1000000;

    subPath = pathName + 'P' + str(porosity).replace('.', 'p') + '_AR'
        + str(axes_ratio).replace('.', 'p') + '_MS' =
        str(seed_mesh).replace('.', 'p')
        + '_CL' + str(crack_length_5deci).replace('.', 'p') + "/"

    if not os.path.exists(subPath):
        os.makedirs(subPath)
    os.chdir(subPath)

# PARTS -----
# PART: Virtual Point at x coordinate
mdb.models[modelName].Part(dimensionality=TWO_D_PLANAR, name='part_VPx',
type=DEFORMABLE_BODY)
mdb.models[modelName].parts['part_VPx'].ReferencePoint(point=(0.0, 0.0, 0.0))
# PART: Virtual Point at y coordinate
mdb.models[modelName].Part(dimensionality=TWO_D_PLANAR, name='part_VPy',
type=DEFORMABLE_BODY)
mdb.models[modelName].parts['part_VPy'].ReferencePoint(point=(0.0, 0.0, 0.0))
# Part Name
partName = 'RVE_Plate'
# PART: Base plate
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].rectangle(
    point1=(-1.0*center_to_center, -1.0*center_to_center),
    point2=(center_to_center, center_to_center))
mdb.models[modelName].Part(dimensionality=TWO_D_PLANAR, name=partName,
type=DEFORMABLE_BODY)
mdb.models[modelName].parts[partName].BaseShell(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
# PART: Center-Center Void
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)

```

```

mdb.models[modelName].sketches['__profile__'].rectangle(
    point1=(-1.0*minor_axis_hole, -1.0*major_axis_hole),
    point2=(minor_axis_hole, major_axis_hole))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(0, major_axis_hole),
    point1=(0, major_axis_hole + stop_hole_radius))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(0, -1.0*major_axis_hole),
    point1=(0, -1.0*major_axis_hole - stop_hole_radius))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
# PART: Center-Top Void
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].rectangle(
    point1=(-1.0*major_axis_hole, center_to_center - minor_axis_hole),
    point2=(major_axis_hole, center_to_center + minor_axis_hole))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(-1.0*major_axis_hole, center_to_center),
    point1=(-1.0*major_axis_hole - stop_hole_radius, center_to_center))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(major_axis_hole, center_to_center),
    point1=(major_axis_hole + stop_hole_radius, center_to_center))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
# PART: Center-Bottom Void
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].rectangle(
    point1=(-1.0*major_axis_hole, -1.0*center_to_center - minor_axis_hole),
    point2=(major_axis_hole, -1.0*center_to_center + minor_axis_hole))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(-1.0*major_axis_hole, -1.0*center_to_center),
    point1=(-1.0*major_axis_hole - stop_hole_radius, -1.0*center_to_center))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(major_axis_hole, -1.0*center_to_center),
    point1=(major_axis_hole + stop_hole_radius, -1.0*center_to_center))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
# PART: Right-Center Void
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].rectangle(

```

```

        point1=(-1.0*major_axis_hole + center_to_center, -1.0*minor_axis_hole),
        point2=(major_axis_hole + center_to_center, minor_axis_hole)
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(-1.0*major_axis_hole + center_to_center, 0),
    point1=(-1.0*major_axis_hole - stop_hole_radius + center_to_center, 0))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(major_axis_hole + center_to_center, 0),
    point1=(major_axis_hole + stop_hole_radius + center_to_center, 0))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
# PART: Left-Center Void
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].rectangle(
    point1=(-1.0*major_axis_hole - center_to_center, -1.0*minor_axis_hole),
    point2=(major_axis_hole - center_to_center, minor_axis_hole))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(-1.0*major_axis_hole - center_to_center, 0),
    point1=(-1.0*major_axis_hole - stop_hole_radius - center_to_center, 0))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(major_axis_hole - center_to_center, 0),
    point1=(major_axis_hole - stop_hole_radius - center_to_center, 0))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
# PART: Top-Left-Corner Void
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].rectangle(
    point1=(-1.0*minor_axis_hole - center_to_center, -1.0*major_axis_hole
+ center_to_center),
    point2=(minor_axis_hole - center_to_center, major_axis_hole
+ center_to_center))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(-1.0*center_to_center, major_axis_hole + center_to_center),
    point1=(-1.0*center_to_center, major_axis_hole + stop_hole_radius
+ center_to_center))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(-1.0*center_to_center, -1.0*major_axis_hole + center_to_center),
    point1=(-1.0*center_to_center, -1.0*major_axis_hole - stop_hole_radius
+ center_to_center))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']

```

```

# PART: Top-Right-Corner Void
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].rectangle(
    point1=(-1.0*minor_axis_hole + center_to_center, -1.0*major_axis_hole
    + center_to_center),
    point2=(minor_axis_hole + center_to_center, major_axis_hole
    + center_to_center))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(center_to_center, major_axis_hole + center_to_center),
    point1=(center_to_center, major_axis_hole + stop_hole_radius
    + center_to_center))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(center_to_center, -1.0*major_axis_hole + center_to_center),
    point1=(center_to_center, -1.0*major_axis_hole - stop_hole_radius
    + center_to_center))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
# PART: Bottom-Left-Corner Void
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].rectangle(
    point1=(-1.0*minor_axis_hole - center_to_center, -1.0*major_axis_hole
    - center_to_center),
    point2=(minor_axis_hole - center_to_center, major_axis_hole
    - center_to_center))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(-1.0*center_to_center, major_axis_hole - center_to_center),
    point1=(-1.0*center_to_center, major_axis_hole - stop_hole_radius
    - center_to_center))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(-1.0*center_to_center, -1.0*major_axis_hole - center_to_center),
    point1=(-1.0*center_to_center, -1.0*major_axis_hole - stop_hole_radius
    - center_to_center))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
# PART: Bottom-Right-Corner Void
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].rectangle(
    point1=(-1.0*minor_axis_hole + center_to_center, -1.0*major_axis_hole
    - center_to_center),
    point2=(minor_axis_hole + center_to_center, major_axis_hole
    - center_to_center))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(center_to_center, major_axis_hole - center_to_center),
    point1=(center_to_center, major_axis_hole - stop_hole_radius
    - center_to_center))

```

```

mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].CircleByCenterPerimeter(
    center=(center_to_center, -1.0*major_axis_hole - center_to_center),
    point1=(center_to_center, -1.0*major_axis_hole - stop_hole_radius
    - center_to_center))
mdb.models[modelName].parts[partName].Cut(sketch=
mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
# PART: Left-Crack and Right Crack
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].Line(
    point1=(-1.0*center_to_center + major_axis_hole + stop_hole_radius, 0.0),
    point2=(-1.0*center_to_center + major_axis_hole + stop_hole_radius
    + crack_length,0.0))
mdb.models[modelName].sketches['__profile__'].Line(
    point1=(center_to_center - major_axis_hole - stop_hole_radius, 0.0),
    point2=(center_to_center - major_axis_hole - stop_hole_radius
    - crack_length,0.0))
mdb.models[modelName].parts[partName].PartitionFaceBySketch(
    faces=mdb.models[modelName].parts[partName].faces, sketch=
    mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']

# MATERIAL -----
mdb.models[modelName].Material(description='Linear elastic material model', name=
materialName).Elastic(table=((Youngs_Modulus, Poissons_Ratio),))

# SECTION -----
mdb.models[modelName].HomogeneousSolidSection(material=materialName,
name='unit_cell', thickness=None)
mdb.models[modelName].parts[partName].SectionAssignment(region=
Region(faces=mdb.models[modelName].parts[partName].faces),
sectionName='unit_cell')
# ASSEMBLY -----
instName = 'voided_plate'
mdb.models[modelName].rootAssembly.DatumCsysByDefault(CARTESIAN)
mdb.models[modelName].rootAssembly.Instance(dependent=OFF, name=instName,
part=mdb.models[modelName].parts[partName])
# Virtual point to constrain x motion
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name='inst_VPx',
part=mdb.models[modelName].parts['part_VPx'])
# Virtual point to constrain y motion
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name='inst_VPy',
part=mdb.models[modelName].parts['part_VPy'])

# INTERACTION -----
# center of left crack at x-direction (in mm)
left_crack_x = -1.0*center_to_center + major_axis_hole + stop_hole_radius
+ crack_length/2.0;
# center of right crack at x-direction (in mm)
right_crack_x = center_to_center - major_axis_hole - stop_hole_radius
- crack_length/2.0;
# INTERACTION: Assign Crack Set
mdb.models[modelName].rootAssembly.Set(edges=
mdb.models[modelName].rootAssembly.instances[instName].edges.findAt(((
right_crack_x, 0.0, 0.0), )), name='right_crack')
mdb.models[modelName].rootAssembly.Set(edges=
mdb.models[modelName].rootAssembly.instances[instName].edges.findAt(((
left_crack_x, 0.0, 0.0), )), name='left_crack')
# INTERACTION: Assign Seam
mdb.models[modelName].rootAssembly.engineeringFeatures.assignSeam(regions=
mdb.models[modelName].rootAssembly.sets['right_crack'])
mdb.models[modelName].rootAssembly.engineeringFeatures.assignSeam(regions=
mdb.models[modelName].rootAssembly.sets['left_crack'])

```

```

# INTERACTION: Contour Integral
# tip of left crack at x-direction (in mm)
left_crack_tip = -1.0*center_to_center + major_axis_hole + stop_hole_radius
                + crack_length;
# tip of right crack at x-direction (in mm)
right_crack_tip = center_to_center - major_axis_hole - stop_hole_radius
                - crack_length;
mdb.models[modelName].rootAssembly.engineeringFeatures.ContourIntegral(
    collapsedElementAtTip=NONE,crackFront=Region(edges=

mdb.models[modelName].rootAssembly.instances[instName].edges.findAt(((
    right_crack_x, 0.0, 0.0), ), )), crackTip=Region(vertices=

mdb.models[modelName].rootAssembly.instances[instName].vertices.findAt((
    (right_crack_tip, 0.0, 0.0), ), ),
    extensionDirectionMethod=Q_VECTORS,
    midNodePosition=0.5, name='right_crack', qVectors=((

mdb.models[modelName].rootAssembly.instances[instName].vertices.findAt((
    right_crack_tip, 0.0, 0.0), ), (-1.0, 0.0, 0.0)), ),
    symmetric=ON)
mdb.models[modelName].rootAssembly.engineeringFeatures.ContourIntegral(
    collapsedElementAtTip=NONE,crackFront=Region(edges=
    mdb.models[modelName].rootAssembly.instances[instName].edges.findAt((
    (left_crack_x, 0.0, 0.0), ), ), ), crackTip=Region(vertices=

mdb.models[modelName].rootAssembly.instances[instName].vertices.findAt(((
    left_crack_tip, 0.0, 0.0), ),)),
    extensionDirectionMethod=Q_VECTORS,
    midNodePosition=0.5, name='left_crack', qVectors=((

mdb.models[modelName].rootAssembly.instances[instName].vertices.findAt((
    left_crack_tip, 0.0, 0.0), ), (1.0, 0.0, 0.0)), ),
    symmetric=ON)

# STEP -----
mdb.models[modelName].StaticStep(description='Uniaxial Tension in y-y direction',
    name='Tension', previous='Initial')
mdb.models[modelName].steps['Tension'].setValues(adaptiveDampingRatio=None,
    continueDampingFactors=False, matrixSolver=DIRECT,
    solutionTechnique=FULL_NEWTON,
    stabilizationMethod=NONE)

# MESH(based on Assembly) -----
# MESH: Seed Mesh
mdb.models[modelName].rootAssembly.seedPartInstance(deviationFactor=0.1,
    minSizeFactor=0.1, regions=(
    mdb.models[modelName].rootAssembly.instances[instName], ), size=seed_mesh)
# MESH: Element Type
mdb.models[modelName].rootAssembly.setElementType(elemTypes=(
    ElemType(elemCode=CPS8, elemLibrary=STANDARD), ElemType(elemCode=CPS8,
    elemLibrary=STANDARD)), regions=(mdb.models[modelName].rootAssembly.
    instances[instName].faces.findAt((center_to_center/2.0,
    center_to_center/2.0,
    thickness/2.0),),))
# MESH: Control
mdb.models[modelName].rootAssembly.setMeshControls(elemShape=QUAD,

regions=(mdb.models[modelName].rootAssembly.instances[instName].faces.findAt(
    (center_to_center/2.0, center_to_center/2.0, thickness/2.0),),))
# MESH: Generate Mesh
mdb.models[modelName].rootAssembly.generateMesh(regions=
    (mdb.models[modelName].rootAssembly.instances[instName], ))

# -----
# Set: All nodes
mdb.models[modelName].rootAssembly.Set(name='set_AllElements', elements=

```



```

        mdb.models[modelName].rootAssembly.instances[instName].elements)
mdb.models[modelName].rootAssembly.Set(name='set_AllNodes', nodes=
mdb.models[modelName].rootAssembly.instances[instName].nodes)
# Create arrays and Sets containing node numbers for all faces of plate -----

# initialize arrays for edges
nodes_rightEdge = []
nodes_leftEdge = []
nodes_topEdge = []
nodes_bottomEdge = []
node_RBM = []

# define arbitrary tolerance for boolean comparison
eps = seed_mesh/100.0

# loop over all nodes and sort out nodes on the edges
for N in mdb.models[modelName].rootAssembly.instances[instName].nodes:

    nodeCoord = N.coordinates

    #print 'nodeCoord: ' + str(nodeCoord [0]) + ',' + str(nodeCoord [1])

    if (fabs(nodeCoord[0]-major_axis_hole - stop_hole_radius) < 100.0*eps)
        and (fabs(nodeCoord[1]-major_axis_hole - stop_hole_radius) <
            100.0*eps):
        node_RBM.append(N.label)

    elif (fabs(nodeCoord[0] + center_to_center) < eps):
        nodes_leftEdge.append(N.label)

    elif (fabs(nodeCoord[0] - center_to_center) < eps):
        nodes_rightEdge.append(N.label)

    elif (fabs(nodeCoord[1] + center_to_center) < eps):
        nodes_bottomEdge.append(N.label)

    elif (fabs(nodeCoord[1] - center_to_center) < eps):
        nodes_topEdge.append(N.label)

mdb.models[modelName].rootAssembly.SetFromNodeLabels(name=
'set_NodesRightEdge', nodeLabels=((instName, nodes_rightEdge),))
mdb.models[modelName].rootAssembly.SetFromNodeLabels(name=
'set_NodesLeftEdge', nodeLabels=((instName, nodes_leftEdge),))
mdb.models[modelName].rootAssembly.SetFromNodeLabels(name=
'set_NodesTopEdge', nodeLabels=((instName, nodes_topEdge),))
mdb.models[modelName].rootAssembly.SetFromNodeLabels(name=
'set_NodesBottomEdge', nodeLabels=((instName, nodes_bottomEdge),))
mdb.models[modelName].rootAssembly.SetFromNodeLabels(name=
'set_NodeRBM', nodeLabels=((instName, (node_RBM[0],)),))

# Set: Virtual Points
mdb.models[modelName].rootAssembly.Set(name='set_VPx', referencePoints=

(mdb.models[modelName].rootAssembly.instances['inst_VPx'].referencePoints[1], ))
mdb.models[modelName].rootAssembly.Set(name='set_VPy', referencePoints=

(mdb.models[modelName].rootAssembly.instances['inst_VPy'].referencePoints[1], ))

# Create sets of periodic node pairs -----

# Look at left and right sides
for i in range (0, len(nodes_leftEdge)):
    leftCoords = mdb.models[modelName].rootAssembly.
        sets['set_NodesLeftEdge'].nodes[i].coordinates
    mdb.models[modelName].rootAssembly.SetFromNodeLabels(
        name='set_NodesLPair_'
        + str(i), nodeLabels=((instName , (nodes_leftEdge[i],)),))

```

```

for j in range (0, len(nodes_rightEdge)):
    rightCoords = mdb.models[modelName].rootAssembly.
        sets['set_NodesRightEdge'].nodes[j].coordinates
    if (fabs(leftCoords[1] - rightCoords[1]) < eps):
        mdb.models[modelName].rootAssembly.SetFromNodeLabels(
            name='set_NodesRPair_'
            + str(i), nodeLabels=((instName, (
                nodes_rightEdge[j],)),))

# Look at top and bottom sides
for i in range (0, len(nodes_topEdge)):
    topCoords = mdb.models[modelName].rootAssembly.
        sets['set_NodesTopEdge'].nodes[i].coordinates

mdb.models[modelName].rootAssembly.SetFromNodeLabels(name='set_NodesTPair_'
    + str(i), nodeLabels=((instName, (nodes_topEdge[i],)),))
for j in range (0, len(nodes_bottomEdge)):
    bottomCoords = mdb.models[modelName].rootAssembly.
        sets['set_NodesBottomEdge'].nodes[j].coordinates
    if (fabs(topCoords[0] - bottomCoords[0]) < eps):
        mdb.models[modelName].rootAssembly.SetFromNodeLabels(
            name='set_NodesBPair_'
            + str(i), nodeLabels=((instName, (
                nodes_bottomEdge[j],)),))

# BOUNDARY CONDITIONS -----

# fix point to prevent rigid body motion
mdb.models[modelName].DisplacementBC(amplitude=UNSET, createStepName=
'Tension', distributionType=UNIFORM, fieldName='', fixed=OFF,
localCsys=None, name='bc_preventRBM', region=
mdb.models[modelName].rootAssembly.sets['set_NodeRBM']
, u1=0.0, u2=0.0, ur3=UNSET)

# externally applied strain through the virtual points (x-dir)
#-----
mdb.models[modelName].DisplacementBC(amplitude=UNSET, createStepName=
'Tension', distributionType=UNIFORM, fieldName='', fixed=OFF,
localCsys=None, name='bc_VPx', region=
mdb.models[modelName].rootAssembly.sets['set_VPx']
, u1=UNSET, u2=0.0, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)

# externally applied strain through the virtual points (y-dir)
#-----
mdb.models[modelName].DisplacementBC(amplitude=UNSET, createStepName=
'Tension', distributionType=UNIFORM, fieldName='', fixed=OFF,
localCsys=None, name='bc_VPy', region=
mdb.models[modelName].rootAssembly.sets['set_VPy']
, u1=0.0, u2=1.0*strain_load, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)

# Set up periodic constraint equations -----

# right and left edges
for i in range(0,len(nodes_leftEdge)):

# preparation of Coefficients
leftCoord=mdb.models[modelName].rootAssembly.sets['set_NodesLPair_' +
str(i)].nodes[0].coordinates
rightCoord=mdb.models[modelName].rootAssembly.sets['set_NodesRPair_' +
str(i)].nodes[0].coordinates

coeff1 = -(rightCoord[0]-leftCoord[0])

# x-coordinate (Ux_Vpx, H11)
mdb.models[modelName].Equation(name='constraint_xLR_' + str(i), terms=(
    ( 1.0, 'set_NodesRPair_' + str(i), 1),
    (-1.0, 'set_NodesLPair_' + str(i), 1),

```

```

        (coeff1, 'set_VPx', 1)))

# y-coordinate (Uy_Vpx, H21)
mdb.models[modelName].Equation(name='constraint_yLR_' + str(i), terms=(
    ( 1.0, 'set_NodesRPair_' + str(i), 2),
    (-1.0, 'set_NodesLPair_' + str(i), 2),
    (coeff1, 'set_VPx', 2)))

# top and bottom edges
for i in range(0,len(nodes_bottomEdge)):
    # preparation of Coefficients
    bottomCoord=mdb.models[modelName].rootAssembly.sets['set_NodesBPair_'
        + str(i)].nodes[0].coordinates
    topCoord=mdb.models[modelName].rootAssembly.sets['set_NodesTPair_'
        + str(i)].nodes[0].coordinates
    coeff2 = -(topCoord[1]-bottomCoord[1])
    # x-coordinate (Ux_Vpy, H12)
    mdb.models[modelName].Equation(name='constraint_xTB_' + str(i), terms=(
        ( 1.0, 'set_NodesTPair_' + str(i), 1),
        (-1.0, 'set_NodesBPair_' + str(i), 1),
        (coeff2, 'set_VPy', 1)))

    # y-coordinate (Uy_Vpy, H22)
    mdb.models[modelName].Equation(name='constraint_yTB_' + str(i), terms=(
        ( 1.0, 'set_NodesTPair_' + str(i), 2),
        (-1.0, 'set_NodesBPair_' + str(i), 2),
        (coeff2, 'set_VPy', 2)))

# OUTPUT REQUEST -----
# OUTPUT REQUEST: Field Output Request
mdb.models[modelName].fieldOutputRequests['F-Output-1'].setValues(
    variables=('S', 'E', 'U', 'RF', 'CF'))
# OUTPUT REQUEST: History Output Request
# History Output Request: Right Crack
mdb.models[modelName].historyOutputRequests['H-Output-1'].setValues(
    contourIntegral='right_crack', numberOfContours=1, rebar=EXCLUDE,
    sectionPoints=DEFAULT)
mdb.models[modelName].HistoryOutputRequest(contourIntegral='right_crack',
    contourType=K_FACTORS, createStepName='Tension', kFactorDirection=MERR,
    name='H-Output-2', numberOfContours=1, rebar=EXCLUDE,
    sectionPoints=DEFAULT)
# History Output Request: Left Crack
mdb.models[modelName].HistoryOutputRequest(contourIntegral='left_crack',
    createStepName='Tension', name='H-Output-3', numberOfContours=1, rebar=
    EXCLUDE, sectionPoints=DEFAULT)
mdb.models[modelName].HistoryOutputRequest(contourIntegral='left_crack',
    contourType=K_FACTORS, createStepName='Tension', kFactorDirection=MERR,
    name='H-Output-4', numberOfContours=1, rebar=EXCLUDE,
    sectionPoints=DEFAULT)

# JOB -----
jobName = 'job_crack_SH' + str(int(crack_counter + 1)) + 'P' +
str(porosity).replace('.', 'p')
    + '_AR' + str(axes_ratio).replace('.', 'p') + '_MS' +
    str(seed_mesh).replace('.', 'p') + '_CL'
    + str(crack_length_5deci).replace('.', 'p')
    mdb.Job(atTime=None, contactPrint=OFF, description='', echoPrint=OFF,
    explicitPrecision=DOUBLE, getMemoryFromAnalysis=True, historyPrint=OFF,
    memory=16000, memoryUnits=MEGA_BYTES, model=modelName, modelPrint=
    OFF, multiprocessingMode=DEFAULT, name=jobName,
    nodalOutputPrecision=SINGLE
    , numCpus=1, numGPUs=0, queue=None, resultsFormat=ODB, scratch='', type=
    ANALYSIS, userSubroutine='', waitHours=0, waitMinutes=0,
    parallelizationMethodExplicit=DOMAIN, numDomains=1)
mdb.jobs[jobName].submit(consistencyChecking=OFF)
mdb.jobs[jobName].waitForCompletion()
mdb.saveAs(pathName=subPath + str(int(crack_counter + 1)) + 'Stophole_CL'
    + str(crack_length_5deci).replace('.', 'p') + '.cae')

```

A.2 Python Script for generating 130 models with increasing crack length using finite boundary condition on the whole specimen (Stop-hole void model). Same codes were applied to the circular void model but with different part geometries.

```

# Garivalde Dominguez
# 06062018

pathName = "C:/Users/gdomingu/Python_Abaqus_Script/Whole_Model_StopholeVoid_R2/"
os.chdir(pathName)

# Library
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 math
import os

# Sketch Source
path_Dogbone =
'C:/Users/gdomingu/Python_Abaqus_Script/Whole_Model_StopholeVoid_R2/Sketch_Source/dog_bon
e.stp'
path_Stopholevoid =
'C:/Users/gdomingu/Python_Abaqus_Script/Whole_Model_StopholeVoid_R2/Sketch_Source/stophol
e_void.stp'

session.journalOptions.setValues(replayGeometry=COORDINATE,recoverGeometry=COORDINATE)

# MODEL -----
modelName = 'Unit_Circle_Void'
mdb.models.changeKey(fromName='Model-1', toName=modelName)

mdb.openStep(path_Dogbone, scaleFromFile=OFF)
mdb.models[modelName].ConstrainedSketchFromGeometryFile(geometryFile=mdb.acis,
name='Dog_bone')

mdb.openStep(path_Stopholevoid, scaleFromFile=OFF)
mdb.models[modelName].ConstrainedSketchFromGeometryFile(geometryFile=mdb.acis,
name='Stophole_void')

# Material Properties -----
materialName = 'stainless_steel'
Youngs_Modulus = 65.4e3 # Young's modulus (in MPa)
Poissons_Ratio = 0.32 # Poisson's Ratio

# Geometric Properties -----
center_to_center = 10 # center to center distance for the holes (in mm)
porosity = 0.10 # porosity
thickness = 0.0 # thickness of the plates
seed_mesh = 1 # seed-mesh (in mm)

```

```

width_plate = 2.0*center_to_center; # width of plate
height_plate = 2.0*center_to_center; # height of plate

stop_hole_radius = 0.6250 # stop hole radius (in mm)
minor_axis_hole = 0.4500 # major axis of each void
major_axis_hole = 4.625751 # major axis of each void

# ratio between major and minor axis for the holes
axes_ratio = 11.6677;

# Crack Geometries -----
# minimum hole distance (in mm)
minimum_hole_distance = center_to_center - minor_axis_hole - major_axis_hole -
stop_hole_radius;
# number of increments
num_increments = 130;
# crack length increment
crack_length_increment = (0.99*minimum_hole_distance -
0.01*minimum_hole_distance)/(num_increments/2-1);

# Displacement Load -----
strain_load = 0.001 # strain load (in mm/mm)
# displacement_load = strain_load*center_to_vertend; # displacement (in mm)
displacement_load = 0.027055

for crack_counter in range(0, num_increments):

    if crack_counter + 1.0 <= num_increments/2:

        crack_length = 0.01*minimum_hole_distance +
crack_counter*crack_length_increment; # crack length (in mm)
        print 'counter: ' + str(crack_counter + 1 )
        print 'frist region crack: ' + str(crack_length)
        print 'crack_length: ' + str(crack_length)
        print 'num_increments: ' + str(num_increments)
        print 'crack_length_increment: ' + str(crack_length_increment)

        #round crack length for naming
        crack_length_5deci = math.ceil(crack_length*1000000)/1000000;

    if crack_counter + 1.0 > num_increments/2:
        # crack length (in mm)
        crack_length = 0.01*minimum_hole_distance + (crack_counter -
num_increments/2)*crack_length_increment;
        print 'counter: ' + str(crack_counter + 1 )
        print 'second region crack: ' + str(crack_length)
        print 'crack_length: ' + str(crack_length + center_to_center +
major_axis_hole)
        print 'num_increments: ' + str(num_increments)
        print 'crack_length_increment: ' + str(crack_length_increment)

        # round off crack length for naming
        crack_length_5deci = math.ceil((crack_length + center_to_center +
major_axis_hole)*1000000)/1000000;

    subPath = pathName + 'P' + str(porosity).replace('.', 'p') + '_AR' +
str(axes_ratio).replace('.', 'p') + '_MS' + str(seed_mesh).replace('.', 'p') + '_CL'
+ str(crack_length_5deci).replace('.', 'p') + "/"

    if not os.path.exists(subPath):
        os.makedirs(subPath)
    os.chdir(subPath)

# PART -----
partName = 'whole_spec'

```

```

# PART: Dog bone
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].sketchOptions.setValues(
    gridOrigin=(0.0, 0.0))
mdb.models[modelName].sketches['__profile__'].retrieveSketch(
    sketch=mdb.models[modelName].sketches['Dog_bone'])
mdb.models[modelName].Part(dimensionality=TWO_D_PLANAR, name=partName,
type=DEFORMABLE_BODY)
mdb.models[modelName].parts[partName].BaseShell(
    sketch=mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']

# PART: Circle void
mdb.models[modelName].ConstrainedSketch(
    name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].sketchOptions.setValues(
    gridOrigin=(0.0, 0.0))
mdb.models[modelName].sketches['__profile__'].retrieveSketch(
    sketch=mdb.models[modelName].sketches['Stophole_void'])
mdb.models[modelName].parts[partName].Cut(
    sketch=mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']

#PART: Grip Partition: top and bottom
mdb.models[modelName].ConstrainedSketch(
    name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].Line(
    point1=(-20.0, 70.0),
    point2=(20.0,70.0))
mdb.models[modelName].sketches['__profile__'].Line(
    point1=(-20.0, -70.0),
    point2=(20.0,-70.0))
mdb.models[modelName].parts[partName].PartitionFaceBySketch(
    faces=mdb.models[modelName].parts[partName].faces,
    sketch=mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']

# PART: Left-Crack and Right Crack (First Regions)
if crack_counter + 1.0 <= num_increments/2:
    mdb.models[modelName].ConstrainedSketch(
        name='__profile__', sheetSize=200.0)
    mdb.models[modelName].sketches['__profile__'].Line(
        point1=(-1.0*center_to_center - major_axis_hole - stop_hole_radius, 0.0),
        point2=(-1.0*center_to_center - major_axis_hole - stop_hole_radius -
        crack_length, 0.0))
    mdb.models[modelName].sketches['__profile__'].Line(
        point1=(1.0*center_to_center + major_axis_hole + stop_hole_radius, 0.0),
        point2=(1.0*center_to_center + major_axis_hole + stop_hole_radius +
        crack_length,0.0))
    mdb.models[modelName].parts[partName].PartitionFaceBySketch(
        faces=mdb.models[modelName].parts[partName].faces,sketch=mdb.models[modelName].sketches['__profile__'])
    del mdb.models[modelName].sketches['__profile__']

# PART: Left-End-Seam1
mdb.models[modelName].ConstrainedSketch(
    name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].rectangle(
    point1=(-1.0*center_to_center, -0.001),
    point2=(-1.0*center_to_center+major_axis_hole+stop_hole_radius+
    crack_length, 0.001))
mdb.models[modelName].parts[partName].Cut(
    sketch=mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']

# PART: Right-End-Seam1
mdb.models[modelName].ConstrainedSketch(

```

```

        name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].rectangle(
    point1=(1.0*center_to_center, -0.001),
    point2=(1.0*center_to_center-major_axis_hole-
        stop_hole_radius-crack_length, 0.001))
mdb.models[modelName].parts[partName].Cut(
    sketch=mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']

# PART: Left-Crack and Right Crack (Second Region)
if crack_counter + 1.0 > num_increments/2:
    mdb.models[modelName].ConstrainedSketch(
        name='__profile__', sheetSize=200.0)
    mdb.models[modelName].sketches['__profile__'].Line(
        point1=(-1.0*center_to_center + major_axis_hole + stop_hole_radius, 0.0),
        point2=(-1.0*center_to_center + major_axis_hole + stop_hole_radius +
            crack_length,0.0))
    mdb.models[modelName].sketches['__profile__'].Line(
        point1=(1.0*center_to_center - major_axis_hole
            - stop_hole_radius, 0.0),
        point2=(1.0*center_to_center - major_axis_hole - stop_hole_radius
            - crack_length,0.0))
    mdb.models[modelName].parts[partName].PartitionFaceBySketch(
        faces=mdb.models[modelName].parts[partName].faces, sketch=mdb.models
            [modelName].sketches['__profile__'])
    del mdb.models[modelName].sketches['__profile__']

# PART: Left-End-Seam2
mdb.models[modelName].ConstrainedSketch(
    name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].rectangle(
    point1=(-1.0*center_to_center, -0.001),
    point2=(-1.0*center_to_center - major_axis_hole - stop_hole_radius
        -crack_length, 0.001))
mdb.models[modelName].parts[partName].Cut(
    sketch=mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']

# PART: Right-End-Seam2
mdb.models[modelName].ConstrainedSketch(
    name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].rectangle(
    point1=(1.0*center_to_center, -0.001),
    point2=(1.0*center_to_center + major_axis_hole
        +stop_hole_radius+crack_length, 0.001))
mdb.models[modelName].parts[partName].Cut(
    sketch=mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']

# MATERIAL -----
mdb.models[modelName].Material(
    description='Linear elastic material model',
    name=materialName).Elastic(table=((Youngs_Modulus, Poissons_Ratio),))
# SECTION -----
mdb.models[modelName].HomogeneousSolidSection(
    material=materialName,
    name='WM_CircleVoid', thickness=None)
mdb.models[modelName].parts[partName].SectionAssignment(
    region=Region(faces=mdb.models[modelName].parts[partName].faces),
    sectionName='WM_CircleVoid')
# ASSEMBLY -----
instName = 'WM_CircleVoid'
mdb.models[modelName].rootAssembly.DatumCsysByDefault(CARTESIAN)
mdb.models[modelName].rootAssembly.Instance(
    dependent=OFF,
    name=instName,
    part=mdb.models[modelName].parts[partName])

```

```

# INTERACTION -----
if crack_counter + 1.0 <= num_increments/2:
    # center of left crack at x-direction (in mm)
    left_crack_x = -1.0*center_to_center - major_axis_hole - stop_hole_radius
    - crack_length/2.0;
    # center of right crack at x-direction (in mm)
    right_crack_x = 1.0*center_to_center + major_axis_hole + stop_hole_radius
    + crack_length/2.0;
    # INTERACTION: Assign Crack Set
    mdb.models[modelName].rootAssembly.Set(
        edges=mdb.models[modelName].rootAssembly.instances[instName].edges.
    findAt(
        ((left_crack_x, 0.0, 0.0), )), name='left_crack')
    mdb.models[modelName].rootAssembly.Set(
        edges=mdb.models[modelName].rootAssembly.instances[instName].edges.
    findAt(
        ((right_crack_x, 0.0, 0.0), )), name='right_crack')
    # INTERACTION: Assign Seam
    mdb.models[modelName].rootAssembly.engineeringFeatures.assignSeam(
        regions=mdb.models[modelName].rootAssembly.sets['left_crack'])
    mdb.models[modelName].rootAssembly.engineeringFeatures.assignSeam(
        regions=mdb.models[modelName].rootAssembly.sets['right_crack'])

    # INTERACTION: Contour Integral
    # tip of left crack at x-direction (in mm)
    left_crack_tip = -1.0*center_to_center - major_axis_hole
    - stop_hole_radius - crack_length;
    # tip of right crack at x-direction (in mm)
    right_crack_tip = 1.0*center_to_center + major_axis_hole + stop_hole_radius
    + crack_length;
    mdb.models[modelName].rootAssembly.engineeringFeatures.ContourIntegral(
    collapsedElementAtTip=NONE,
    crackFront=Region(
        edges=mdb.models[modelName].rootAssembly.instances[instName].edges.
    findAt(((right_crack_x, 0.0, 0.0), )), ),
    crackTip=Region(vertices=mdb.models[modelName].rootAssemblyinstanc
es[instName].vertices.findAt(((right_crack_tip, 0.0, 0.0), )), ),
    extensionDirectionMethod=Q_VECTORS, midNodePosition=0.5,
    name='right_crack', qVectors=((
    mdb.models[modelName].rootAssembly.instances[instName].vertices.fin
dAt((right_crack_tip, 0.0, 0.0), ), (20.0, 0.0, 0.0)), ),
    symmetric=ON)

    mdb.models[modelName].rootAssembly.engineeringFeatures.ContourIntegral(
    collapsedElementAtTip=NONE, crackFront=Region(edges=mdb.models[modelName].r
ootAssembly.instances[instName].edges.findAt(((left_crack_x, 0.0, 0.0), ),
    )), crackTip=Region(
        vertices=mdb.models[modelName].rootAssembly.instances[instName].ver
tices.findAt(((left_crack_tip, 0.0, 0.0), )), ),
    extensionDirectionMethod=Q_VECTORS, midNodePosition=0.5,
    name='left_crack', qVectors=((
    mdb.models[modelName].rootAssembly.instances[instName].vertices.fin
dAt((left_crack_tip, 0.0, 0.0), ), (-20.0, 0.0, 0.0)), ),
    symmetric=ON)

if crack_counter + 1.0 > num_increments/2:

    # center of left crack at x-direction (in mm)
    left_crack_x = -1.0*center_to_center + major_axis_hole + stop_hole_radius
    + crack_length/2.0;
    # center of right crack at x-direction (in mm)
    right_crack_x = 1.0*center_to_center - major_axis_hole - stop_hole_radius
    - crack_length/2.0;
    print 'left_crack_x :' + str(left_crack_x)
    print 'right_crack_x :' + str(right_crack_x)
    # INTERACTION: Assign Crack Set

```



```

mdb.models[modelName].rootAssembly.Set(edges=mdb.models[modelName].rootAss
embly.instances[instName].edges.findAt(((left_crack_x, 0.0, 0.0), )),
name='left_crack')
mdb.models[modelName].rootAssembly.Set(
edges=mdb.models[modelName].rootAssembly.instances[instName].edges.
findAt(((right_crack_x, 0.0, 0.0), )), name='right_crack')

# INTERACTION: Assign Seam
mdb.models[modelName].rootAssembly.engineeringFeatures.assignSeam(
regions=mdb.models[modelName].rootAssembly.sets['left_crack'])
mdb.models[modelName].rootAssembly.engineeringFeatures.assignSeam(
regions=mdb.models[modelName].rootAssembly.sets['right_crack'])

# INTERACTION: Contour Integral
# tip of left crack at x-direction (in mm)
left_crack_tip = -1.0*center_to_center + major_axis_hole
+ stop_hole_radius + crack_length;
# tip of right crack at x-direction (in mm)
right_crack_tip = 1.0*center_to_center - major_axis_hole
- stop_hole_radius - crack_length;
mdb.models[modelName].rootAssembly.engineeringFeatures.ContourIntegral(
collapsedElementAtTip=NONE,crackFront=Region(edges=mdb.models[modelName].r
ootAssembly.instances[instName].edges.findAt(((right_crack_x, 0.0, 0.0),
)), ), ),
crackTip=Region(vertices=mdb.models[modelName].rootAssembly.instances[inst
Name].vertices.findAt(((right_crack_tip, 0.0, 0.0), ), )),
extensionDirectionMethod=Q_VECTORS, midNodePosition=0.5,
name='right_crack', qVectors=((
mdb.models[modelName].rootAssembly.instances[instName].vertices.findAt((ri
ght_crack_tip, 0.0, 0.0), ), (-1.0, 0.0, 0.0)), ), symmetric=ON)

mdb.models[modelName].rootAssembly.engineeringFeatures.ContourIntegral(
collapsedElementAtTip=NONE,crackFront=Region(edges=mdb.models[modelName].r
ootAssembly.instances[instName].edges.findAt(((left_crack_x, 0.0, 0.0), ),
)),
crackTip=Region(vertices=mdb.models[modelName].rootAssembly.instances[inst
Name].vertices.findAt(((left_crack_tip, 0.0, 0.0), ), )),
extensionDirectionMethod=Q_VECTORS, midNodePosition=0.5,
name='left_crack', qVectors=((
mdb.models[modelName].rootAssembly.instances[instName].vertices.findAt((le
ft_crack_tip, 0.0, 0.0), ), (1.0, 0.0, 0.0)), ), symmetric=ON)

# STEP -----
mdb.models[modelName].StaticStep(description='Uniaxial Tension in y-y direction',
name='Tension', previous='Initial')
mdb.models[modelName].steps['Tension'].setValues(adaptiveDampingRatio=None,
continueDampingFactors=False, matrixSolver=DIRECT, solutionTechnique=FULL_NEWTON,
stabilizationMethod=NONE)
#mdb.models[modelName].StaticStep(initialInc=0.001, maxInc=0.01, maxNumInc=10000,
minInc=1e-09, name='Tension', previous='Initial')

# MESH (based on Assembly)
# MESH: Seed Mesh
mdb.models[modelName].rootAssembly.seedPartInstance(deviationFactor=0.1,
minSizeFactor=0.1, regions=(
mdb.models[modelName].rootAssembly.instances[instName], ), size=seed_mesh)
# MESH: Element Type
mdb.models[modelName].rootAssembly.setElementType(elemTypes=(
ElemType(elemCode=CPS8, elemLibrary=STANDARD), ElemType(elemCode=CPS6,
elemLibrary=STANDARD)),
regions=(mdb.models[modelName].rootAssembly.instances[instName].faces.findAt((cent
er_to_center/2.0, center_to_center/2.0, thickness/2.0),),))
# MESH: Control
mdb.models[modelName].rootAssembly.setMeshControls(elemShape=QUAD,
regions=(mdb.models[modelName].rootAssembly.instances[instName].faces.findAt((cent
er_to_center/2.0, center_to_center/2.0, thickness/2.0),),))
# MESH: Generate Mesh
mdb.models[modelName].rootAssembly.generateMesh(

```

```

regions=(mdb.models[modelName].rootAssembly.instances[instName], )

# BOUNDARY CONDITION -----
# BOUNDARY CONDITION: Grips Set
mdb.models[modelName].rootAssembly.Set(
    faces=mdb.models[modelName].rootAssembly.instances[instName].faces.findAt(
        ((0,100,0),)), name='top_grip')
mdb.models[modelName].rootAssembly.Set(
    faces=mdb.models[modelName].rootAssembly.instances[instName].faces.findAt(
        ((0,-100,0),)), name='bottom_grip')

# BOUNDARY CONDITION: Top Displacement Load
mdb.models[modelName].DisplacementBC(
    amplitude=UNSET, createStepName='Tension', distributionType=UNIFORM,
    fieldName='', fixed=OFF, localCsys=None, name='top_disp',
    region=mdb.models[modelName].rootAssembly.sets['top_grip'], u1=UNSET,
    u2=displacement_load, ur3=UNSET)
# BOUNDARY CONDITION: Bottom Displacement Load
mdb.models[modelName].DisplacementBC(
    amplitude=UNSET, createStepName='Tension', distributionType=UNIFORM,
    fieldName='', fixed=OFF, localCsys=None, name='bottom_disp',
    region=mdb.models[modelName].rootAssembly.sets['bottom_grip'], u1=UNSET, u2=-
    1.0*displacement_load, ur3=UNSET)

# FIELD OUTPUT REQUEST -----
mdb.models[modelName].fieldOutputRequests['F-Output-1'].setValues(variables=('S',
    'E', 'U', 'RF', 'CF'))
# HISTORY OUTPUT REQUEST -----
# HISTORY OUTPUT REQUEST: Right Crack
mdb.models[modelName].historyOutputRequests['H-Output-1'].setValues(
    contourIntegral='right_crack', numberOfContours=1, rebar=EXCLUDE,
    sectionPoints=DEFAULT)
mdb.models[modelName].HistoryOutputRequest(contourIntegral='right_crack',
    contourType=K_FACTORS, createStepName='Tension', kFactorDirection=MERR,
    name='H-Output-2', numberOfContours=1, rebar=EXCLUDE,
    sectionPoints=DEFAULT)
# HISTORY OUTPUT REQUEST: Left Crack
mdb.models[modelName].HistoryOutputRequest(contourIntegral='left_crack',
    createStepName='Tension', name='H-Output-3', numberOfContours=1, rebar=
    EXCLUDE, sectionPoints=DEFAULT)
mdb.models[modelName].HistoryOutputRequest(contourIntegral='left_crack',
    contourType=K_FACTORS, createStepName='Tension', kFactorDirection=MERR,
    name='H-Output-4', numberOfContours=1, rebar=EXCLUDE,
    sectionPoints=DEFAULT)

# JOB -----
jobName = 'job_crack_C' + str(int(crack_counter + 1)) + 'P' +
str(porosity).replace('.', 'p') + '_AR' + str(axes_ratio).replace('.', 'p') + '_MS'
+ str(seed_mesh).replace('.', 'p') + '_CL' +
str(crack_length_5deci).replace('.', 'p')

mdb.Job(atTime=None, contactPrint=OFF, description='', echoPrint=OFF,
    explicitPrecision=SINGLE, getMemoryFromAnalysis=True, historyPrint=OFF,
    memory=16000, memoryUnits=MEGA_BYTES, model=modelName, modelPrint=
    OFF, multiprocessingMode=DEFAULT, name=jobName,
    nodalOutputPrecision=SINGLE
    , numCpus=1, numGPUs=0, queue=None, resultsFormat=ODB, scratch='', type=
    ANALYSIS, userSubroutine='', waitHours=0, waitMinutes=0,
    parallelizationMethodExplicit=DOMAIN, numDomains=1)

mdb.jobs[jobName].submit(consistencyChecking=OFF)
mdb.jobs[jobName].waitForCompletion()
mdb.saveAs(pathName=subPath + str(int(crack_counter + 1)) + 'Circle_CL' +
str(crack_length_5deci).replace('.', 'p') + '.cae')

```