3-2019

Generation and Analysis of Stop-Hole Geometries for Crack-Like Structures in Auxetic Materials

Max de Jesús Barillas Velásquez

Recommended Citation

Velásquez, Max de Jesús Barillas, "Generation and Analysis of Stop-Hole Geometries for Crack-Like Structures in Auxetic Materials" (2019). *Mechanical Engineering Master's Theses*. 39.
https://scholarcommons.scu.edu/mech_mstr/39

Santa Clara University

DEPARTMENT OF MECHANICAL ENGINEERING

DATE: **June 5, 2020**

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPEVISION BY

Max de Jesús Barillas Velásquez

ENTITLED

GENERATION AND ANALYSIS OF STOP-HOLE GEOMETRIES FOR CRACK-LIKE STRUCTURES IN AUXETIC MATERIALS.

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

Master of Science in Mechanical Engineering

[illegible]

Thesis Advisor
Dr. Michael Taylor

[illegible]

Thesis Reader
Dr. Timothy Hight

[illegible]

Chairman of the Department
Dr. Drazen Fabris

GENERATION AND ANALYSIS OF STOP-HOLE GEOMETRIES FOR CRACK-LIKE STRUCTURES IN AUXETIC MATERIALS.

by Max de Jesús Barillas Velásquez
B.S. Mechanical Engineering
Department of Structural Mechanics
University of the Central America "José Simeón Cañas"

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

March 2019

Santa Clara, California

ABSTRACT

In this investigation, the main interest was studying low porosity auxetic metamaterials generated out of linearly elastic materials, meaning bodies made out of linearly elastic materials (e.g., metals) that, due to alternating patterns of elongated voids perforated on them, exhibit a negative effective Poisson ratio (property commonly called auxeticity). This kind of metamaterials, often obtained by a pattern of elongated ellipsis, generally face the issue of presenting high stress concentration when loaded. The objective of this study is to solve this problem by adding rounded shapes (stop-holes) at the end of elongated grooves, as a replacement for the previously mentioned elongated elliptical voids. Particularly, the “superformula”, a generalized ellipse equation in polar coordinates, was utilized in this investigation as a way of parametrization to determine the shapes to be added in a flexible through way by the alteration of 6 parameters. For the process of choosing adequate parameters to ensure optimum stress concentration, firstly, a careful selection from a catalog of shapes took place. Then, static FEA simulations of a totally parametric Representative Volume Element model that included the selected shapes were executed. To do this computer scripts were developed for interconnecting the operation of multiple engineering software tools. Finally, the effect that the size of the stop-holes, thus the porosity, had over the stress induced in the material and its auxetic deformation response due to the new geometry of the pattern of voids was evaluated. The investigation successfully found shapes that produced a significant stress reduction, by reducing the stress concentration, and in the process found several corollaries and observations of the behavior of the metamaterial depending on the shape and size of the stop-holes.

TABLE OF CONTENTS

TABLE OF FIGURES

I. Introduction

Poisson’s ratio is defined as the negative ratio between the transverse and axial strain in a material under load, considering an extensional strain as positive and a compressive strain as negative [1]. Typical engineering materials have a positive Poisson’s ratio value meaning that, if an axial extensional strain is applied, the transverse directions show a shrinking effect. However, a negative effective Poisson’s ratio may exist according to thermodynamics principles applied to strain-energy theory (in three dimensions: $-1 \leq \nu \leq 0.5$), and it has been observed in several materials in nature (such as cubic metal lattices [2] [3], zeolites [4] and ferroelectric materials under electrical loads [5]). Materials that present this behavior are called “auxetics”.

From the very early stages of research in the field of auxetic materials and their properties, it has been known that geometry, at the micro and macro levels, plays an important role in controlling the response of the material. One early example of this is the work of Lakes [6], utilizing reentrant cell geometries in foams to obtain negative effective Poisson’s ratios. After that, the study of auxeticity turned also into the study of metamaterials, bodies which effective properties are controlled by the properties of its constituent bulk material, but also, by their macrostruct, ure. An early example of this are the analytical efforts of Wei [7] [8] [9], that pointed out the possibility of generating auxeticity in elastic composite materials by introducing ellipsoidal inclusions that induce this behavior to linear elastic materials. Also, the more recent observations of Bertoldi [10] and Overvelde [11], in their experiments introducing various porous shapes to a elastomeric matrix, identified how the shape controlled the effective Poisson’s ratio and other properties allowing, in some cases, auxetic behavior.

Recent interest in periodic cellular structures created via planar tessellation has increased [10] [11] given the potential properties and applications they have. In particular the ones that lead to auxetic behavior, of interest for this investigation, show good potential for use in improved acoustics [12], improved penetration properties [13], energy absorption [14], among others [15]. This also awakened a recent interest in applying this type of porous structure to metals and other linearly elastic materials, like rubber, to control their effective properties over the bulk properties of the material. Some of the recent experimental, numerical and analytical efforts in this field are diamond and star shaped voids introduced to sheets of isotropic materials [16] and tessellated skeletal structures 3D printed in Ti-6Al-4V [17].

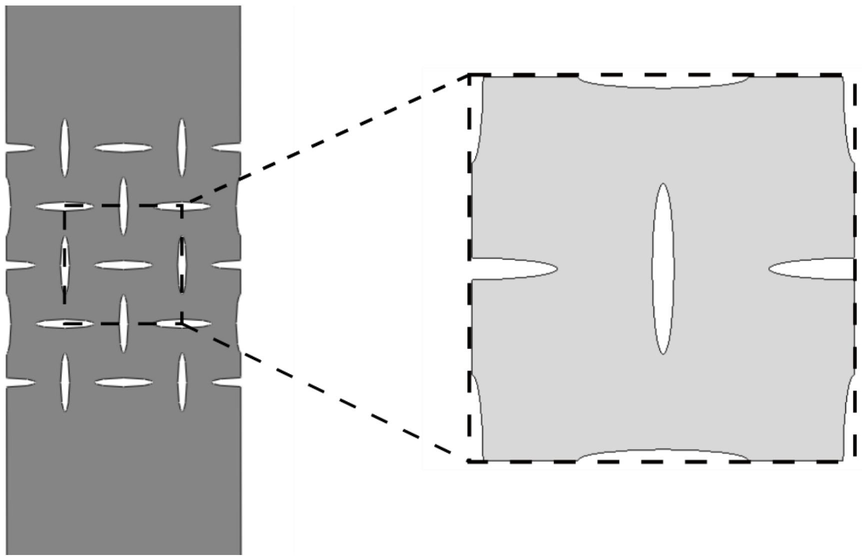

Figure 1 Periodic geometry used as a baseline comparison point. Inset shows a representative volume element, or base cell.

Taylor, et al., [18] showed a structured path to design porous periodic 2D structures with specific Poisson's ratios by introducing a low porosity pattern of alternatingly oriented elliptical voids, and the correlation of high aspect (b/a) ratio voids with auxetic behavior. Even though several different geometries have been introduced as periodic voids to induce auxetic behavior in metallic materials, most of them have in common a crack-like shape [16] [17] [19] producing high stress concentrations, as can be seen in Figure 1, a geometry derived from Taylor's work [18]. These kind of geometries, however, are known to have high or even unbounded stress concentrations (in the case of cracks) as it can be seen in Figure 2 composed from figures extracted from a commonly used elasticity textbook [20]

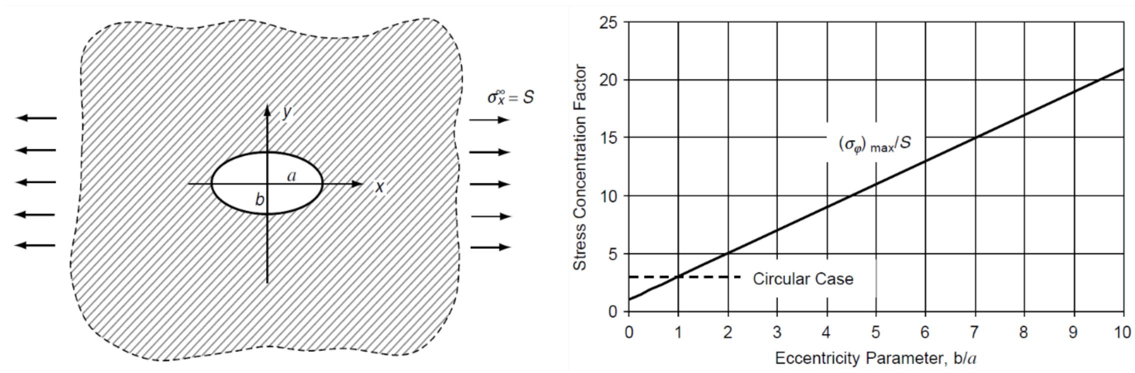

Figure 2 Analytical stress concentration of an infinite plate with an elliptical hole (left: representation of the geometrical and loading condition; right: analytical stress concentration curve as a function of the aspect ratio of the elliptical void) (reproduced without permission) [20]

Given the potential benefits that auxetic structures have, there is a rising interest in addressing the problem of high stress concentrations while maintaining auxeticity. Different approaches to a solution have been attempted, e.g., by cleverly choosing the shape of the void [21] or a stress reducing geometry to be added at the ends of the void [22]. Also, it has been shown how these features and the auxetic behavior might affect the fatigue life of the part, i.e., through numerically showing how the shapes selected for inducing this behavior increased or decreased the likelihood of a crack to propagate from the edges of it [21]. This is directly related to the stress concentration associated with the geometry of the voids. Improved void shapes might lead to improved auxetic materials with better fatigue behavior.

The goal of this investigation is to find, in a rigorous way, improved auxetic void geometries with reduced stress concentration, by adding rounded shapes at the ends of elongated crack-like shapes. The final improved geometry must also keep or improve the auxetic behavior that these elongated features produce in metallic materials when introduced in a low porosity alternating periodic pattern. Achieving this goal is expected to improve the fatigue behavior of auxetic structures induced by low porosity patterns of voids.

Adding a rounded shape to the ends of crack-like geometries is inspired by a common remedy to arrest crack propagation: drilling circular “stop” holes at the end of the crack, increasing the radius of the tip of the void. Several attempts to identify better geometries for slowing down crack propagation have been studied [23] [24], showing that it is possible to considerably decrease the stress at the tip of crack-like geometries. An improvement over previous research on reduction of stress concentration might be found in considering as many stop-hole shapes as possible.

The so-called “super-formula”, a generalized equation derived from the equation of the ellipse and super-ellipse equations that describes in polar coordinates a large number of shapes controlled by a total of 6 constants [25], was selected in this investigation as the way to parametrize the geometries to add at the ends of the voids. The benefit that this equation offered to the goal of this investigation is its versatility at generating shapes, ranging from quasi-straight shapes to rounded organic shapes, including circles.

To achieve the goal of this investigation, finite element simulations were performed on multiple periodic geometries to compare their behavior in terms of stress and average Poisson's ratio under uniaxial loading. The periodic geometries to be considered in this comparison were based on previous periodic geometries with a proven effect of inducing auxetic behavior (see Figure 1) where the elliptical voids were replaced with straight slots with rounded shapes added at the ends of the slots that are the result of the superformula for a specific set of parameters (see Figure 3). Different geometric constraints and variables were considered to set uniform conditions and ensure a fair comparison of the different geometries and determine which of them present the best improvement. Several steps of screening and filtering were required to narrow down the infinite number of shapes that can be obtained from the superformula by altering the parameters. Finally, the chosen geometry is expected to present an improved behavior independently of the size of the shape added to the end of the slot, compared to the other options analyzed at similar conditions, at least for the geometrical conditions stablished in terms of the dimensions of the slot.

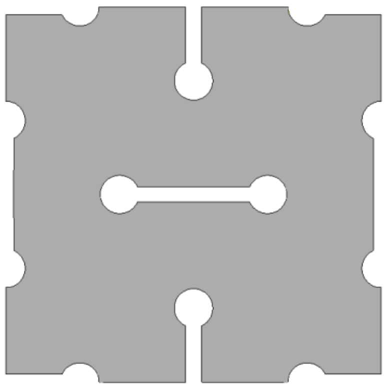

Figure 3 Representative periodic cell of slotted shape with circular stop-holes

In Chapter II, the details of the procedure followed to screen and choose the most adequate stop-hole geometry by looking at the different possibilities that the super formula can offer, controlled by its 6 parameters are discussed. This will include, but is not limited to, a brief explanation of the superformula as a parametrization tool for generating stop-hole shapes, the description and delimitation of the void shape and details on the procedure followed to use a parametric finite element model to simulate the behavior of the selected void shapes and compare them. Next, in Chapter III, I will present the results obtained from the procedure described previously, focusing on the mesh refinement study performed to ensure the quality of the results, shape comparisons at specific conditions (i.e., the porosity the compounded void represents and geometric delimitations) and a broader study of the behavior of the shapes (i.e., auxeticity and peak stress) against the variation of the geometric conditions (i.e., porosity). Finally, in Chapter IV, the most important findings obtained from the procedure followed are listed and discurssed, as well as, suggestions that this work may leave for future research work in this field.

II. Procedures

2.1 Geometrical parametrization: Superformula

To accomplish the objective of this research, finding improved geometries to induce auxetic behavior in metallic materials, the approach selected was to use slot voids, similar in behavior to an ellipse but easier to manufacture, in a perpendicular alternating pattern, and add at the tips rounded stop-hole geometries that may reduce the stress. As mentioned in Chapter I, a similar procedure is followed to slow down crack growth by adding circular holes at the tip of the crack, called stop-holes; however, the main motivation of using only circular stop-holes is that in-site manufacturing for aeronautics is limited to certain operations most times. In this investigation the purpose is to find better geometries than the circle for stress reduction knowing already how flexible the manufacturing process needs to be to generate auxetic structures. Therefore, a way to parametrize the geometries to add at the tips was required. The main requirement of the stop-hole geometry is that it must be capable of reducing the stress concentration at the tips, this is directly related to having a large curvature radius in the orientation of the stress concentration. A circle and other rounded geometries may accomplish this task; however, as their curvature radius increases, the behavior of the overall structure may be affected negatively. Some of the foreseen possible drawbacks that adding these rounded geometries may produce are: causing even higher stresses than the ones already present in the structures with elliptical voids, weakening the thinnest sections of the structure by reducing the distance between the voids or reducing the auxetic response achieved by introducing the slot pattern in the first place. Therefore, achieving this improved behavior while keeping overall similar geometric conditions was also a requirement. Given this objective and limitations, a generalization of an ellipse called the superformula [25] was selected as a way to parametrize the possible stop-hole geometries due to its capability to generate rounded shapes, like the circle, but also limitless geometries that may comply with these conditions.

The superformula is given in polar coordinates by the equation [25]:

$$r_{(\theta)} = \left(\left| \frac{\cos\left(\frac{m\,\theta}{4}\right)}{a} \right|^{n_2} + \left| \frac{\sin\left(\frac{m\,\theta}{4}\right)}{b} \right|^{n_3} \right)^{-\frac{1}{n_1}} \tag{1}$$

where θ is the angle and $a, b, m, n_1, n_2\ and\ n_3$ are parameters to control the geometry to be generated.

2.1.1 Shape selection and exploration procedure

Due to the limitation of trying to maintain uniformity in geometry and behavior, some restrictions in the dimensions of the overall geometry were enforced. We restrict the slot width to 0.9 mm (dimension considered due to manufacturing limitations and the condition of keeping high equivalent aspect ratios in order to maintain the auxetic behavior [18]) and the overall length of 9.9 mm (keeping an equivalent aspect ratio of 11 and a constant ligament distance). These dimensions are illustrated in Figure 4.

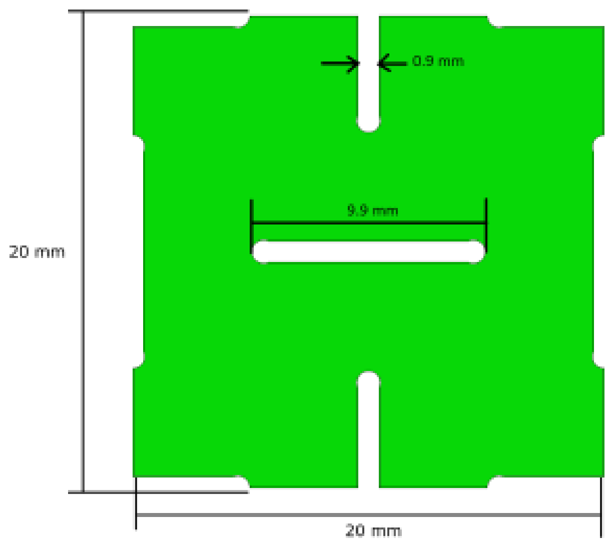

Figure 4 General periodic cell (RVE) geometry without stop-holes and enforced dimensions

Also, previous investigations in the field of introducing auxetic behavior through tessellation of voids have shown the impact that porosity has on the behavior of the body (e.g., auxeticity) [18]. In this investigation, as in previous ones, the porosity of the Representative Volume Element (RVE) is defined as seen in equation (2). It can be described as the ratio between the area of the RVE considering the voids and the area of RVE without them (i.e. the area of the matrix containing the voids). Due to the fact that porosity affects the behavior of the RVE, and to be able to compare the results of this investigation to the results of previous investigations, it has been introduced as a main variable of comparison.

$$Porosity = \frac{A_{voids}}{A_{matrix}} \tag{2}$$

Porosity is a useful measure considering the intention of adding at the tips of the single slot a geometry generated by the superformula (see (1)) where changes in the exponents

(n_1, n_2 *and* n_3) causes significant changes in the topology of the result, but also changes in the size and superficial area of the geometry generated. This means that a desired change in the stop-hole geometry, i.e., to optimize the curvature radius of a certain region, may also cause significant changes in the area of the void (A_{void}), changes that may result in undesirable changes in the behavior of the RVE given that the area of the matrix without voids (A_{matrix}) is already fixed. Therefore, the porosity of the RVE was used in this investigation as a way to control and define the scale factor that must be applied to the resulting geometry of the superformula in each comparison case. Considering the already mentioned constraints to the slot geometry, the porosity of the RVE is, in general, controlled by the superficial area of the feature to be added at the tips that is directly related to the square of the scale factor applied to the resulting geometry of the superformula.

2.1.2 General shape comparison

After defining the framework of comparison for the different proposed geometries, a first selection of parametric stop-hole shapes was selected for this exploration. Knowing the capabilities of the superformula for yielding an indefinite number of geometries, by changing any of its 6 parameters, a selection of 25 different geometries were chosen to better understand the behavior of the superformula and its parameters. For that reason, the selected sets of parameters display several unique shapes that can be achieved by the superformula. The 25 sets of parameters and their corresponding plotted geometries are shown in Figure 5.

After carefully inspecting the 25 shapes, it was clear the capability of the superformula to generate unique shapes, but also shapes that are similar with each other in their characteristics (e.g., number of inflexions and curvature). The objective was to reduce the field of the search from the 25 obtained shapes to 10, so that the comparison is more manageable. To do it in this first screening of geometries, the intention was to select geometries as unique as possible while eliminating those that may represent unwanted behavior, either by increasing the stress or increasing the Poisson's ratio, and those that may be unfeasible from a manufacturing point of view. By applying the mentioned logic, the 10 highlighted shapes in Figure 5 were selected (for reference, the 10 selected shapes as part of the RVE geometry are shown in Figure 6). Keeping in mind that the remaining steps of analysis are applicable to any parametric shape, reducing the search space is just a way to make the analysis more tractable.

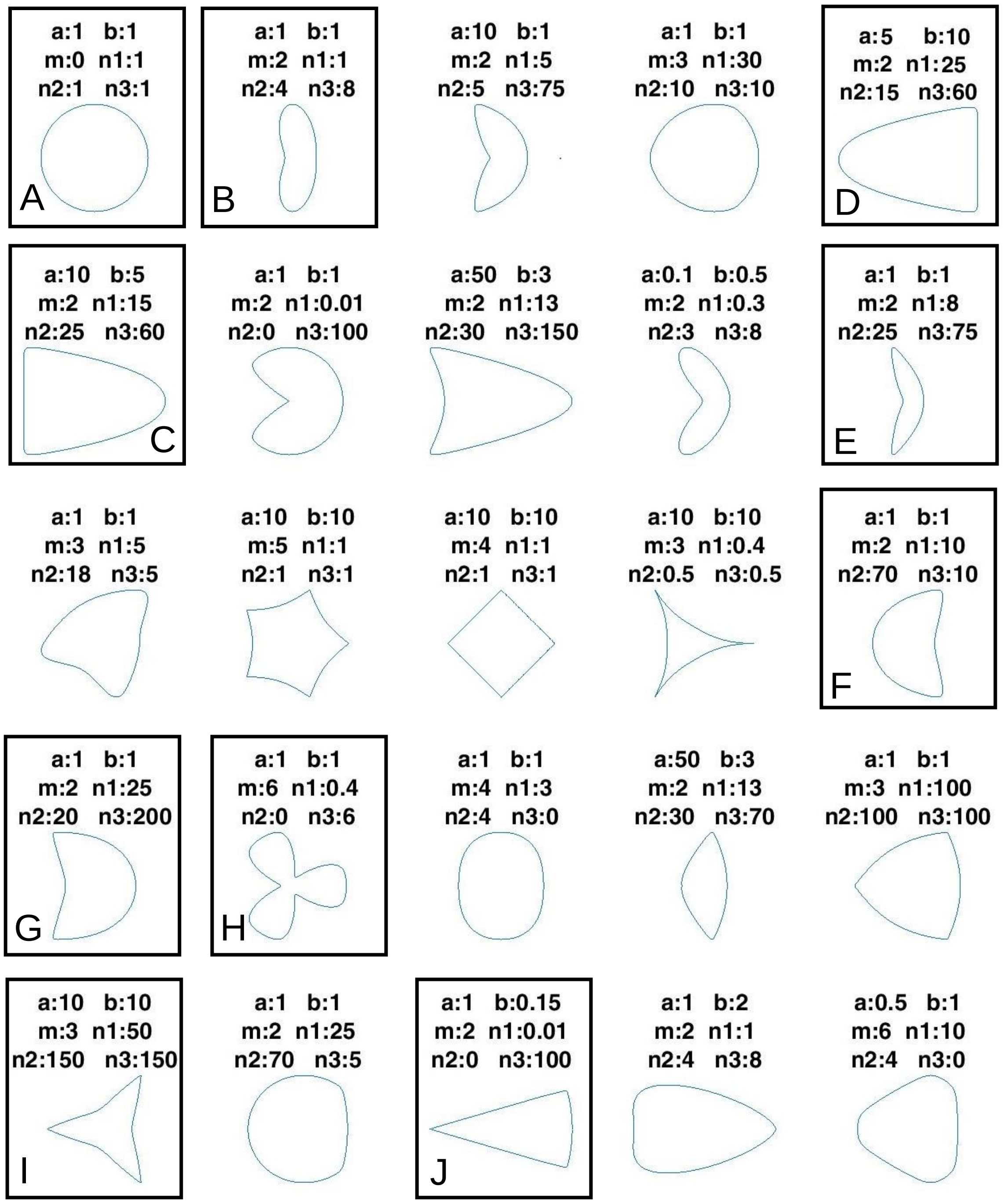
a:1 b:1 m:0 n1:1 n2:1 n3:1
A
a:1 b:1 m:2 n1:1 n2:4 n3:8
B
a:10 b:1 m:2 n1:5 n2:5 n3:75
a:1 b:1 m:3 n1:30 n2:10 n3:10
a:5 b:10 m:2 n1:25 n2:15 n3:60
D
a:10 b:5 m:2 n1:15 n2:25 n3:60
C
a:1 b:1 m:2 n1:0.01 n2:0 n3:100
a:50 b:3 m:2 n1:13 n2:30 n3:150
a:0.1 b:0.5 m:2 n1:0.3 n2:3 n3:8
a:1 b:1 m:2 n1:8 n2:25 n3:75
E
a:1 b:1 m:3 n1:5 n2:18 n3:5
a:10 b:10 m:5 n1:1 n2:1 n3:1
a:10 b:10 m:4 n1:1 n2:1 n3:1
a:10 b:10 m:3 n1:0.4 n2:0.5 n3:0.5
a:1 b:1 m:2 n1:10 n2:70 n3:10
F
a:1 b:1 m:2 n1:25 n2:20 n3:200
G
a:1 b:1 m:6 n1:0.4 n2:0 n3:6
H
a:1 b:1 m:4 n1:3 n2:4 n3:0
a:50 b:3 m:2 n1:13 n2:30 n3:70
a:1 b:1 m:3 n1:100 n2:100 n3:100
a:10 b:10 m:3 n1:50 n2:150 n3:150
I
a:1 b:1 m:2 n1:25 n2:70 n3:5
a:1 b:0.15 m:2 n1:0.01 n2:0 n3:100
J
a:1 b:2 m:2 n1:1 n2:4 n3:8
a:0.5 b:1 m:6 n1:10 n2:4 n3:0

Figure 5 Geometries obtained from the 25 sets of parameters. Boxes highlight the 10 selected shapes

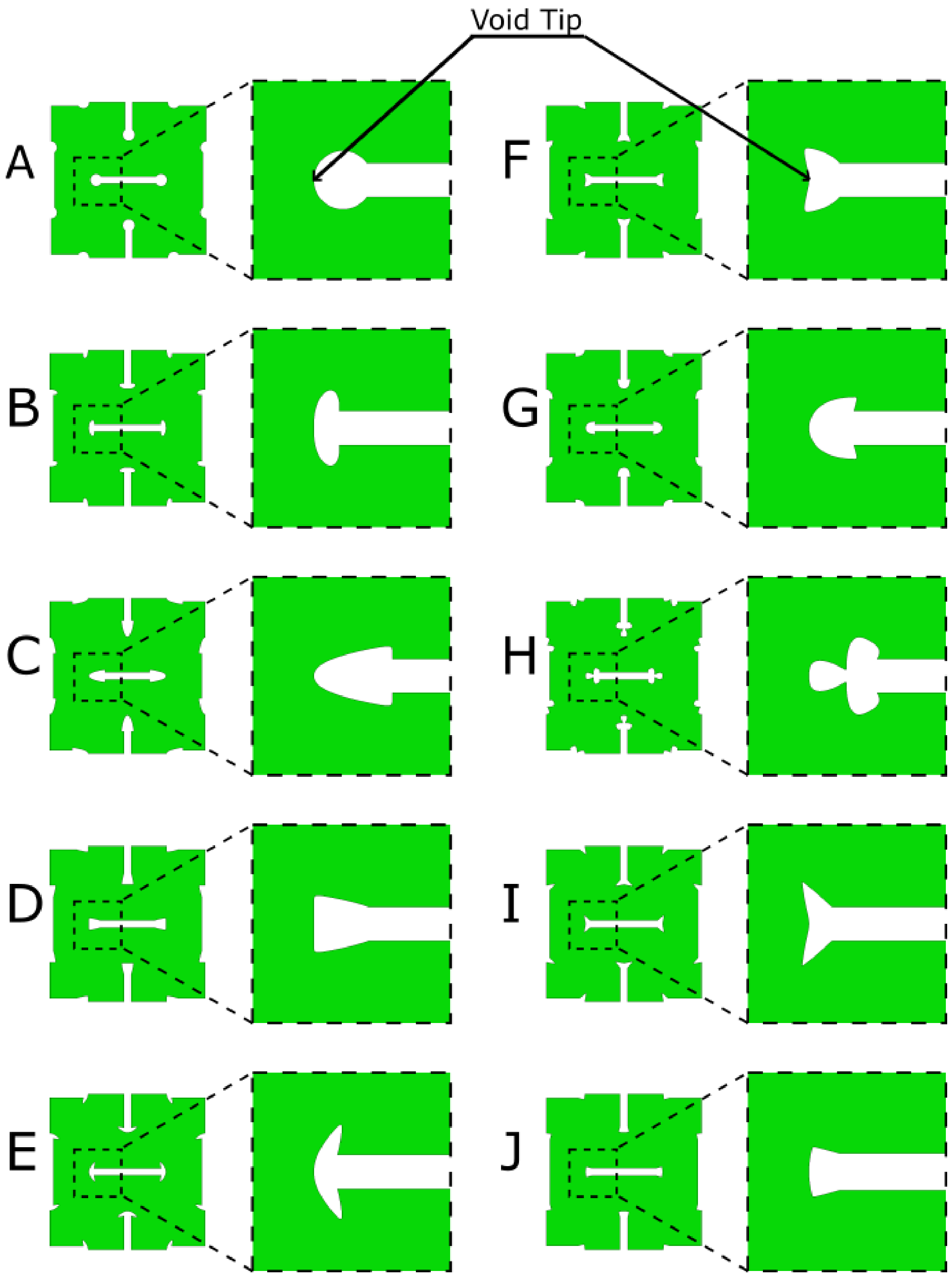

Figure 6 RVE models of the 10 selected shapes used as stop-holes at a porosity of 10% (Insets show superformula shapes added at the slot ends)

2.2 Modeling and simulation

A finite element model was used as a numerical tool to determine the behavior of the different geometries under uniaxial loading. The two magnitudes that were used to compare the behavior of each of the shapes were the effective Poisson's ratio of the structure and the peak normal stress in the vertical direction; as they represent the two main interests of the investigation, reducing the stress concentration and maintaining or improving the auxetic properties of the meta-material. In this case, the package used was ABAQUS/Standard, due to its flexibility, capabilities and the familiarity of the author with it. However, the finite element method has some limitations when it comes to these kinds of periodic structures of geometries with critically sharp geometrical changes like thin voids, due to FEM dependency on uniform and smooth variable fields to maintain its approximation accuracy. Therefore, in order to have appropriate and accurate results, it requires a fine discretization of the model. However, this may result in unfeasible computational times due to the increase in the required numeric operations. This limitation was alleviated in this investigation by using a Representative Volume Element with periodic boundary conditions and carefully reviewing the mesh parameters of the model, both procedures are discussed in this document.

As mentioned, a Representative Volume Element was used to approximate the behavior that the introduction of the void pattern in the material would cause. These kind of models have been implemented to understand the response of materials with periodic patterns like these in previous work [18] [21] [22]. The RVE simulation models incorporate the assumption of an infinite body that can be represented by the same partial geometry, known as the cell, being reproduced indefinitely in every direction. This assumption can be considered as a good approximation of real conditions when the features, in this case voids, and its pattern are small compared to the size of the body. In this case, for example, being a 2-dimensional model, it refers to reproducing the same geometry indefinitely in the x and y directions.

In practice for RVE models, where only one cell is modeled, it is required to set in place constraints for the boundary of the model known as periodic boundary conditions. To ensure the periodicity of the model and its behaviors, two main conditions must be imposed [26]. The first is that the displacement at parallel boundaries must be equal in magnitude and direction, this limitation ensures the continuity in the displacement field meaning that gaps or overlaps will not form at the boundaries between the RVEs forming the body after the deformation takes place. The second condition is that the tractions at parallel boundaries must be equal in magnitude but opposite in direction, these conditions impose equilibrium at the RVE in analysis and ensures action-reaction at the boundary between one RVE and the next one.

In order to implement these conditions, the displacement field inside of the RVE ($\boldsymbol{u}_{(\boldsymbol{x})}$) can be modeled as uniform average strain tensor ($\bar{\boldsymbol{\varepsilon}}$), that represents the average condition of the whole body, times the position vector ($\boldsymbol{x}$) inside the RVE and, added to that, an unknown local deviation from average displacement distribution ($\boldsymbol{u}^{*}_{(\boldsymbol{x})}$) that depends on the geometrical and material properties of the RVE. This is described by [27]:

$$\boldsymbol{u}_{(\boldsymbol{x})} = \bar{\boldsymbol{\varepsilon}}\,\boldsymbol{x} + \boldsymbol{u}^{*}_{(\boldsymbol{x})} \tag{3}$$

Then, to understand what happens in the boundary of the RVE the following equations can be formulated for parallel opposite boundaries.

$$\boldsymbol{u}^{(+\boldsymbol{j})}_{(\boldsymbol{x})} = \bar{\boldsymbol{\varepsilon}}\,\boldsymbol{x}^{(+\boldsymbol{j})} + \boldsymbol{u}^{*(+\boldsymbol{j})}_{(\boldsymbol{x})} \quad \& \quad \boldsymbol{u}^{(-\boldsymbol{j})}_{(\boldsymbol{x})} = \bar{\boldsymbol{\varepsilon}}\,\boldsymbol{x}^{(-\boldsymbol{j})} + \boldsymbol{u}^{*(-\boldsymbol{j})}_{(\boldsymbol{x})} \tag{4}$$

Where the superscript (+j) means the j boundary at one side of the RVE, while (-j) represents that boundary's counterpart. Also, consider that in this relation the unknown displacement distribution in both boundaries must be equal to each other to maintain the first condition of periodic boundary conditions.

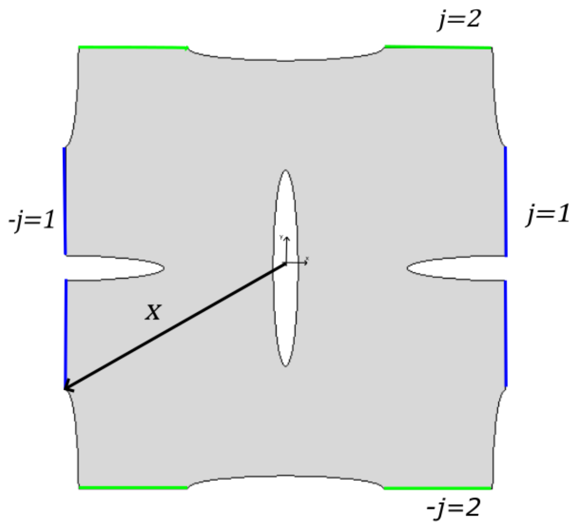

Figure 7 Location of the periodic boundary conditions for a generic RVE of an auxetic structure

Therefore, to relate the displacement in each of the parallel boundaries the displacement between them can be stated as follows:

$$\Delta\boldsymbol{u}_{(\boldsymbol{x})} = \bar{\boldsymbol{\varepsilon}}\,\Delta\boldsymbol{x} \tag{5}$$

This relation is independent from the unknown displacement distribution that depends on the geometry and the material properties, allowing a direct relation between the displacements of each boundary only as a function of the average strain tensor. This can be expressed as if external conditions applied over the body can't be introduced in the model through the boundaries, so the main external condition applied in this model is the average strain ($\bar{\varepsilon}$).

$$\bar{\varepsilon} = \begin{bmatrix} \varepsilon_{xx} & \varepsilon_{xy} \\ \varepsilon_{yx} & \varepsilon_{yy} \end{bmatrix} \tag{6}$$

In the particular case of this investigation, this was set to be a simple compressive strain in the vertical direction (represented by ε_{yy} in (6)), this means that on average the shear strain was restrained to be zero (represented by ε_{xy} & ε_{yx} in (6)); however, the normal strain in the horizontal direction in the average strain tensor of the RVE was allowed to be defined by the response of the system in the FEA simulation (represented by ε_{xx} in (6)). This allowed an equivalent external condition for all the simulations, giving a framework for comparison between the different geometries to be suggested.

To implement periodic boundary conditions into the ABAQUS/Standard simulation model in the form presented in equation (5), it is required to relate in a discrete manner the displacement of each node in one parallel boundary with its counterpart in the opposite boundary, meaning the node that shares the same position coordinate parallel to the boundaries in question. This can be achieved using the ABAQUS/Standard utility of displacement constraint equations. However, several hundreds or even thousands of equations relating the displacement of paired nodes in the top and bottom boundaries respectively, as well as the right and left boundaries, are required. This may result in an impractical workload to approach by the GUI of ABAQUS/Standard, therefore it is necessary to use the python scripting capabilities that ABAQUS/Standard offers to automate this process. This also represents benefits from the repeatability point of view, since this first exploration may represent the comparison of hundreds of geometries, as will be explained in further detail later in this document. Further explanation of the python code used to execute the model in ABAQUS standard can be found in Appendix A .

However, using the ABAQUS/Standard utility of displacement constraint equations also presents a different kind of limitation since, as its name implies, the utility only allows displacement constraints. Equation (5) also involves the average strain tensor, which cannot be directly modeled by this utility. Therefore, the work around used in this case is the creation of a set of two virtual points (Vpx and Vpy), which are out of the current RVE model and do not interact with it other than through the constraint equations set for the displacements of the nodes at the parallel boundaries. These virtual points account for set of displacements that are going to play the role of the components of the required strain tensor, this is better described by equation (7) for a 2-dimensional system.

$$\bar{\boldsymbol{\varepsilon}} = \begin{bmatrix} u_x^{Vpx} & u_x^{Vpy} \\ u_y^{Vpx} & u_y^{Vpy} \end{bmatrix} \tag{7}$$

Where $u_x^{V_{px}}$ is the displacement in the horizontal direction of the virtual point V_{px}, which represents the normal strain ε_{xx}; $u_y^{V_{px}}$ is the displacement in the vertical direction of the virtual point V_{px}, which represents the shearing strain ε_{xy}; similarly it occurs with the displacements $u_x^{V_{py}}$ and $u_y^{V_{py}}$ of the virtual point V_{py}, which are related to the normal and shearing strains ε_{xx} and ε_{yx} respectively. In this way, the equations constraining the displacement of each node for each pair of parallel boundaries can be written as shown in (8) & (9).

$$u_x^{(+n)} - u_x^{(-n)} - (u_x^{(Vpx)} * \Delta x + u_x^{(v_p y)} * \Delta y) = 0 \qquad (8)$$

$$u_y^{(+n)} - u_y^{(-n)} - (u_y^{(Vpx)} * \Delta x + u_y^{(v_p y)} * \Delta y) = 0 \qquad (9)$$

From the previous discussion can be inferred also that from the information of the virtual points displacements the effective Poisson's ratio of the body can be calculated. This is because, as mentioned, the virtual points displacements are a representation of the average strain tensor; therefore, the negative of the fraction of the normal vertical strain, set to be defined by the FEA model, and the normal horizontal strain represent the effective Poisson's relation for the body. This is better represented by (10).

$$\nu_{effective} = -\frac{\varepsilon_{yy}}{\varepsilon_{xx}} = -\frac{u_y^{Vpy}}{u_x^{Vpx}} \qquad (10)$$

Some other assumptions and conditions taken to perform the simulations using the finite element method were properties of a linear elastic bulk material (Young's modulus of 200,000 MPa and Poisson's ratio of 0.3, similar to those of common steels), and a 2-dimensional simplification in plane stress (due to thinness) of the body (Discretized in 6 nodes triangular elements, CPS6, with unit thickness and neither reduced integration nor hour-glass implementations).

2.2.1 Software tools: MATLAB-python-ABAQUS interaction

As discussed in Section 2.1, ABAQUS/Standard capabilities of automating, using python scripting to set up the model and execute the simulation, were used to implement the periodic boundary conditions required in for the RVE model. Also, in Section 2.1 it was discussed that the interest of this investigation required the generation of multiple models using a parametric geometry and several combinations of parameters and, then, comparing their behavior thru the simulations, this task may become intractable as the number of models to be simulated increases. In this Section, some of the details of how this series of models and simulations were automated is discussed, while also keeping track of their meaningful results to compare them in an structured way.

Due to limitations of the python interface used by ABAQUS/Standard and lack of knowledge of the author in the usage of python as a programming language to process data, several MATLAB scripts were developed to perform different tasks that required batch runs of simulations, data acquisition and operation, and the generation of different plots. However, a piece of code to act as the interface of the interaction between the MATLAB based operations and python-ABAQUS simulations was required. This code was developed as a MATLAB function that required as input the parameters to be written in python-ABAQUS code, in this case the superformula parameters. Then, the output of the MATLAB code was the values of specific variables of the simulations, as they are the Porosity of the RVE, Poisson's Ratio, maximum and minimum stresses in each direction near the tip of the voids and the Number of Elements.

The general behavior of this interaction is presented in the diagram shown in Figure 8.

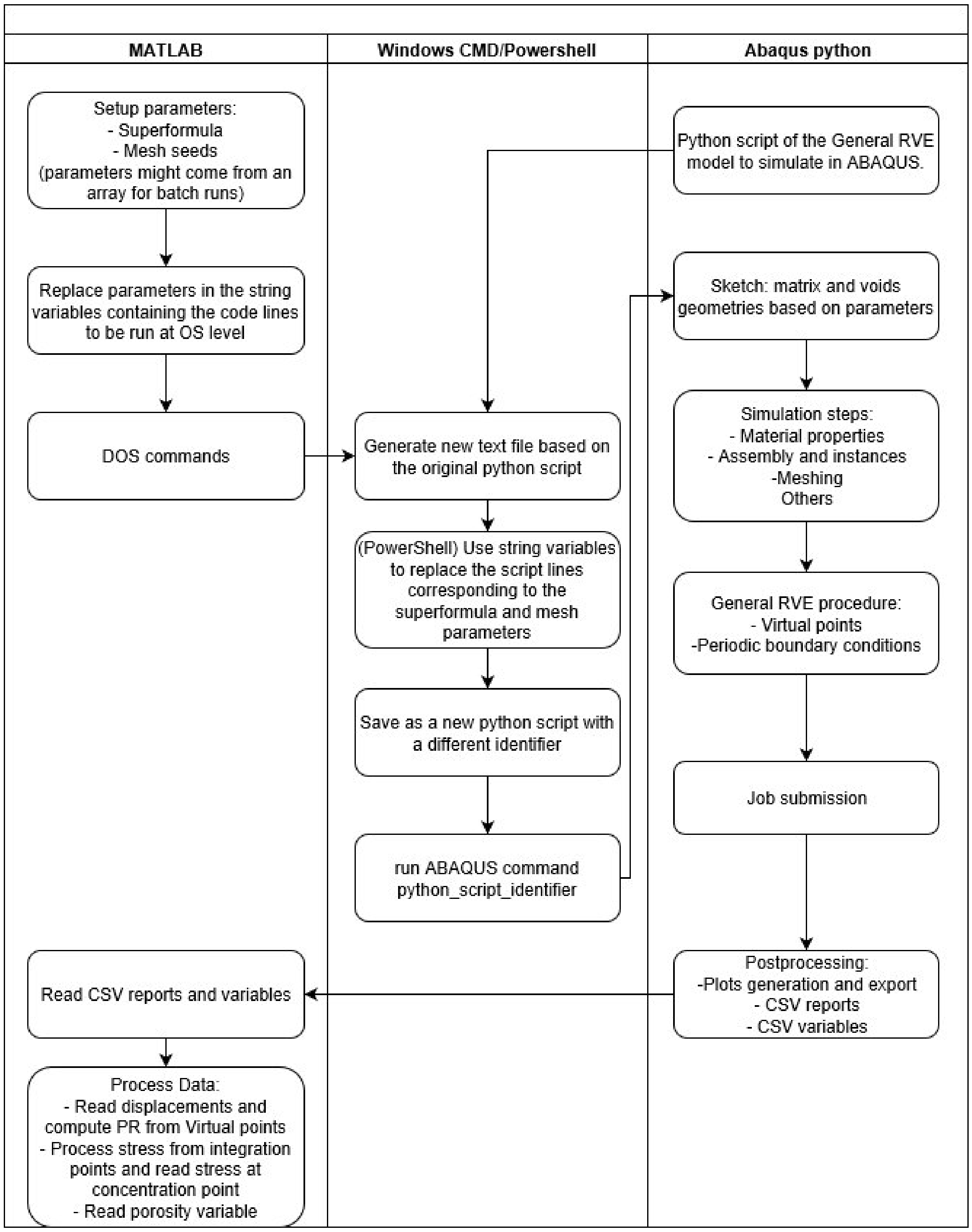
MATLAB
Windows CMD/Powershell
Abaqus python
Setup parameters: - Superformula - Mesh seeds (parameters might come from an array for batch runs)
Replace parameters in the string variables containing the code lines to be run at OS level
DOS commands
Generate new text file based on the original python script
(PowerShell) Use string variables to replace the script lines corresponding to the superformula and mesh parameters
Save as a new python script with a different identifier
run ABAQUS command python_script_identifier
Python script of the General RVE model to simulate in ABAQUS.
Sketch: matrix and voids geometries based on parameters
Simulation steps: - Material properties - Assembly and instances -Meshing Others
General RVE procedure: - Virtual points -Periodic boundary conditions
Job submission
Postprocessing: -Plots generation and export - CSV reports - CSV variables
Read CSV reports and variables
Process Data: - Read displacements and compute PR from Virtual points - Process stress from integration points and read stress at concentration point - Read porosity variable

Figure 8 MATLAB-python-ABAQUS interactions and procedure diagram

2.2.2 Mesh refinement study

Before running the first batch of simulations a revision of the mesh parameters used to perform the finite element simulations was necessary. At this point, the geometries that are going to be simulated are well defined; however, some of the selected shapes for the voids still present sensibly sharp geometries at the tips. Therefore, a refined discretization is required, at least in the neighborhood of these features, to increase the accuracy of the results to an acceptable level. Refining the mesh to ensure a good representation, not only of the geometry, but also of the variable fields of interest (i.e., displacement and stress), while also keeping the computational load as low as possible, requires making significant decisions on how to mesh the model. To make the best decisions possible, a rigorous mesh study is necessary. A typical resulting mesh of the process to be described, and which results are discussed in Section 3.1, is presented in Figure 9.

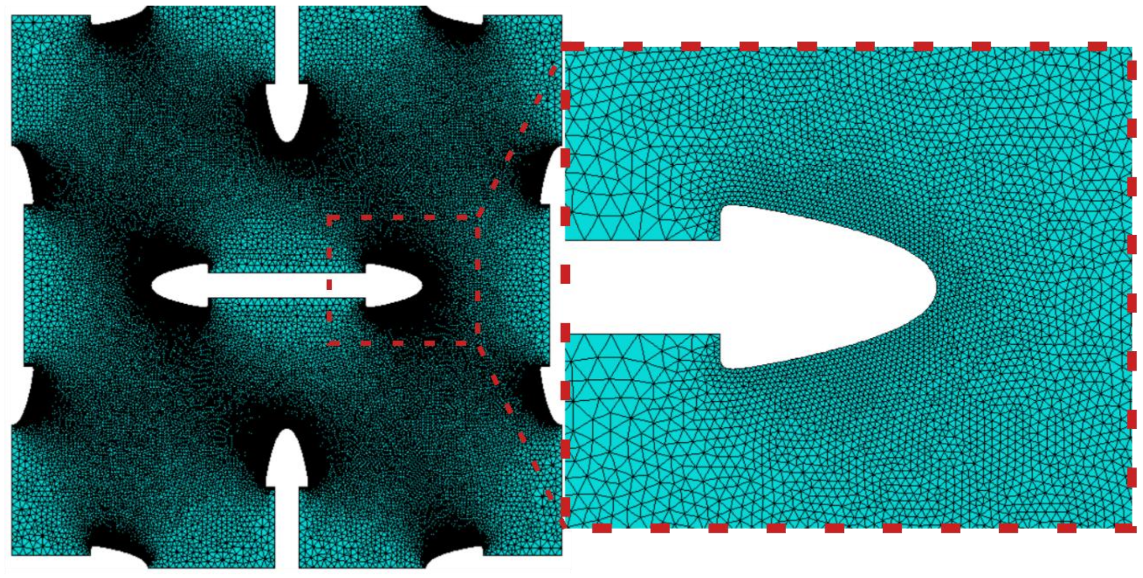

Figure 9 Typical mesh obtained by applying the mesh refinement procedure implemented.

In regular circumstances some FEA software offer tools to determine a good meshing solution for the given case, ABAQUS/standard, it has a built-in adaptive mesh refinement tool. However, in the case of the implemented periodic boundary conditions, it was required to relate the displacement of each node in the boundary edges with its symmetric counterpart in the opposite edge of the boundary, this usually causes conflict with the automated mesh refinement tool given that it changes the definition of the nodes of the mesh at the boundaries without reconsidering the relations that this nodes must have with their counterparts. Due to this fact it was not used in this investigation.

The meshing algorithm that ABAQUS/Standard uses allows the user to interact with the level of refinement to be obtained through different parameters, some of them, and the one to be used to perform this study, are the general seed size and the edge seed size. The general seed size refers to the distance between the starting points of the mesh to be created located in all the edges of the model, while the edge seed size refers to the same distance, but only of the starting points of the mesh that lie in particular edges of the model chosen by the user. Another important definition in this mesh study is that the output variable considered in this revision was the normal stress in the vertical direction (S22) taken from an integration point close to the stress concentration region and with the highest stress value. In this case, the task of interest was modifying the edge seeding of the edges generated by the superformula function for different geometries, while the general seed size is kept to reasonable value determined by previous experience, to minimize the error in the output variable of stress. However, to determine the error in the output variable of stress, several procedures can be applied. The one chosen in this investigation is explained in the following paragraphs.

The normal stress in the vertical direction at the void's tip versus number of elements were analyzed under the idea that the greatest source of error in the model was discretization error, and that other error, such as computer truncation error, was negligible or not relevant for the comparison. For linear behaviors, the error of the finite elements method with respect to the average characteristic length of the elements in the model for any field quantity can be represented as $O(h^{p+1-r})$ [28] where h represents the average characteristic length, p the degree of the highest complete polynomial in the elements and r the derivative order of the field quantity in question. Considering a second order polynomial in the elements (due to the 6 node triangles in use) and that the field in question is the stress, which represents the first derivative of the displacement field (solution variable of the finite element code applied) the expected order of the error is $O(h^2)$.

Therefore, when plotting the stress at a specific point in a structure against the square effective size of the elements in the model a linear tendency can be expected. This consideration was used in this investigation to determine the error present in the FEA simulation.

According with the previous explanation, it is required to assign an effective element size to each of the discretizations performed in the following study, but for complex geometries and different meshing parameters (i.e., global seed size and edge seed size) several approaches can be used. In this case, the average characteristic length in each of the models was computed using $h = \frac{1}{N^{1/n}}$ [28], where h is the characteristic length of the elements, N is the number of elements in the model and n refers to the number of dimensions considered in the model (i.e., two dimensions in this case). In particular, for this study, the number of elements is not evenly distributed, since the parameter used to control the mesh is only the edge seed size, it is expected that the error should decrease faster when the average characteristic length decreases due to this fact.

Therefore, the distribution of the stress at the voids tip versus the average size of the elements pairs was expected to have a good fitting with a quadratic equation, allowing us to extrapolate the value of the stress at the tip when the average size of the elements is close to zero, which can be considered as the most exact result possible. Finally, the error for each of the average size of element executed in the simulations was computed as

$$e = \frac{\phi_x - \phi_0}{\phi_0} \tag{11}$$

where e represents the estimated error for the current simulation, ϕ_x represents the field variable at the comparison location for the current simulation (i.e., the normal stress in the vertical direction) and ϕ_0 the extrapolated field variable at the comparison location, if infinitesimally small elements were used (h≈0), this allowed us to choose the error level that was acceptable for the comparison required in this investigation.

However, it is relevant for some of the shapes in use to understand that the error estimation can only be considered good if the conditions for a linear and continuous response of the model are sustained. In the case of interest, the elongated voids that are being analyzed can be compared to a crack in the behavior of their stress field, meaning that the stress presents a singularity when the position approaches the void tips. This singularity breaks the continuity of the stress field near the crack tip and the assumption of the error behavior previously explained does not apply. Some of the stop-hole shapes to be added to the voids shapes are different enough from a crack shape to overcome this problem, other of the selected shapes present this singularity behavior. This is shown afterwards through the fitting that the quadratic assumption presents in the plots stress at the voids tip versus the average size of the elements. For those cases that did not show a good fitting under the quadratic assumption other polynomial degrees for the fitting where tried to get a better fitting and be able to at least estimate the error in a comparable way to the process explained before.

This procedure of computing the error in the output variable was performed as a revision of the mesh parameters in two of the 10 selected shapes. The main criterion for choosing the two shapes was to choose in at least one case a promising stress reduction and one in which the stress reduction was not prominent, avoiding those that obviously caused singularities in the model. Further information about how the revision was planned and executed using MATLAB code can be found at the Section 2.2.3.

2.2.3 Software tools: MATLAB script: mesh study

A MATLAB script was developed to perform the mesh revision explained in Section 2.2.2 to automate the process of executing simulations of the multiple mesh parameters required and the different models selected. As mentioned in Section 2.2.1, a MATLAB script can interact with the python script through OS commands and force ABAQUS to run it, then the results may be interrogated by the MATLAB script to generate plots, manage data, and perform operations, among other tasks; in this way, the data generated by the simulations was treated for its analysis.

In general, the MATLAB script alters the edge seed size in the general python model script (that already applies this parameter only to the edges generated using the superformula), while keeping the superformula parameters as required for each of the shapes selected. Then, it executed the model in ABAQUS/Standard and waited for the simulation to be completed. After that, it compiles the relevant information, such as the displacement and stress reports. Finally, the MATLAB script performs some required operations, such as finding the average size of the elements, computing the effective Poisson’s ratio, and finding the highest stress near the void tips, and it saves the results for further analysis of the data.

2.2.4 Shape comparison through simulations

After selecting the set of 10 parametric shapes, a first batch of RVE simulations was executed. The 10 shapes were compared at fixed porosities (10%, 12% and 14%) to understand the different behaviors these geometries may have. The intention of fixing the porosity of the models and performing simulations for several porosities was to verify the effect that the scale of the added stop-hole geometry may have over the behavior of the RVE. These conditions represented a total of 30 simulations.

The purpose of these first set of simulations was to start looking at the stress concentration that the pattern of voids may present under a standard external loading and how the different selected shapes may affect the displacement fields. Based on this analysis, the field of search was reduced to 6 shapes, discarding the 4 less promising shapes. To do this, a set of contour plots of the stress, combined with the magnitude in the direction of the applied average strain, were compared to discriminate among the batch of shapes and establish criteria to define which ones were more promising. In this case, the criteria were to have the lowest magnitude of the normal stress in the vertical direction near the void tip, and, for those geometries that were close in stress magnitude, the level of stress concentration that could be recognized in the contour plots. Further information about the MATLAB code used to perform the batch of simulations mentioned can be found in Section 2.2.5.

2.2.5 Software tools: MATLAB script: Batch simulation

After reviewing the mesh discretization for a few geometries, the next step was to start comparing the results that can be achieved in terms of maximum stress reduction at the voids tips and changes in the stress distributions. To do this, after choosing 10 unique shapes from the wide variety that the superformula can offer, a set of simulations of the RVE models incorporating these shapes as stop-hole geometries was prepared. However, arbitrarily choosing a scale for the stop-holes was considered not appropriate given the interest in minimizing the effect on the geometries in this first preliminary set of simulations. Therefore, the scale was associated with a porosity that the RVE achieved when the stop-hole was added to the voids, and 3 porosities were chosen to give a good first view of the stress reduction and stress distribution results.

To execute the batch of 30 simulations that this combination represented, a MATLAB script was developed that used the superformula parameters put in an array previously prepared of each of the 10 stop-hole shapes and the scale corresponding to that shape and one of the porosities chosen. These results were evaluated in the interface function between MATLAB and python-ABAQUS as presented in the Section 2.2.1. As the python general model already was capable of performing the required file management and file saving and exporting, when each of the simulations was executed from the python script, the script saved the data that was processed. It analyzed the afterwards, through stress contour plots, the maximum stress caused by the complete geometry, the effective Poisson's ratio, and displacement contour plots.

2.2.6 Total search over the scale using simulations (6 shapes)

Finally, during this first exploration of the shapes that the superformula yields, and having chosen 6 promising stop-hole geometries using the previous simulations, the next step was to determine which one of the remaining shapes represent a better improvement for the stress concentration point of view independently from the porosity. This was achieved by running RVE simulations for each of the 6 shapes selected, but in this case in steps along a range of porosities. The purpose was to generate graphs that will show the behavior of each of the void shapes with the added stop-hole geometry in terms of its maximum magnitude of the normal stress in the vertical direction and Poisson's ratio against the porosity, keeping in mind that the porosity of the RVE is mainly controlled by the size of the added geometry. These graphs showed the limitations of the improvement that can be achieved with each of the added stop-holes, while also reviewing the change on the auxetic behavior that the stop-holes might be causing when added to the pattern. More information about how this study was performed can be found in the Section 2.2.7.

2.2.7 Software tools: MATLAB script: Total search of scale for a batch of geometries.

After the results obtained with the execution of the codes mentioned in the Section above, a more in-depth review of the behavior caused by the addition of these shapes was required. The intention was to be able to understand the dependency of variables, such as the maximum stress in the model in the direction of the applied average strain and the Poisson ratio, had with respect to the porosity that the scale of the stop-hole feature caused. To get that understanding several more simulations were required.

At the beginning, a set of 20 equally distributed scales in between a range was executed. However, given the non-linear relationship between the scale and the porosity, the variable that was desired as a control variable, more simulations were required to generate enough resulting points to fill the graph up to a desired porosity value. For most superformula stop-hole shapes, around 9% porosity was the minimum reachable due to the geometric constraints previously established, and for most shapes in the preliminary simulations around 25% porosity was identified as having a more stable behavior, reducing the capacity of the shape to modify the behavior. Thus, the range of porosities desired was set at 10% to 25%,

To perform the previously explained batch of simulations, and similar to the case presented in previous sections, a MATLAB script was developed to execute the task. Before the execution, an array with the superformula parameters and the boundaries of the scale range for each of the superformula stop-hole shapes was prepared. In the beginning, the analysis was performed for the total 10 shapes, but then it was reduced to only 6 of them. This represented around 300 simulations. For, each of the parameter sets the simulations were performed by using the MATLAB function presented at Section 2.2.1 to execute the corresponding models in ABAQUS and then process the data generated into plots of the maximum stress and Poisson’s ratio. Finally, the data was processed in Microsoft Excel to present it in condensed graphs.

III. Results and Discussion

3.1 Mesh study

As described in Sections 2.2.2 and 2.2.3, a mesh study to determine optimal mesh parameters was performed. In this case, the normal stress in the vertical direction, in one specific location, the tip of the voids (see Figure 6), was used as the comparison variable to obtain an extrapolated estimate of it when the mesh is absolutely refined and to use that value to define an error estimate for each mesh parameter combination. The mesh study was performed for 2 of the first geometries of interest, geometries B and C (see Figure 6), at 2 of the lowest porosities to be studied in this investigation (10% and 12%).

As explained in the Sections 2.2.3, multiple combinations of one general seed size and increasingly smaller edge seed sizes were simulated, obtaining different element quantities, Normal stresses in the vertical direction at the tip of the voids, average Poisson's ratio (v) and times of execution. Using the element number, an average element size was computed (h) and then squared (h^2). Finally, in each case, a plot of the stress as a function of the average size squared was presented, and with it a linear regression that allowed the stress to be extrapolated when the value of h is zero, which hypothetically can be considered the most accurate value of stress at that location. Against that hypothetical value of stress, the error percentage of each seeding combination was computed by using (11).

3.1.1 Geometry B – 10% porosity

In this case, the extrapolated stress for a mesh with infinitesimally small elements (h≈0) was of -2954.7 MPa (See Figure 10). Also, the correlation coefficient of the linear regression between h^2 and S22 was of R^2=0.9792, this shows that for this geometry there is a good linear correlation between the variables used, something that can be expected from a stress field with no singularities. Also, most of the seeding combinations used were below 4% error (See Table 1), which can be considered as a good metric of accuracy for the discretization model.

Table 1 Mesh study results for geometry B at 10% porosity

Edge Seed (mm)	Element Number (#)	Normal stress 22 at tip (MPa)	Poison's ratio - ν (adim)	Process Time (s)	Average element size - h (mm)	Squared average element size - h^2 (mm^2)	Error (%)
0.14	18970	-2860.74	-0.2666	63.55	0.1195	0.01428	3.18%
0.13	19880	-2860.65	-0.2666	62.62	0.1167	0.01363	3.18%
0.12	20988	-2863.22	-0.26658	66.26	0.1136	0.01291	3.10%
0.11	22052	-2873.06	-0.26658	68.64	0.1108	0.01229	2.76%
0.1	22624	-2880.16	-0.26658	69.57	0.1094	0.01198	2.52%
0.09	24996	-2884.27	-0.26658	74.64	0.1041	0.01084	2.38%
0.08	28884	-2897.78	-0.26658	80.60	0.0969	0.00938	1.93%
0.07	32430	-2903.29	-0.26658	87.05	0.0914	0.00836	1.74%
0.06	38618	-2908.58	-0.26658	97.30	0.0838	0.00702	1.56%
0.05	45094	-2914.41	-0.26658	111.10	0.0775	0.00601	1.36%
0.04	55348	-2920.93	-0.26658	133.34	0.0700	0.00490	1.14%
0.03	72282	-2926.49	-0.26658	174.18	0.0612	0.00375	0.95%

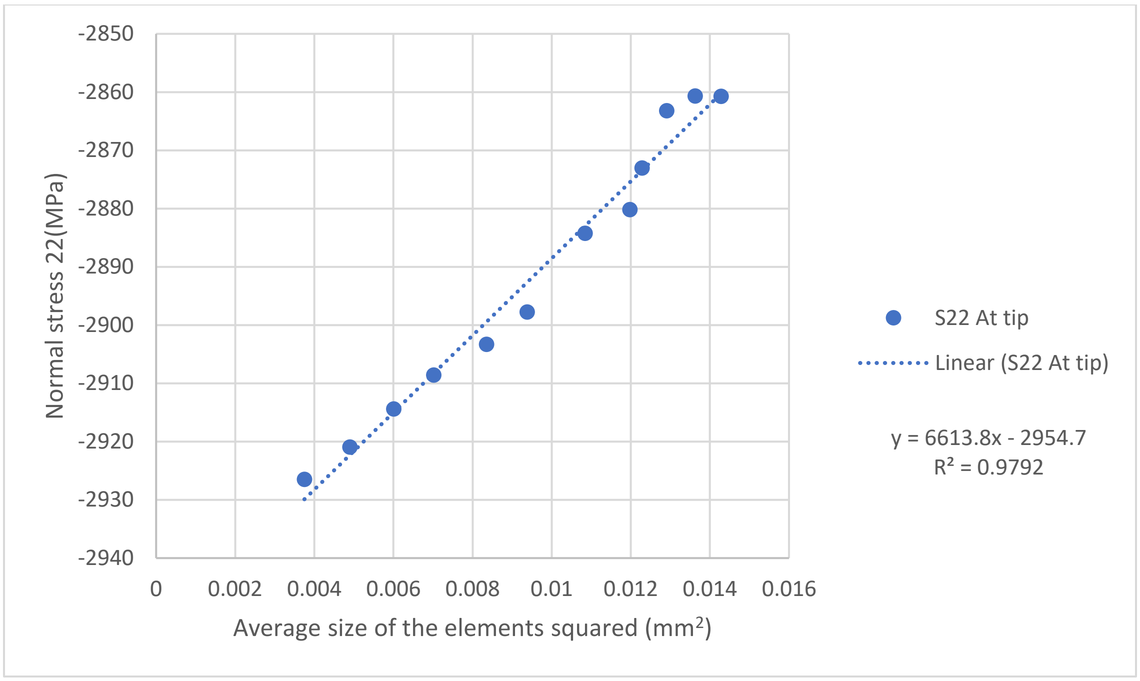

Figure 10 Normal stress vs Average size of the elements squared for geometry B at 10% porosity

3.1.2 Geometry B – 12% porosity

Comparing these results with the previous ones now the extrapolated stress was of -2476.5 MPa and the correlation coefficient of 0.9906 (see Figure 11). The results obtained can be considered as expected, since the porosity was increased, meaning that the feature added at the tip of the void increased in size, therefor the stress concentration was diminished. Here the linear correlation between the average size of the elements squared and the stress was even better, and the errors were smaller too. It suggests that for geometries that effectively reduce the stress concentration when the porosity increases, and with comparable mesh parameters, the accuracy of the simulation increases (see Table 2).

Table 2 Mesh study results for geometry B at 12% porosity

Edge Seed (mm)	Element Number (#)	Vertical Normal stress at tip (MPa)	Poison's ratio - v (adim)	Process Time (s)	Average element size - h (mm)	Squared average element size - h^2 (mm2)	Error (%)
0.14	20444	-2426.93	-0.2653	63.10	0.1151	0.01325	2.00%
0.13	21882	-2431.14	-0.26528	58.94	0.1113	0.01238	1.83%
0.12	23826	-2431.45	-0.2653	59.87	0.1066	0.01137	1.82%
0.11	25696	-2438	-0.26528	60.46	0.1027	0.01055	1.55%
0.1	28344	-2443.03	-0.26528	61.17	0.0978	0.00956	1.35%
0.09	33222	-2447.35	-0.26528	65.16	0.0903	0.00816	1.18%
0.08	38004	-2448.45	-0.26528	67.80	0.0844	0.00713	1.13%
0.07	43836	-2453.43	-0.26528	72.65	0.0786	0.00618	0.93%
0.06	52356	-2456.93	-0.26528	78.65	0.0719	0.00518	0.79%
0.05	65654	-2459.83	-0.26528	89.06	0.0642	0.00413	0.67%
0.04	84564	-2464.67	-0.26528	98.24	0.0566	0.00320	0.48%
0.03	117660	-2468.91	-0.26528	118.28	0.0480	0.00230	0.31%

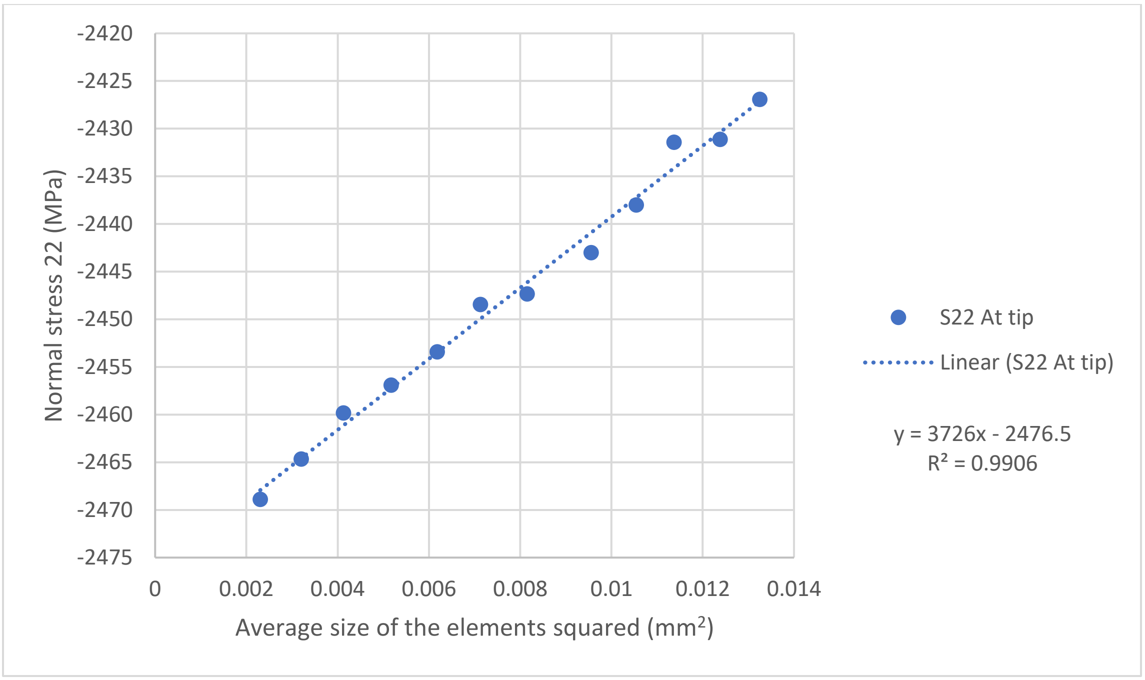

Figure 11 Normal stress vs Average size of the elements squared for geometry B at 12% porosity

3.1.3 Geometry C - 10% porosity

At geometry C and a lower porosity, the extrapolated stress and linear correlation coefficient were -6677.6 MPa and 0.9397 (see Figure 12). This correlation coefficient is much lower than the ones found previously for geometry B. That effect can be attributed to having a sharper curve at the tip of the void, getting closer to crack conditions, which may affect the stress field about the void tip causing singularities. However, this correlation coefficient can be considered acceptable. On the other hand, the stress is increasing in magnitude rapidly when the average size of the elements decreases, causing a high magnitude extrapolated stress. At lower element counts the error computed can be considered high. To keep the error below 5%, the edge seed size must be lower than 0.07 (see Table 3).

Table 3 Mesh study results for geometry C at 10% porosity

Edge Seed (mm)	Element Number (#)	Vertical Normal stress at tip (MPa)	Poison's ratio - v (adim)	Process Time (s)	Average element size - h (mm)	Squared average element size - h^2 (mm2)	Error (%)
0.14	20552	-6004.84	-0.20392	72.10	0.1161	0.01348	10.08%
0.13	22294	-6128.54	-0.20392	71.27	0.1115	0.01243	8.22%
0.12	24390	-6028.51	-0.20398	72.67	0.1066	0.01136	9.72%
0.11	26572	-6099.11	-0.20398	76.96	0.1021	0.01043	8.66%
0.1	29420	-6171.24	-0.20398	82.68	0.0971	0.00942	7.58%
0.09	35246	-6286.29	-0.204	87.58	0.0887	0.00786	5.86%
0.08	40810	-6338.4	-0.20402	91.18	0.0824	0.00679	5.08%
0.07	47928	-6312.73	-0.20402	102.31	0.0760	0.00578	5.46%
0.06	57464	-6454.63	-0.20402	117.90	0.0694	0.00482	3.34%
0.05	71570	-6494.95	-0.20402	129.96	0.0622	0.00387	2.74%
0.04	91836	-6550.41	-0.20402	154.89	0.0549	0.00302	1.91%

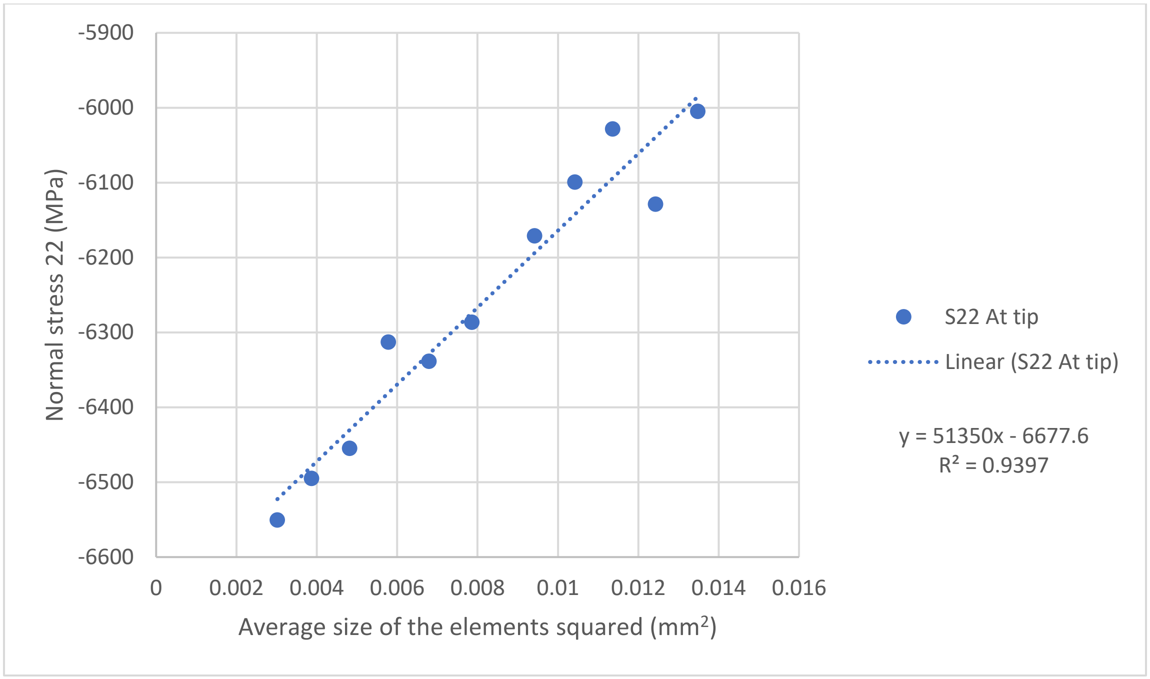

Figure 12 Normal stress vs Average size of the elements squared for geometry C at 10% porosity

3.1.4 Geometry C - 12% porosity

Again, for a high porosity, even for a sharper void geometry, the magnitude of the extrapolated stress decreased, and the linear correlation coefficient improved. This is probably caused by the larger curvature radius presented by the geometry at high scales (meaning higher porosities). For the conditions presented, to maintain an error percentage below 5% it is enough to keep the edge seed size below 0.1 (see Table 4). Another point is that for geometry C when the porosity level was increased, the execution times also increased, meaning a higher computation load.

Table 4 Mesh study results for geometry C at 12% porosity

Edge Seed (mm)	Element Number (#)	Vertical Normal stress at tip (MPa)	Poison's ratio - v (adim)	Process Time (s)	Average element size - h (mm)	Squared average element size - h^2 (mm2)	Error (%)
0.14	23430	-5407.98	-0.19771	72.83	0.1075	0.01157	9.09%
0.13	25372	-5538.14	-0.19771	75.38	0.1033	0.01068	6.91%
0.12	27134	-5562.38	-0.1977	79.25	0.0999	0.00999	6.50%
0.11	31416	-5594.49	-0.19772	85.26	0.0929	0.00863	5.96%
0.1	36030	-5632.68	-0.19772	88.47	0.0867	0.00752	5.32%
0.09	42456	-5664.24	-0.19774	99.02	0.0799	0.00638	4.79%
0.08	49694	-5728.89	-0.19775	109.45	0.0738	0.00545	3.70%
0.07	59364	-5800	-0.19775	118.37	0.0676	0.00456	2.50%
0.06	71916	-5813.12	-0.19775	140.00	0.0614	0.00377	2.28%
0.05	91318	-5812.1	-0.19775	164.74	0.0545	0.00297	2.30%
0.04	120976	-5819.2	-0.19775	200.93	0.0473	0.00224	2.18%

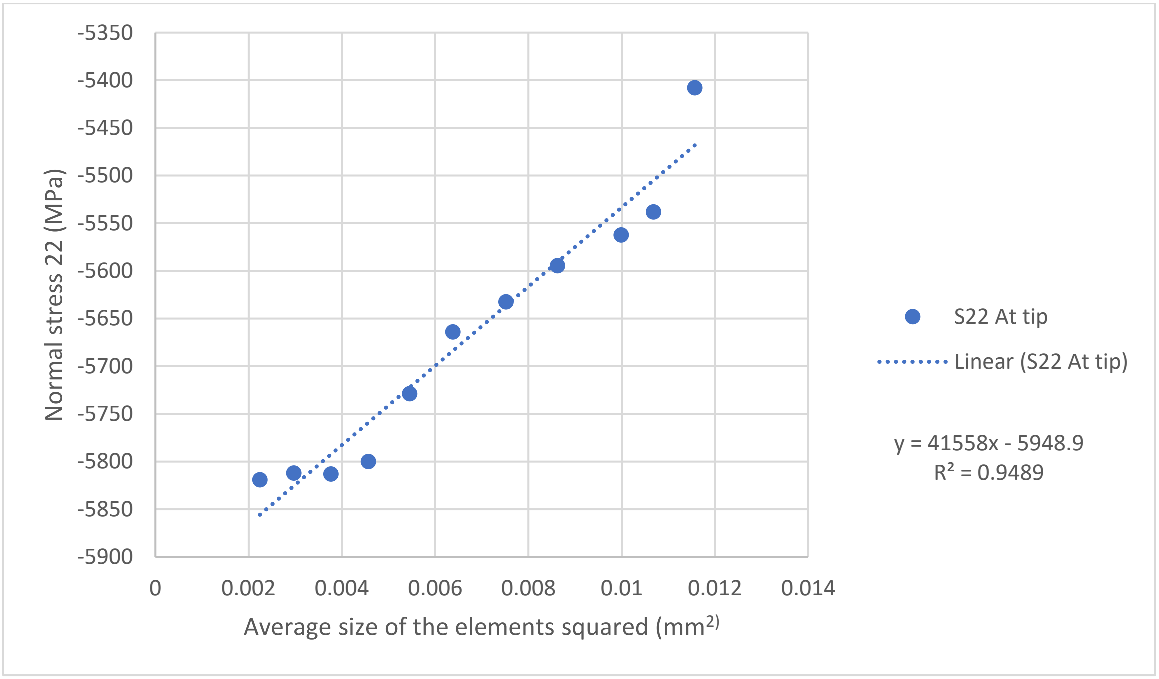

Figure 13 Normal stress vs Average size of the elements squared for geometry C at 12% porosity

In general, the mesh study suggests that to keep the accuracy of the results better than the 5% error, the edge seed size should be less than 0.06 for most geometries, even the ones that show sharper curvature radius at the tips. Also, it can be considered that, at higher porosities, larger edge seed sizes can be used without substantially affecting the accuracy of the results to be obtained. This can be useful to reduce the computational load that this may represent.

3.2 Shape comparison through simulations (10 Shapes)

As discussed in Section 2.2.4, after carefully reviewing the variety of geometries that can be parametrized using the superformula, a total of 10 shapes were chosen to be evaluated. This evaluation consisted of the simulation of an RVE using each of the selected shapes as stop-holes geometries. That was reproduced for at least three different low porosity levels, due to being close to the lowest possible porosity that can be achieved for the geometrical conditions enforced for the voids in Figure 4.

3.2.1 10 % Porosity

From Figure 14, the first observation that can be made is that stress concentration is apparent in all the selected shapes since in all the cases the stress level only gets higher in the neighborhood of the void tip. Also, looking at these isolated results, we identify that a change in the stress distribution is generated by each of the geometries, since even while keeping a constant porosity among them, the stress patterns are remarkably different. Another important observation that can be made from Figure 14 is that some of the selected geometries present a better stress distribution, meaning an improvement in the stress concentration; however, this is going to be more evident in further simulation results.

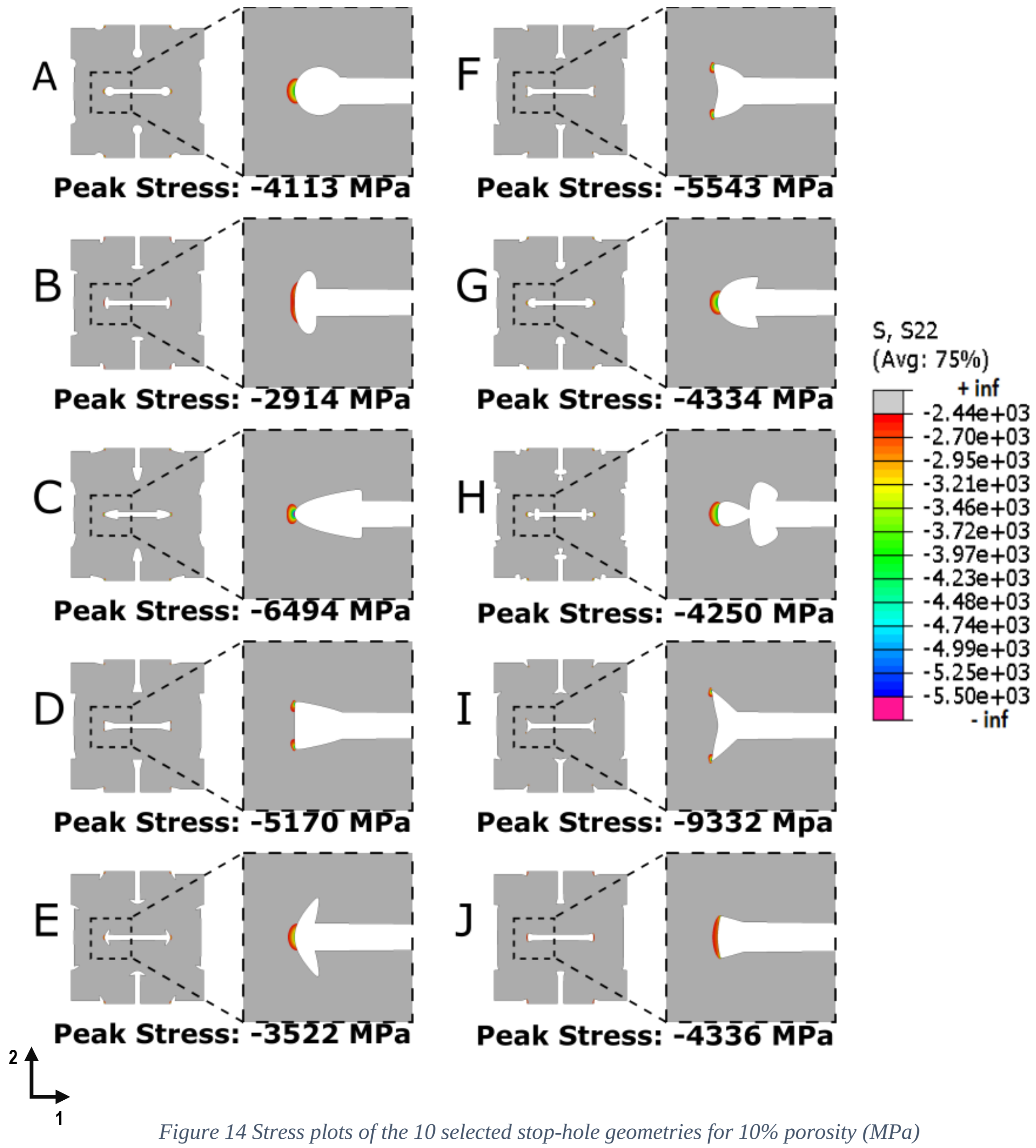

Figure 14 Stress plots of the 10 selected stop-hole geometries for 10% porosity (MPa)

Considering that the average normal strain was applied in the vertical direction, a high compressive displacement response can be appreciated in the vertical direction (see Figure 15). More interestingly, in the opposite direction the displacement response also presents an average compressive response. This shows that the RVE has an effective negative

Poisson's ratio, which represents an auxetic response. At this level that fact is enough for considering the geometries effective for the objective of this investigation.

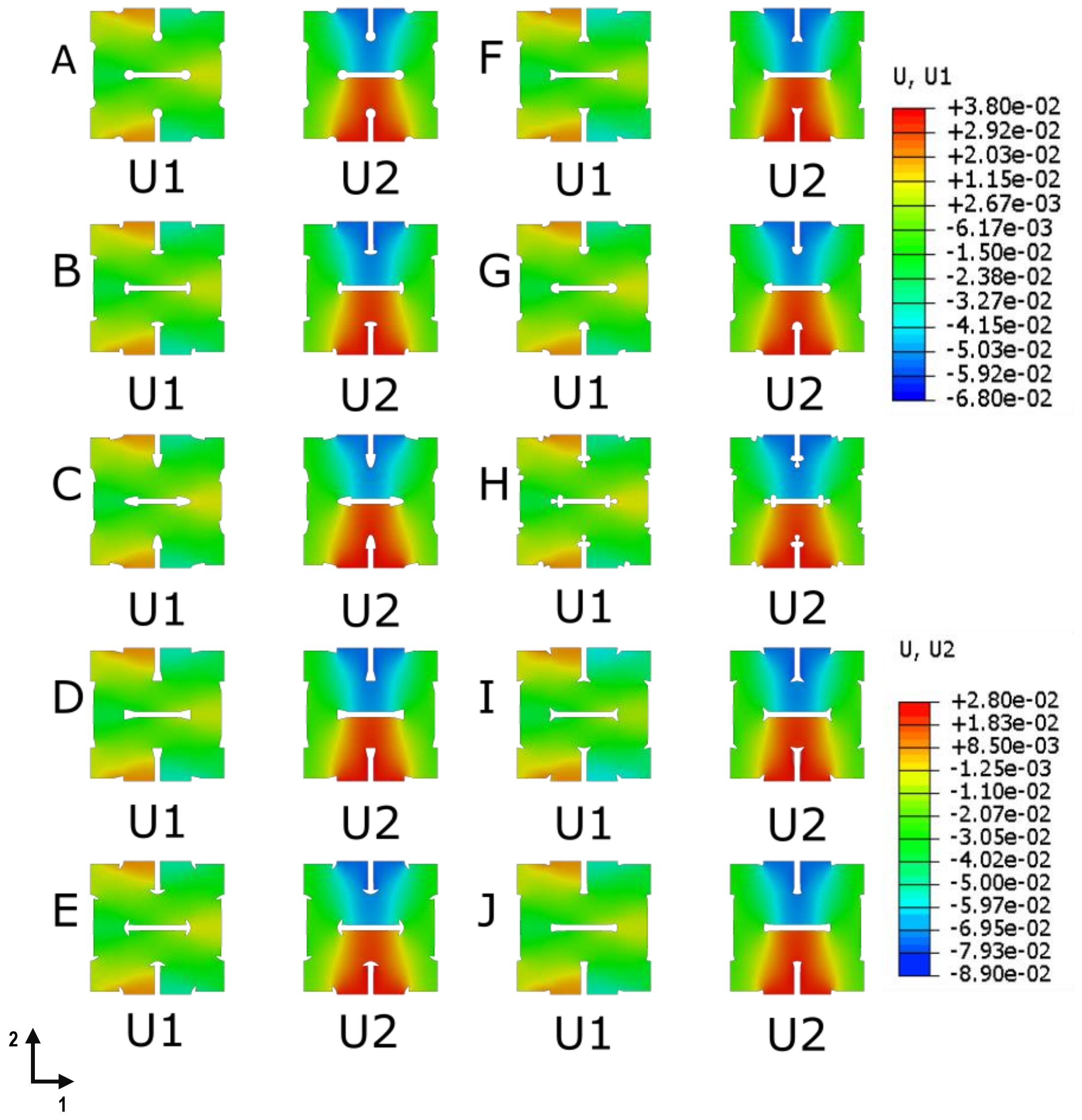

Figure 15 Displacement plots in the directions 1 and 2 for each of the 10 selected geometries (mm)

3.2.2 12% Porosity

Comparing this set of results for the stress in the vertical direction with the results presented in the previous Section the stress reduction can already be seen for all geometries, this means that for all 10 selected geometries the magnitude of the stress shown in response to the applied average strain is less intense. Another positive observation is that in this case there is already one geometry showing no stresses over the value of -2440 MPa, geometry B, meaning that this geometry already shows an advantage over the others in the stress reduction sense. However, the stress reduction and the concentration improvements are not so evident for some of the geometries such as C, F, I and J.

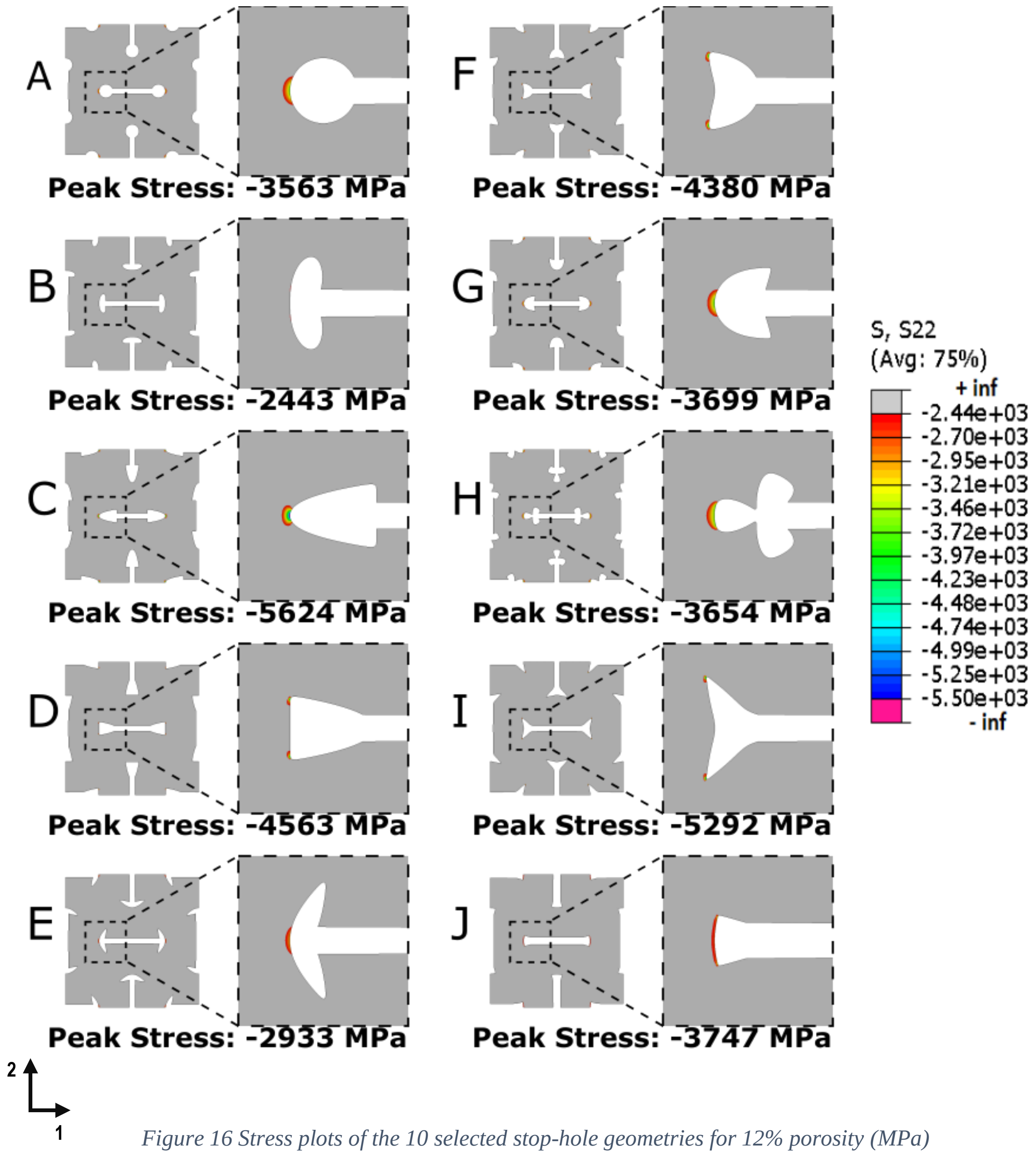

Figure 16 Stress plots of the 10 selected stop-hole geometries for 12% porosity (MPa)

3.2.3 14% Porosity

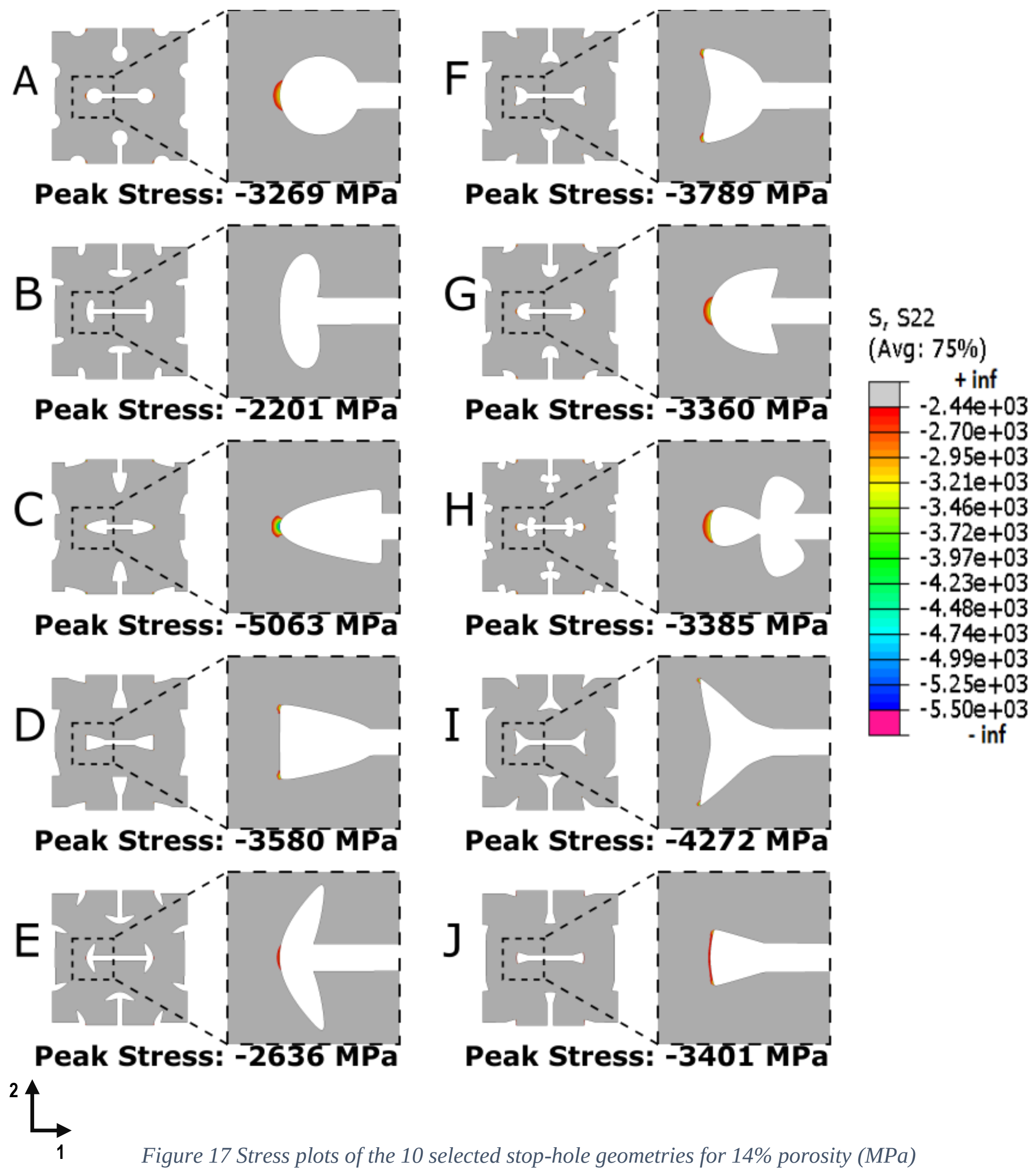

Figure 17 Stress plots of the 10 selected stop-hole geometries for 14% porosity (MPa)

Again, when the porosity was increased minimum stress in most of the models decreased in magnitude, this demonstrates that the scale, that is the curvature radius, affects the stress distribution in a positive manner. However, the stress concentration was not effectively reduced in all cases, at least from the visual results obtained.

At this point of the analysis, it is apparent that some shapes perform better than others even when the size of the stop-hole is increased and, therefore, the porosity of the voids. Some particularly interesting observations about the geometries that exhibit the best results regarding stress reduction are, for example, that the peak stress is strongly related to the curvature of the shape at the mid axis of the void, this observation is similar to what can be deduced from the idea of stress concentration (see Figure 17, shapes B and E). Also, it can be observed that the shapes performing well are significantly better at utilizing the provided area of the hole to accommodate large curvature radius in this orientation (see Figure 17, shapes B, E and J), meanwhile less effective shapes in this regard perform significantly worse (see Figure 17, shapes G and H). However, incorrect selection of the shapes can lead to conditions in which, for example, the maximum stress moves from the mid axis of the void, generating in most cases a higher stress concentration (see Figure 17, shapes D, F and I). This general observation of how geometrical properties of the shapes can lead to different peak stresses and stress distributions can be of further use for selecting adequate shapes for specific objectives.

Further studies were executed to better understand how the scale of the features can affect the behavior of the material. However, these more extensive studies were performed only for a few of the more promising geometries. The geometries that presented less stress reduction or more concentered stress distributions were discarded from the next set of analyses, this means that the geometries C, F, I and J were not considered in further analysis.

3.3 Total search over the scale using simulations (6 shapes)

After performing the analysis presented in Section 3.2, some of the geometries were discarded due to poor results in stress reduction and stress concentration. However, this preliminary study did not show in detail how the scale can affect the resulting behavior of the material for these conditions. Therefore, several more simulations were performed for each of the 6 remaining geometries of interest (A, B, D, E, G and H). Their condensed results are presented in this Section, along the results of a similarly sized RVE model simulated with elliptical voids at several different porosities for comparison. The output variables of interest in this case were the average Poisson's ratio and the minimum normal stress in the vertical direction present in the model for the applied conditions. The average Poisson's ratio was important for showing how different porosities were affecting the auxetic behavior obtained, meanwhile the minimum normal stress in the vertical direction shows how the geometries improve the stress distribution.

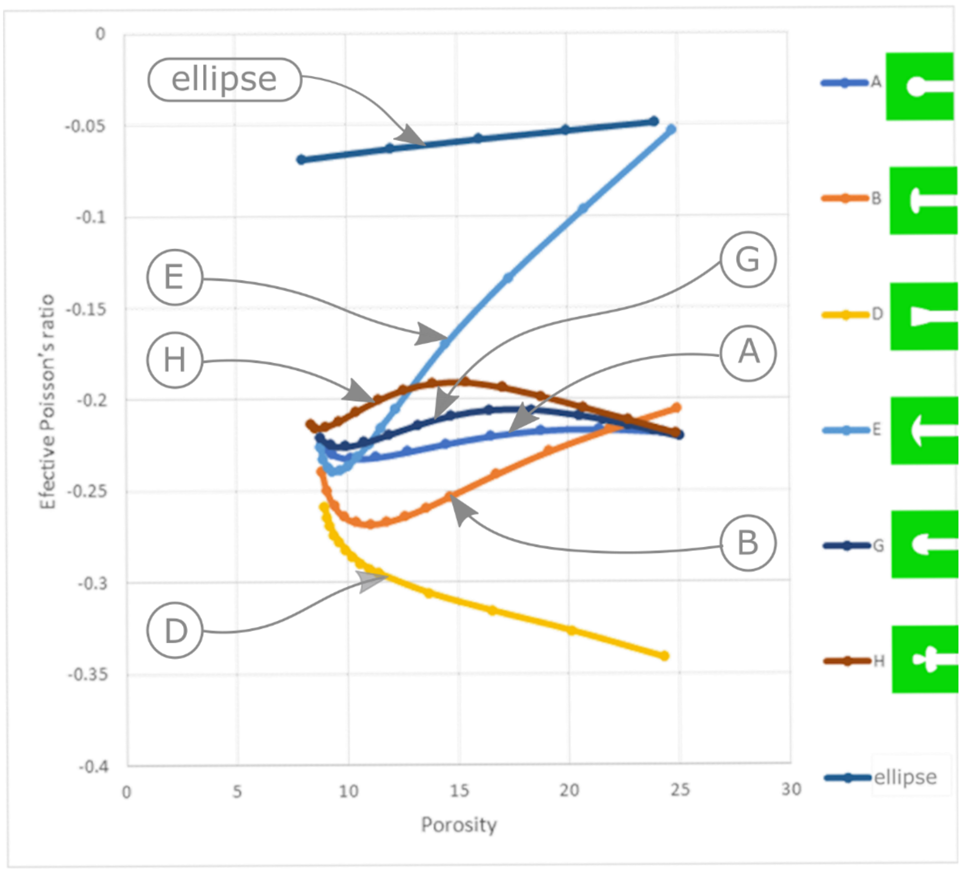

Figure 18 Average Poisson's ratio vs porosity for the 6 selected geometry and the ellipses for comparison.

In Figure 18, the Poisson's ratio is plotted against the porosity of each of the models corresponding to the different geometries that are being studied. Several observations can be made from these results, some of them are pointed out in the following paragraphs.

At lower porosities, the average Poisson's ratio shows a similar value for all the geometries. This is caused by the similarity of all the geometries at low porosities (low scale factors for the stop-holes) to a simple slot void pattern. Also, for all the cases the effective Poisson's ratio is well below the results obtained for the similar sized models with elliptical voids.

Inside of a range of porosities, particular for each geometry, the average Poisson's ratio seams to become more negative. This means that the addition of that geometry at that scale as a stop-hole is contributing to the auxetic behavior of the material. However, the behavior of the average Poisson's ratio against the porosity shows an inflection point for most geometries where it starts an increasing tendency. This happens inside the range of porosities studied for all geometries except for D.

After the inflection point, two different kinds of behaviors were noticed. The first behavior (geometries B and E) is a tendency to increase with no asymptotic behavior found, at least for the porosity range studied. The second behavior (geometries H, G and A) is some fluctuating behavior around a constant value, this was noticed also for the porosity range studied.

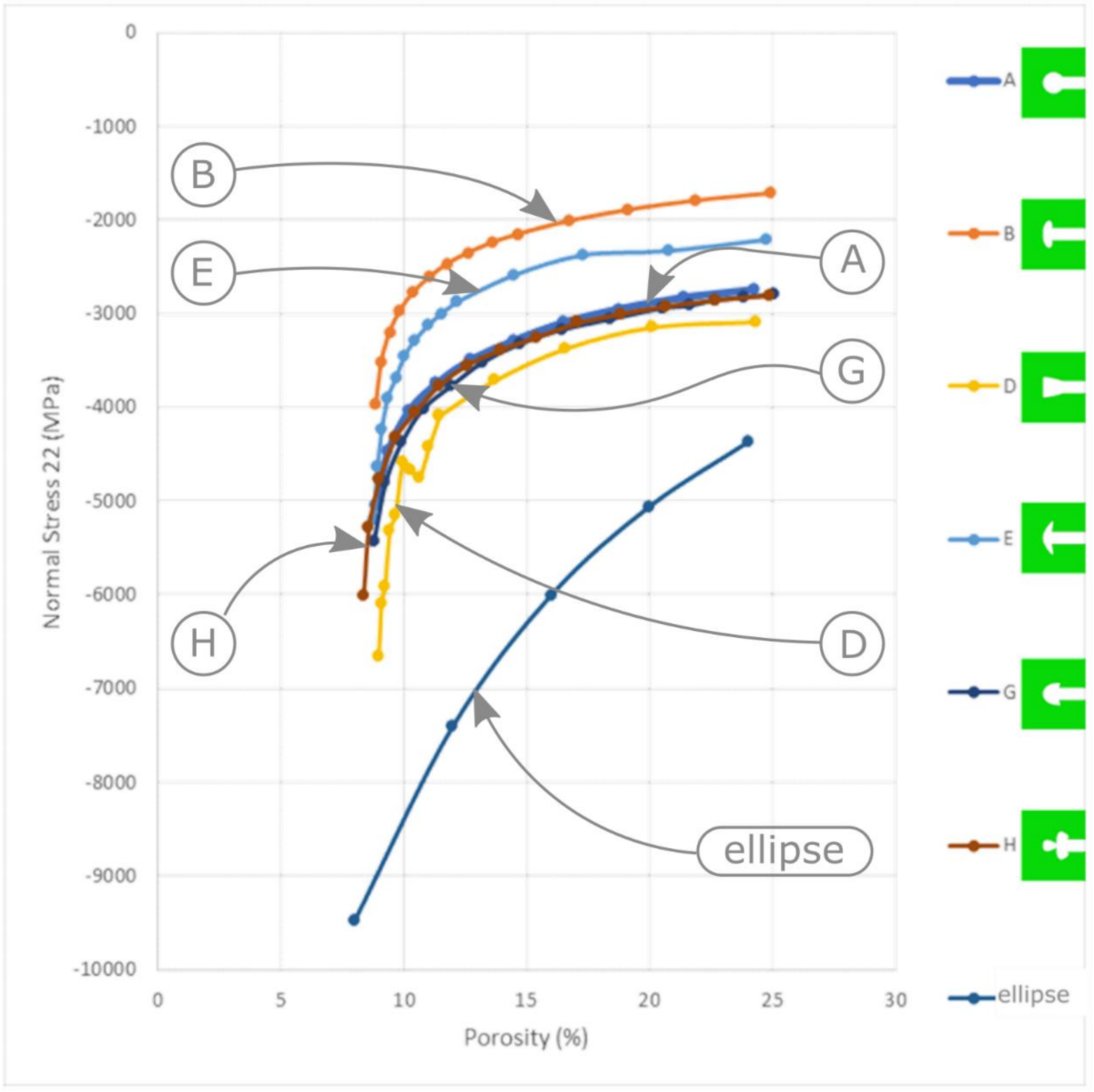

Figure 19 Normal stress in the vertical direction vs porosity for the 6 selected geometry and the ellipsis for comparison.

In Figure 19, are presented the minimum normal stress in the vertical direction against the porosity that each model presents for the corresponding scale of the stop-hole geometry. Several important observations can be obtained from these results:

All stop-hole geometries presented similar behaviors when the scale of the stop-holes and, therefore, the porosity, was increased. For all the stop-hole geometries the magnitude of the minimum normal stress decreases when the porosity is increased. Also, all the geometries present an asymptotic behavior when they approach a specific stress value, particular for each of the geometries. Also, it can be noticed that all the geometries presented a better reduction in the minimum normal stress in the vertical direction than simply increasing the porosity of the elliptical voids in a similarly sized model.

On one hand, geometries A, D, G and H present very similar stress reduction behaviors. Notice that the geometry A was the typical circular stop-hole geometry; this geometry has been previously analyzed for slowing down the crack propagation with acceptable results [24]. On the other hand, geometries B and E present a slight advantage over the other geometries, meaning a better reduction of the minimum normal stress in the vertical direction, uniformly over the range of porosities studied.

IV. Summary & Conclusions

In the present investigation, the objective was to find a structured way to reduce the stresses in the neighborhood of the tips of the elongated voids required in the patterns that induce effective auxetic behavior in linear materials. To achieve this objective, the idea of the stop-hole was used since it has proven its effectiveness to reduce stress concentration and crack propagation for crack-like geometries. However, to reduce the stress while keeping the porosity low and maintaining or increasing the auxetic behavior of the structure, it was necessary find a more effective shape of stop-hole than a circle. We used a generalized ellipse known as the "superformula", a parametric equation capable of yielding an almost infinite variety of rounded shapes controlled by 6 parameters, allowing the parametrization of the geometry. Given the broad variety of shapes that the superformula may yield, a first set of parameters was selected due to the uniqueness of the shapes they yielded and, from that initial set, a smaller group of shapes was selected to be evaluated as potential stop-hole geometries for the purpose of the investigation.

Once a parametric model of the geometry was accessible it was necessary to evaluate the stress concentration and the effective Poisson ratio of the structure generated by the new void geometry. To do this, a finite element model was prepared using ABAQUS standard and its built-in python scripting capabilities. The model was a Representative Volume Element representation, under the assumption of periodic boundary conditions subjected to an equivalent average compressive strain in the vertical direction equal for each of the geometries evaluated. The mesh and mesh parameter applied uniformly to each of the modeled geometries were validated via an estimation of the error in the stress located at the tip. This error was calculated by comparing the stress obtained by running the simulation at specific meshing parameters with an ideal stress extrapolated from multiple simulations and the idea of the error being proportional to h^2 (average size of the elements squared). This validation was executed on the model for a set of the evaluated shapes, some of which presented the sharpest changes in geometry; after that, the assumption was made that under similar meshing parameters the error was going to be similar or lower in the rest of the more rounded shapes to be evaluated. This way, a final python script to define the parametric geometry, material, mesh parameters, element selection and periodic boundary conditions was set up in order to evaluate each of the selected shapes.

The next step was to evaluate each of the selected shapes of the superformula using the developed simulation model. At this level of the investigation, it was possible to run FEA simulations of multiple shapes, particularly the set of 10 shapes selected in previous steps of the investigation. All evaluations at this step were performed with the stop-hole shapes scaled to achieve an specific porosity of the voids. The analysis was focused on evaluating the contour plots of the displacements in the horizontal and vertical directions (U1, U2) to make sure of the auxetic response of the structure, and evaluating the contour plot and maximum magnitude of normal compressive stress in the vertical direction (S22) to determine which geometry performs the best regarding stress reduction. From this evaluation, a smaller group of promising shapes was selected for a broader analysis.

Even though the results regarding stress reduction were promising for the final selected set of shapes, the results are still incomplete since the analysis was performed only for specific porosity values for the voids. To ensure that the final selected shape performs with better effectiveness, regarding stress reduction, than any of its counterparts the analysis should show that this is true for any porosity. At this point, an automated MATLAB-python script was implemented to perform FEA analysis of the reduced set of shapes for a significant range of porosities recording their Poisson's ratio and maximum compressive stress in the vertical direction. Using this data, a set of two plots were generated to compare the performance of the shapes at multiple porosities within this range and exposing their behavior.

From the results of this process, it can be concluded that adding stop-holes at the ends of the elongated voids used to induce auxetic behavior in meta materials is an effective method to reduce the stress concentration. Also, that the stress reduction generated by circular stop-holes can be improved by using a different parametric shape (compare A and B in Figure 19) and that the improvement was maintained along the range of porosities evaluated. All this can be done while maintaining or improving the auxetic behavior obtained (compare A and B in Figure 18) for the range of porosities evaluated.

As a corollary of the evaluation of maximum compressive stress and effective Poisson's ratio, a few more things can be concluded about the behavior of these structures. The effectiveness of the stop-hole geometry converges for any geometry when the porosity approaches the limit value of the porosity of the elongated void, this can be explained due the fact that the size of the stop-hole is not comparable any more to the size of the stop-hole. Also, the stress reduction increases rapidly when the porosity is initially increased from that critical porosity of an elongated void with no stop-hole; however, it reaches an asymptotic value of stress reduction as the porosity increases. Regarding the behavior of the Poisson’s ratio as the porosity increases, its evaluation infers that different behaviors can be achieved, in some cases the auxeticity of the structure kept increasing along the porosity range evaluated, while in some other cases it presented an inflexion point where it started increasing in auxeticity up to a critical point.

Other relevant observations showed interesting relationships between simple geometrical properties and the behavior of the structure (i.e., peak stress and stress distribution). Particularly, one relevant observation was that high performing shapes, regarding stress reduction, are usually capable of accommodating large curvature radii at the mid axis of the void for a given area. This observation can lead to a more effective shape selection in future work, since it points to promising shapes even before performing any FEA analysis, allowing for a more effective shape selection.

In general, the investigation was successful in finding improved geometries to induce auxeticity in linear materials using low porosity patterns of voids, regarding their stress concentration while keeping or improving their auxetic behavior.

However, there is significant work to be done regarding finding the best possible shape to induce auxeticity in linear elastic materials. One suggested area of interest is performing numerical optimization, meaning the use of search algorithms to numerically find optimal shapes with the objective of reducing the stress in the material while maintaining or improving their auxetic behavior. Also, mentioned in this investigation is the fact that the decrease in stress concentration can improve the fatigue life of these metamaterials; however, this has not been proven yet. Further numerical and experimental work can be done regarding the performance of the new shapes to determine how much better they perform under fatigue loading. This can greatly improve the usability of auxetic metamaterials and increase the confidence in their capabilities, increasing their direct use in industry.

References

[1] A. Love, "Ratio of lateral contration to longitudinal extenxion," in ***A treatise on the mathematical theory of elasticity***, 1892.

[2] R. H. Baughman, J. M. Shacklette, A. A. Zakhidov and S. Stafström, "Negative poisson's ratios as a common feature of cubic metals," *NAture,* vol. 392, pp. 362-365, 1998.

[3] V. V. Krasavin and A. V. Krasavin, "Auxetic properties of cubic metal single crystals," *Physica Status Solidi (B) Basic Research*, vol. 251, no. 11, pp. 2314-2320, 2014.

[4] J. Grima, R. Jackson, A. Alderson and K. E. Evans, "Do zeolites have negative Poisson's ratios?," *Advanced materials,* vol. 12, no. 24, pp. 1912-1918, 2000.

[5] X. Tan, W. Jo, T. Granzow, J. Frederick, E. Aulbach and J. Rödel, "Auxetic behavior under electrical loads in an induced ferroelectric phase," *Applied Physics Letters,* vol. 94, no. 4, 2009.

[6] R. S. Lakes, "Foam Structures with a Negative Poisson ' s Ratio," *Science,* vol. 235, no. 4792, pp. 1038-1040, 1987.

[7] G. Wei and S. Edwards, "Auxeticity windows for composites," *Physica A: Statistical Mechanics and its Applications,* vol. 258, no. 1-2, pp. 5-10, 1998.

[8] G. Wei and S. Edwards, "Effective elastic properties of composites of ellipsoids (II). Nearly disk- and needle-like inclusions," *Physica A: Statistical Mechanics and its Applications,* vol. 264, no. 3-4, pp. 404-423, 1999.

[9] G. Wei and S. Edwards, "Effective elastic properties of composites of ellipsoids (I). Nearly spherical inclusions," *Physica A: Statistical Mechanics and its Applications,* vol. 264, no. 3-4, pp. 388-403, 1999.

[10] K. Bertoldi, M. C. Boyce, S. Deschanel, S. M. Prange and T. Mullin, "Mechanics of deformation-triggered pattern transformations and superelastic behavior in periodic elastomeric structures," *Journal of the Mechanics and Physics of Solids,* vol. 56, no. 8, pp. 3642-2668, 2008.

[11] J. T. B. Overvelde, S. Shan and K. Bertoldi, "Compaction through buckling in 2D periodic, soft and porous structures: Effect of pore shape," *Advanced Materials,* vol. 24, no. 17, pp. 2337-2342, 2012.

[12] F. Scrapa, W. Bullough and P. Lumely, "Trends in acoustic properties of iron particule seeded auxetic plyurathene foam," *Smart Material Structure,* vol. 218, pp. 241-244, 2004.

[13] K. E. Evans and A. Alderson, "Auxetic materials: Functional materials and structures from lateral thinking!," *Advanced Materials,* vol. 12, no. 9, pp. 617-628, 2000.

[14] M. Ahsanfar and S. A. Galehdari, "Optimum Design for Graded Honeycomb as Energy Absorber Device in Elevator Cabin," *Procedia Engineering,* vol. 173, pp. 1292-1298, 2017.

[15] Y. Liu and H. Hu, "A review on auxetic structures and polymeric materials," *Scientific Research and Essays,* vol. 5, no. 10, pp. 1052-1063, 2010.

[16] J. N. Grima and R. Gatt, "Perforated sheets exhibiting negative Poisson's ratios," *Advanced Engineering Materials,* vol. 12, no. 6, pp. 460-464, 2010.

[17] H. Mitschke, J. Schwerdtfeger, F. Schury, M. Stingl, C. Körner, R. F. Singer, V. Robins, K. Mecke and G. E. Schröder-Turk, "Finding auxetic frameworks in periodic tessellations," *Advanced Materials,* vol. 23, no. 22-23, pp. 2669-2674, 2011.

[18] M. Taylor, L. Francesconi, M. Gerendás, A. Shanian, C. Carson and K. Bertoldi, "Low porosity metallic periodic structures with negative poisson's ratio," *Advanced Materials,* vol. 26, no. 15, pp. 2365-23-70, 2014.

[19] A. Ghaedizadeh, J. Shen, X. Ren and Y. M. Xie, "Tuning the performance of metallic auxetic metamaterials by using buckling and plasticity," *Materials,* vol. 9, no. 1, pp. 1-17, 2016.

[20] M. H. Sadd, **Elasticity: Theory, Applications, and Numerics**, Burlington, MA: Elsevier Inc., 2014.

[21] F. Javid, J. Liu, A. Rafsanjani, M. Schaenzer, M. Q. Pham, D. Backman, S. Yandt, M. C. Innes, C. Booth-Morrison, M. Gerendas, T. Scarinci, A. Shanian and K. Bertoldi, "On the design of porous structures with enhanced fatigue life," *Extreme Mechanics Letters,* vol. 16, pp. 13-17, 2017.

[22] K. Bertoldi, M. Taylor, A. Shanian, M. Gerendas and C. Carson, "Void structures with repeating elongated-aperture pattern". United States of America Patent 2016/0025343 A1, 28 January 2016.

[23] M. R. Ayatollahi, S. M. J. Razavi, C. Sommitsch and C. Moser, "Fatigue Life Extension by Crack Repair Using Double Stop-Hole Technique," *Materials Science Forum,* vol. 879, pp. 3-8, 2016.

[24] M. Fanni, N. Fouda, M. A. N. Shabara and M. Awad, "New crack stop hole shape using structural optimizing technique," *Ain Shams Engineering Journal,* vol. 6, no. 3, pp. 987-999, 2015.

[25] J. Gielis, "A generic geometric transformation that unifies a wide range of natural and abstract shapes," *American Journal of Botany,* vol. 90, no. 3, pp. 333-338, 2003.

[26] X. Zihui, Z. Chuwei, Y. Qiaoling and W. Xinwei, "On selection of repeated unit cell model and application of unified periodic boundary conditions in micro-mechanical analysis of composites," *International Journal of Solids and Structures*, 2006.

[27] Z. Xia, C. Zhou, Q. Yong and X. Wang, "On selection of repeated unit cell model and application of unified periodic boundary conditions in micro-mechanical analysis of composites," *International Journal of Solids and Structures*, vol. 43, no. 2, pp. 266-278, 2006.

[28] R. Cook, D. Malkus, M. Plesha and R. Witt, **Concepts and Applications of Finite Element Analysis**, Wisconsin: John Wiley & Sons, 2002.

APPENDIX

Appendix A python script: General model.

As mentioned before, finite element simulations were used to compare the behaviors of several geometries. The conditions used to perform the simulations of the models are those of a periodic model following the RVE definitions. To implement this in ABAQUS standard was required to use its capability of receiving instructions via python scripts that can be run using it as a platform, due to its limitation to implement periodic boundary conditions. Other advantages that the usage of python scripting on ABAQUS standard to fully define the model to be simulated presented was that this allowed more repeatability of the analysis performed.

This python script was based on previous work of Dr. Michael Taylor. The code that served as a base for the code used in this investigation was used in Dr. Taylor's research [18] and in his introductory course of finite elements offered in Santa Clara University at the undergraduate level. Therefore, it already implemented the generation of the geometrical model, definition of the material properties, meshing, implementation of the periodic boundary conditions, simulation configuration and execution. However, several additions and modifications were required to adapt this code to the new tasks required. One of the main additions was the introduction of a python function to generate xy points in an array corresponding to positions in the superformula shape to be added at the tip, afterwards this array was going to be used to generate the geometry of the void connecting the points using a spline. Followingly was required the modification of the voids geometry definition, in the original python script the shape of the voids were simply a vertical and a horizontal ellipse, but in this investigation the voids can be described as a symmetrical disposition of two stop-hole features generated by the superformula and connected by the center using a rectangular slot, this can by generated horizontally for the first void and rotated 90 degrees to generate the geometry of an equivalent vertical void to be subtracted from a rectangular matrix geometry by assembly operations in order to generate the RVE of the pattern. Due to the limitation the finite element method to represent accurately the stress fields of geometrical sharp discontinuities and, considering that even though the added stop-holes reduced the sharpness of the voids, they still represented a challenge for the discretization, edge refinement was added to the python script of the model at every geometry related to the stop-hole feature at the tips of the pattern of voids. Those modifications meant that some alterations to the periodic boundary conditions implementation was required to ensure the functionality and validity of it, this meant some deep understanding of the concept of the periodic boundary conditions and its assumptions. Finally, some code was added to the python script to add the capability of generating particular csv reports, exporting images of the different contour plots of interest, and other functions.

The main structure of the python code used to define and execute the simulations of the models in this investigation is outlined below, along with some of the concepts used in its development:

- Import ABAQUS libraries
- Import OS capabilities and CSV capabilities
- Define the superformula points generator function
- Set up parameters:
    - Distance center to center
    - Minor axis (slot width)
    - Pseudo-Axes-Ratio (Enforced void length)

    - Material’s linear properties (Youngs Modulus, Poisson Ratio)
    - Mesh Parameters (Seed Mesh, Edge Seed Mesh)
    - Superformula and scaling parameters:
        - Exponents: n1, n2, n3
        - Frequency: m
        - Amplitude: a, b
        - Scale: B
        - Reference point: x0, y0
- File management conditions for saving information after the execution of the simulations
- Create parts and instances for RVE
    - Matrix
    - Void 1
    - Void 2
    - Assembly
- Create instances for virtual points
- Create mesh
    - General mesh seeding
    - Edge mesh seeding
    - Element type
- Material and section assignment
- Node sets definition
    - General All Nodes and All Elements sets
    - Array of nodes per edge
    - Set of virtual points
    - Periodic node pairing
- Create analysis step
- Boundary Conditions
    - Rigid body motion constraint
    - Virtual points constraints
    - Periodic constraints per node pair top-bottom
    - Periodic constraints per node pair right-left
- Definition of the field output
- Job creation and submission
- Post processing
- File saving
- View setting and image capture
    - View settings
    - Undeformed capture
    - Stress 22 contour plot setting and capture
    - Zoomed stress 22 contour plot setting and capture
    - Displacement 2 contour plot setting and capture
    - Displacement 1 contour plot setting and capture
- Report and .csv file generation

  - Field report of Displacements
  - Field Report of Stress 22
  - Csv file of tip position
  - Csv file of Area of the RVE
  - Csv file of Porosity of the RVE
  - Csv file of Element number

Appendix B python script: ABAQUS/standar- simulation model

```
# -----------------------------------------------------------------------------------------------------
# Michael Taylor & Max Barillas
# 10/6/2016
# 12/27/2017
#
#
# 2D Periodic RVE with Adjustable Hole Shape
# -----------------------------------------------------------------------------------------------------
pathName = "Z:/dcengr/Downloads/Auxetic_Stopholes/"
## "C:/Users/Max Barillas/Documents/Research(Auxtetics)/"
os.chdir(pathName)
# Includes -------------------------------------------------------------------------------------------
from abaqus import *
from abaqusConstants import *
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 *
import visualization
session.journalOptions.setValues(replayGeometry=COORDINATE,recoverGeometry=COORDINATE)
# -----------------------------------------------------------------------------------------------------

##Superformula shape point generator---------------------------------------------------------------------
def superforTip(n1,n2,n3,m,a,b,A,B):
        import math as math
        td=0
        t=td*(math.pi/180.0)
        c1=math.cos(t)
        s1=math.sin(t)
        alpha=m*t*0.25
        c=math.cos(alpha)
        s=math.sin(alpha)
        e=-1/float(n1)
        p1=abs(c/a)**n2
```

```
            p2=abs(s/b)**n3
            r=(p1+p2)**e
            xb=r*c1
            yb=r*s1
            x=(A*xb)
            y=(B*yb)
            pointTip=[x,y]
            return pointTip
# ---------------------------------------------------------------------------------------------------------


##Superformula shape point generator------------------------------------------------------------------------
def superfor(n1,n2,n3,m,a,b,A,B,xo,yo):
    import math as math
    points=[]
    for td in range(0,3600):
        t=td*(math.pi/1800.0)
        c1=math.cos(t)
        s1=math.sin(t)
        alpha=m*t*0.25
        c=math.cos(alpha)
        s=math.sin(alpha)
        e=-1/float(n1)
        p1=abs(c/a)**n2
        p2=abs(s/b)**n3
        r=(p1+p2)**e
        xb=r*c1
        yb=r*s1
        x=(A*xb)+xo
        y=(B*yb)+yo
        pair=[x,y]
        points.append(pair)
    points.append(points[0])
    return points
# ---------------------------------------------------------------------------------------------------------
# Rename model ----------------------------------------------------------------------------------------
modelName = 'model_2DVoidPlate'
mdb.models.changeKey(fromName='Model-1', toName=modelName)
# ---------------------------------------------------------------------------------------------------------
# Create virtual point parts ---------------------------------------------------------------------------
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))
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))
# ---------------------------------------------------------------------------------------------------------
# Material properties ------------------------------------------------------------------------------------
YoungsMod = 200e3   # Young's modulus (in MPa)
PoissonRatio = 0.3
# ---------------------------------------------------------------------------------------------------------
# Geometric properties ---------------------------------------------------------------------------------
center_to_center = 10        # center to center distance for the holes (in mm)
```

```
width_plate = 2.0*center_to_center;             # width of plate
height_plate = 2.0*center_to_center;            # height of plate
minor_axis = 0.45            # minor axis of each void
axes_ratio = 11.0                         # ratio between major and minor axis for the holes
thickness = 0.0                                 # thickness of the plates
seed_mesh =#      # seed-mesh (in mm)
edge_seed_mesh =#          # edge-seed-mesh (in mm)edge_seed_mesh =#
major_axis = minor_axis*axes_ratio;                       # major axis of each void
print minor_axis, major_axis

#Parameters for the superformula shaped stop holes
_______________________________________________________________
n1=#  #exponent
n2=#  #exponent
n3=#  #exponent
m=#   #frequency
a=#   #symmetry
b=#   #symmetry

B=#  #scale in y
A=B #scale in x
xTip, yTip=superforTip(n1,n2,n3,m,a,b,A,B)
xo=major_axis - xTip#center position in x
yo=0.0 #center position in y
#____________________________________________________________________________________________
subPath = pathName + "_mA" + str(minor_axis) + "_AR" + str(axes_ratio) + "_sS" + str(int(seed_mesh*100)) + "B" + str(int(round(B*1000000))) + "n1" + str(int(round(n1*100))) + "n2" + str(int(round(n2*100))) + "n3" + str(int(round(n3*100))) + "m" + str(int(round(m*100))) + "a" + str(int(round(a*100))) + "b" + str(int(round(b*100))) + "eSs" + str(int(round(edge_seed_mesh*100))) + "/"
if not os.path.exists(subPath):
   os.makedirs(subPath)
os.chdir(subPath)
# -----------------------------------------------------------------------------------------------------

# Create parts and instances for plate ------------------------------------------------------------------
partName='part_Plate'
#matrix
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].rectangle(point1=(0.0, 0.0),
   point2=(width_plate, height_plate))
mdb.models[modelName].Part(dimensionality=TWO_D_PLANAR,
name='Matrix',type=DEFORMABLE_BODY)
mdb.models[modelName].parts['Matrix'].BaseShell(sketch=mdb.models[modelName].sketches['__profile__'])
del mdb.models[modelName].sketches['__profile__']
#void1
# Points of the parametric function
Points=superfor(n1,n2,n3,m,a,b,A,B,xo,yo)
l=len(Points)-1
i=0
while i<l:
   phi1=1800-i
```

```
    phi2=1800+i
    yPhi1=Points[phi1][1]
    yPhi2=Points[phi2][1]
    diffY=yPhi1-yPhi2
    if diffY>2.0*minor_axis:
        linePoint1=Points[phi1]
        linePoint2=Points[phi2]
        i=l+1
    i=i+1
linePoint0_1=[0,linePoint1[1]]
linePoint0_2=[0,linePoint2[1]]
interestPoint=[center_to_center+Points[0][0],center_to_center+Points[0][1]]
interestPoint2=[center_to_center+Points[phi1-10][0],center_to_center+Points[phi1-10][1]]
interestPoint3=[center_to_center+Points[phi2+10][0],center_to_center+Points[phi2+10][1]]
# Generate Geometry
mdb.models[modelName].ConstrainedSketch(name='--profile--', sheetSize=20.0)
mdb.models[modelName].sketches['--profile--'].Spline(points=Points)
mdb.models[modelName].sketches['--profile--'].Line(point1=linePoint1,point2=linePoint0_1)
mdb.models[modelName].sketches['--profile--'].Line(point1=linePoint2,point2=linePoint0_2)
mdb.models[modelName].sketches['--profile--'].Line(point1=linePoint0_1,point2=linePoint0_2)
mdb.models[modelName].sketches['--profile--
'].autoTrimCurve(curve1=mdb.models[modelName].sketches['--profile--'].geometry.findAt(Points[0],       ),
point1=Points[1800])
mdb.models[modelName].sketches['--profile--
'].copyMirror(mirrorLine=mdb.models[modelName].sketches['--profile--'].geometry.findAt((0.0,0.0), ),
                                    objectList=(mdb.models[modelName].sketches['--profile--
'].geometry.findAt(Points[0], ),
                                            mdb.models[modelName].sketches['--profile--
'].geometry.findAt((linePoint1[0]-0.01,linePoint1[1]), ),
                                            mdb.models[modelName].sketches['--profile--
'].geometry.findAt((linePoint2[0]-0.01,linePoint2[1]), ),
                                            mdb.models[modelName].sketches['--profile--
'].geometry.findAt(Points[10], ),
                                            mdb.models[modelName].sketches['--profile--
'].geometry.findAt(Points[l-5], ),))
mdb.models[modelName].sketches['--profile--
'].autoTrimCurve(curve1=mdb.models[modelName].sketches['--profile--'].geometry.findAt((0.0,0.0),       ),
point1=(0.0,0.0))
mdb.models[modelName].Part(dimensionality=TWO_D_PLANAR,
name='Void1',type=DEFORMABLE_BODY)
mdb.models[modelName].parts['Void1'].BaseShell(sketch=mdb.models[modelName].sketches['--profile--
'])
#void2
mdb.models[modelName].sketches['--profile--'].rotate(centerPoint=(0.0,0.0), angle=90.0,
                                    objectList=(mdb.models[modelName].sketches['--profile--
'].geometry.findAt(Points[0], ),
                                            mdb.models[modelName].sketches['--profile--
'].geometry.findAt((linePoint1[0]-0.01,linePoint1[1]), ),
                                            mdb.models[modelName].sketches['--profile--
'].geometry.findAt((linePoint2[0]-0.01,linePoint2[1]), ),
                                            mdb.models[modelName].sketches['--profile--
'].geometry.findAt(Points[10], ),
                                    mdb.models[modelName].sketches['--profile--
'].geometry.findAt(Points[l-5], ),
                                    mdb.models[modelName].sketches['--profile--
'].geometry.findAt((-Points[0][0],Points[0][1]), ),
                                    mdb.models[modelName].sketches['--profile--
'].geometry.findAt((-(linePoint1[0]-0.01),linePoint1[1]), ),
                                    mdb.models[modelName].sketches['--profile--
'].geometry.findAt((-(linePoint2[0]-0.01),linePoint2[1]), ),
                                    mdb.models[modelName].sketches['--profile--
'].geometry.findAt((-Points[10][0],Points[10][1]), ),
                                    mdb.models[modelName].sketches['--profile--
'].geometry.findAt((-Points[l-5][0],Points[l-5][1]), ),))
mdb.models[modelName].Part(dimensionality=TWO_D_PLANAR,
name='Void2',type=DEFORMABLE_BODY)
mdb.models[modelName].parts['Void2'].BaseShell(sketch=mdb.models[modelName].sketches['--profile--
'])
del mdb.models[modelName].sketches['--profile--']
#Assembly
#matrix
mdb.models[modelName].rootAssembly.DatumCsysByDefault(CARTESIAN)
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
   'Matrix-1', part=mdb.models[modelName].parts['Matrix'])
#Void 1A
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
   'Void1-1', part=mdb.models[modelName].parts['Void1'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void1-1', ),
vector=(0.0, 0.0, 0.0))
#Void 1B
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
   'Void1-2', part=mdb.models[modelName].parts['Void1'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void1-2', ),
vector=(2.0*center_to_center, 0.0, 0.0))
#Void 1C
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
   'Void1-3', part=mdb.models[modelName].parts['Void1'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void1-3', ),
vector=(1.0*center_to_center, 1.0*center_to_center, 0.0))
#Void 1D
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
   'Void1-4', part=mdb.models[modelName].parts['Void1'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void1-4', ),
vector=(0.0, 2.0*center_to_center, 0.0))
#Void 1E
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
   'Void1-5', part=mdb.models[modelName].parts['Void1'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void1-5', ),
vector=(2.0*center_to_center, 2.0*center_to_center, 0.0))
#Void 2A
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
   'Void2-1', part=mdb.models[modelName].parts['Void2'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void2-1', ),
vector=(0.0, 1.0*center_to_center, 0.0))
#Void 2B
```

```
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
    'Void2-2', part=mdb.models[modelName].parts['Void2'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void2-2', ),
vector=(1.0*center_to_center, 0.0, 0.0))
#Void 2C
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
    'Void2-3', part=mdb.models[modelName].parts['Void2'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void2-3', ),
vector=(2.0*center_to_center, 1.0*center_to_center, 0.0))
#Void 2D
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
    'Void2-4', part=mdb.models[modelName].parts['Void2'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void2-4', ),
vector=(1.0*center_to_center, 2.0*center_to_center, 0.0))
#cut
mdb.models[modelName].rootAssembly.InstanceFromBooleanCut(name=partName,
       instanceToBeCut=mdb.models[modelName].rootAssembly.instances['Matrix-1'],
        cuttingInstances=(
        mdb.models[modelName].rootAssembly.instances['Void1-1'],
        mdb.models[modelName].rootAssembly.instances['Void1-2'],
        mdb.models[modelName].rootAssembly.instances['Void1-3'],
        mdb.models[modelName].rootAssembly.instances['Void1-4'],
        mdb.models[modelName].rootAssembly.instances['Void1-5'],
        mdb.models[modelName].rootAssembly.instances['Void2-1'],
        mdb.models[modelName].rootAssembly.instances['Void2-2'],
        mdb.models[modelName].rootAssembly.instances['Void2-3'],
        mdb.models[modelName].rootAssembly.instances['Void2-4'],
        ),originalInstances=SUPPRESS)
instName = 'inst_2DVoidPlate'
mdb.models[modelName].rootAssembly.features.changeKey(fromName=
    'part_Plate-1', toName=instName)
# --------------------------------------------------------------------------------------------------------------------
# Create instances for 2 virtual points ------------------------------------------------------------------
mdb.models[modelName].rootAssembly.DatumCsysByDefault(CARTESIAN)
# virtual point to constrain x direction
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'])
# ----------------------------------------------------------------------------------------------------------
# Create mesh -------------------------------------------------------------------------------------------------
# set the seed
mdb.models[modelName].parts[partName].seedPart(size=seed_mesh)
mdb.models[modelName].parts[partName].setMeshControls(elemShape=TRI, regions=
    mdb.models[modelName].parts[partName].faces.findAt(((center_to_center/2.0,      center_to_center/2.0,
thickness/2.0), )))
#Edge refinement
#Edge 1
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=
(mdb.models[modelName].parts[partName].edges.findAt((interestPoint[0],interestPoint[1],thickness/2.0),
), ), size=edge_seed_mesh)
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=

(mdb.models[modelName].parts[partName].edges.findAt((interestPoint2[0],interestPoint2[1],thickness/2.0)
, ), ), size=edge_seed_mesh)


#Edge 2
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=
    (mdb.models[modelName].parts[partName].edges.findAt((2*center_to_center-
interestPoint[0],interestPoint[1],thickness/2.0), ), ), size=edge_seed_mesh)
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=
    (mdb.models[modelName].parts[partName].edges.findAt((2*center_to_center-
interestPoint2[0],interestPoint2[1],thickness/2.0), ), ), size=edge_seed_mesh)
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=
    (mdb.models[modelName].parts[partName].edges.findAt((2*center_to_center-
interestPoint3[0],interestPoint3[1],thickness/2.0), ), ), size=edge_seed_mesh)

#Edge 3
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=
    (mdb.models[modelName].parts[partName].edges.findAt((interestPoint2[0]-
width_plate/2.0,interestPoint2[1]-height_plate/2.0,thickness/2.0), ), ), size=edge_seed_mesh)

#Edge 4
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=
    (mdb.models[modelName].parts[partName].edges.findAt((interestPoint[0]-
width_plate/2.0,interestPoint[1]+height_plate/2.0,thickness/2.0), ), ), size=edge_seed_mesh)

#Edge 5
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=
    (mdb.models[modelName].parts[partName].edges.findAt((2*center_to_center-
interestPoint3[0]+width_plate/2.0,interestPoint3[1]+height_plate/2,thickness/2.0),          ),          ),
size=edge_seed_mesh)

#Edge 6
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=
    (mdb.models[modelName].parts[partName].edges.findAt((2*center_to_center-
interestPoint[0]+width_plate/2,interestPoint[1]-height_plate/2,thickness/2.0), ), ), size=edge_seed_mesh)
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=
    (mdb.models[modelName].parts[partName].edges.findAt((2*center_to_center-
interestPoint2[0]+width_plate/2,interestPoint2[1]-height_plate/2,thickness/2.0), ), ), size=edge_seed_mesh)
```

```
#Edge 7
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=
    (mdb.models[modelName].parts[partName].edges.findAt((interestPoint2[1]-
width_plate/2,2*center_to_center-interestPoint2[0],thickness/2.0), ), ), size=edge_seed_mesh)

#Edge 8
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=
    (mdb.models[modelName].parts[partName].edges.findAt((interestPoint2[1]-
width_plate/2,interestPoint2[0],thickness/2.0), ), ), size=edge_seed_mesh)

#Edge 9
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=

(mdb.models[modelName].parts[partName].edges.findAt((interestPoint3[1]+width_plate/2,interestPoint3[0
],thickness/2.0), ), ), size=edge_seed_mesh)

#Edge 10
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=

(mdb.models[modelName].parts[partName].edges.findAt((interestPoint3[1]+width_plate/2,2*center_to_ce
nter-interestPoint3[0],thickness/2.0), ), ), size=edge_seed_mesh)

#Edge 11
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=
    (mdb.models[modelName].parts[partName].edges.findAt((interestPoint[1],2*center_to_center-
interestPoint[0]+height_plate/2,thickness/2.0), ), ), size=edge_seed_mesh)
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=
    (mdb.models[modelName].parts[partName].edges.findAt((interestPoint2[1],2*center_to_center-
interestPoint2[0]+height_plate/2,thickness/2.0), ), ), size=edge_seed_mesh)

#Edge 12
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=
    (mdb.models[modelName].parts[partName].edges.findAt((interestPoint[1],interestPoint[0]-
height_plate/2,thickness/2.0), ), ), size=edge_seed_mesh)
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=
    (mdb.models[modelName].parts[partName].edges.findAt((interestPoint2[1],interestPoint2[0]-
height_plate/2,thickness/2.0), ), ), size=edge_seed_mesh)
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedEdgeBySize(constraint=
    FINER, edges=
    (mdb.models[modelName].parts[partName].edges.findAt((interestPoint3[1],interestPoint3[0]-
height_plate/2,thickness/2.0), ), ), size=edge_seed_mesh)

#ElemType
mdb.models[modelName].parts[partName].setElementType(elemTypes=(ElemType(elemCode=CPS8R,
elemLibrary=STANDARD), ElemType(elemCode=CPS6,
    elemLibrary=STANDARD)),
regions=(mdb.models[modelName].parts[partName].faces.findAt(((center_to_center/2.0,
center_to_center/2.0, thickness/2.0), )), ))

mdb.models[modelName].parts[partName].generateMesh()
ElemNum=mdb.models[modelName].parts[partName].getMeshStats().numTriElems
# -----------------------------------------------------------------------------------------------------------
# Create material ----------------------------------------------------------------------------------------
mdb.models[modelName].Material(description='Linearly elastic material model', name='LinearElastic')
mdb.models[modelName].materials['LinearElastic'].Elastic(table=((YoungsMod, PoissonRatio), ))
# ------------------------------------------------------------------------------------------------------
# Create section, assign to part ----------------------------------------------------------------------------
mdb.models[modelName].HomogeneousSolidSection(material='LinearElastic',
name='section_2DVoidPlate', thickness=None)
mdb.models[modelName].parts[partName].SectionAssignment(offset=0.0,                          offsetField='',
offsetType=MIDDLE_SURFACE, region=Region(
    faces=mdb.models[modelName].parts[partName].faces.findAt(((
    center_to_center/2.0,       center_to_center/2.0,       thickness/2.0),       (0.0,       0.0,       1.0)),       )),
sectionName='section_2DVoidPlate')
# ------------------------------------------------------------------------------------------------------
# Define sets containing all nodes and elements ------------------------------------------------------------------
mdb.models[modelName].parts[partName].Set(name='set_AllElements',
elements=mdb.models[modelName].parts[partName].elements)
mdb.models[modelName].parts[partName].Set(name='set_AllNodes',
nodes=mdb.models[modelName].parts[partName].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 = edge_seed_mesh/20.0
# loop over all nodes and sort out nodes on the edges
for N in mdb.models[modelName].parts[partName].nodes:
    nodeCoord = N.coordinates
    if (fabs(nodeCoord[0]-major_axis) < 100.0*eps) and (fabs(nodeCoord[1]-major_axis) < 100.0*eps):
            node_RBM.append(N.label)
    elif (fabs(nodeCoord[0]) < eps):
            nodes_leftEdge.append(N.label)
    elif (fabs(nodeCoord[0]-width_plate) < eps):
            nodes_rightEdge.append(N.label)
    elif (fabs(nodeCoord[1]) < eps):
            nodes_bottomEdge.append(N.label)
    elif (fabs(nodeCoord[1]-height_plate) < eps):
            nodes_topEdge.append(N.label)
mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodesRightEdge',
nodeLabels=nodes_rightEdge)
mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodesLeftEdge',
nodeLabels=nodes_leftEdge)
```

```
mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodesTopEdge',
nodeLabels=nodes_topEdge)
mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodesBottomEdge',
nodeLabels=nodes_bottomEdge)
mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodeRBM',
nodeLabels=(node_RBM[0],))
# create sets for virtual points
mdb.models[modelName].parts['part_VPx'].Set(name='set_VPx',
referencePoints=(mdb.models[modelName].parts['part_VPx'].referencePoints[1], ))
mdb.models[modelName].parts['part_VPy'].Set(name='set_VPy',
referencePoints=(mdb.models[modelName].parts['part_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].parts[partName].sets['set_NodesLeftEdge'].nodes[i].coordinates
	mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodesLPair_' + str(i),
nodeLabels=(nodes_leftEdge[i],))
	for j in range (0, len(nodes_rightEdge)):
		rightCoords                                                                                           =
mdb.models[modelName].parts[partName].sets['set_NodesRightEdge'].nodes[j].coordinates
		if (fabs(leftCoords[1] - rightCoords[1]) < eps/10):

	mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodesRPair_' + str(i),
nodeLabels=(nodes_rightEdge[j],))
# Look at top and bottom sides
for i in range (0, len(nodes_topEdge)):
	topCoords                                                                                                     =
mdb.models[modelName].parts[partName].sets['set_NodesTopEdge'].nodes[i].coordinates
	mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodesTPair_' + str(i),
nodeLabels=(nodes_topEdge[i],))
	for j in range (0, len(nodes_bottomEdge)):
		bottomCoords                                                                                          =
mdb.models[modelName].parts[partName].sets['set_NodesBottomEdge'].nodes[j].coordinates
		if (fabs(topCoords[0] - bottomCoords[0]) < eps/10):

	mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodesBPair_' + str(i),
nodeLabels=(nodes_bottomEdge[j],))
# -----------------------------------------------------------------------------------------------------------------
# Create analysis step -------------------------------------------------------------------------------------
mdb.models[modelName].StaticStep(description=
   'Step for uniaxial compression in 2-2 direction', name='step_Compression',
   previous='Initial')
mdb.models[modelName].steps['step_Compression'].setValues(
   adaptiveDampingRatio=None,
   continueDampingFactors=False, matrixSolver=DIRECT,
   solutionTechnique=FULL_NEWTON, stabilizationMethod=NONE)
# ---------------------------------------------------------------------------------------------------------------
# Set up BCs -----------------------------------------------------------------------------------------------------------
# fix point to prevent rigid body motion
mdb.models[modelName].DisplacementBC(amplitude=UNSET, createStepName=
   'step_Compression', distributionType=UNIFORM, fieldName='', fixed=OFF,
    localCsys=None, name='bc_preventRBM', region=
    mdb.models[modelName].rootAssembly.instances[instName].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=
    'step_Compression', distributionType=UNIFORM, fieldName='', fixed=OFF,
    localCsys=None, name='bc_VPx', region=
    mdb.models[modelName].rootAssembly.instances['inst_VPx'].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=
    'step_Compression', distributionType=UNIFORM, fieldName='', fixed=OFF,
    localCsys=None, name='bc_VPy', region=
    mdb.models[modelName].rootAssembly.instances['inst_VPy'].sets['set_VPy']
    , u1=UNSET, u2=-0.005, 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].parts[partName].sets['set_NodesLPair_'                    +
str(i)].nodes[0].coordinates
    rightCoord=mdb.models[modelName].parts[partName].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, 'inst_2DVoidPlate.set_NodesRPair_' + str(i), 1),
        (-1.0, 'inst_2DVoidPlate.set_NodesLPair_' + str(i), 1),
        (coeff1, 'inst_VPx.set_VPx', 1)))
    # y-coordinate (Uy_Vpx, H21)
    mdb.models[modelName].Equation(name='constraint_yLR_' + str(i), terms=(
        ( 1.0, 'inst_2DVoidPlate.set_NodesRPair_' + str(i), 2),
        (-1.0, 'inst_2DVoidPlate.set_NodesLPair_' + str(i), 2),
        (coeff1, 'inst_VPx.set_VPx', 2)))
# top and bottom edges
for i in range(0,len(nodes_bottomEdge)):
    # preparation of Coefficients
    bottomCoord=mdb.models[modelName].parts[partName].sets['set_NodesBPair_'                  +
str(i)].nodes[0].coordinates
    topCoord=mdb.models[modelName].parts[partName].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, 'inst_2DVoidPlate.set_NodesTPair_' + str(i), 1),
        (-1.0, 'inst_2DVoidPlate.set_NodesBPair_' + str(i), 1),
        (coeff2, 'inst_VPy.set_VPy', 1)))
    # y-coordinate (Uy_Vpy, H22)
    mdb.models[modelName].Equation(name='constraint_yTB_' + str(i), terms=(
        ( 1.0, 'inst_2DVoidPlate.set_NodesTPair_' + str(i), 2),
        (-1.0, 'inst_2DVoidPlate.set_NodesBPair_' + str(i), 2),
        (coeff2, 'inst_VPy.set_VPy', 2)))
# ---------------------------------------------------------------------------------------------------------------------
# Field Output --------------------------------------------------------------------------------------------------------------
# Force/Displacement at the Virtual Points
mdb.models[modelName].FieldOutputRequest(createStepName=
    'step_Compression', name='output_VPx', rebar=EXCLUDE, region=
    mdb.models[modelName].rootAssembly.instances['inst_VPx'].sets['set_VPx']
    , sectionPoints=DEFAULT, variables=('RF', 'U', 'S'))
mdb.models[modelName].FieldOutputRequest(createStepName=
    'step_Compression', name='output_VPy', rebar=EXCLUDE, region=
    mdb.models[modelName].rootAssembly.instances['inst_VPy'].sets['set_VPy']
    , sectionPoints=DEFAULT, variables=('RF', 'U', 'S'))
#----------------------------------------------------------------------------------------------------------------------
#Create and submit the job for processing ---------------------------------------------------------------------------------
---
mdb.Job(contactPrint=OFF, description='', echoPrint=OFF, explicitPrecision=
    DOUBLE, historyPrint=OFF, memory=16000, memoryUnits=MEGA_BYTES, model=
    modelName, modelPrint=OFF, multiprocessingMode=DEFAULT, name='job_2DVoidPlate',
    nodalOutputPrecision=SINGLE, numCpus=1, numDomains=1,
    parallelizationMethodExplicit=DOMAIN, scratch='', type=ANALYSIS,
    userSubroutine='')
mdb.jobs['job_2DVoidPlate'].submit(consistencyChecking=ON)
mdb.jobs['job_2DVoidPlate'].waitForCompletion()
mdb.saveAs(pathName=subPath + 'job_2DVoidPlate.cae')
#----------------------------------------------------------------------------------------------------------------------
#Postprocessing ----------------------------------------------------------------------------------------------------------
o1 = session.openOdb(name= subPath + 'job_2DVoidPlate.odb')
session.viewports['Viewport: 1'].setValues(displayedObject=o1)
odb = session.odbs[subPath +'job_2DVoidPlate.odb']

##View-------------------------------


myViewport = session.Viewport(name='myViewport', origin=(10, 10), width=180, height=180)
myOdb = visualization.openOdb(path=subPath +'job_2DVoidPlate.odb')
myViewport.setValues(displayedObject=myOdb)

# set viewport settings
v = 'Front'
myViewport.view.setValues(session.views[v])
myViewport.maximize()
myViewport.view.fitView()
myViewport.odbDisplay.basicOptions.setValues(coordSystemDisplay=ON, translucencySort=ON)
myViewport.odbDisplay.commonOptions.setValues(visibleEdges=FEATURE, uniformScaleFactor=1.0)  #
NONE
myViewport.odbDisplay.contourOptions.setValues(contourStyle=CONTINUOUS)        #        DISCRETE
CONTINUOUS
#myViewport.odbDisplay.contourOptions.setValues(showMinLocation=ON,showMaxLocation=ON)
#myViewport.odbDisplay.contourOptions.setValues(numIntervals=6)
myViewport.viewportAnnotationOptions.setValues(triad=OFF, title=OFF, state=OFF,  compass=OFF,
                                   legend=OFF, legendPosition=(75, 95), legendBox=OFF,
                                   legendFont='-*-verdana-medium-r-normal-*-*-120-*-*-p-*-*-*',
                                   statePosition=(1, 15),
                                   titleFont='-*-verdana-medium-r-normal-*-*-120-*-*-p-*-*-*',
                                   stateFont='-*-verdana-medium-r-normal-*-*-120-*-*-p-*-*-*')


# saving undeformed image
myViewport.odbDisplay.display.setValues(plotState=(UNDEFORMED, ))
path_filename = subPath +'job_2DVoidPlate_UNDEFORMED'
try:
   session.printToFile(path_filename, PNG, (myViewport,))
except:
   pass

# save stress plots
o = 'S' ; c = 'S22' ; s = 0 ; f = -1
myViewport.odbDisplay.display.setValues(plotState=(CONTOURS_ON_DEF, ))
myViewport.odbDisplay.setFrame(step=s, frame=f)
myViewport.odbDisplay.setPrimaryVariable(variableLabel=o,outputPosition=INTEGRATION_POINT,ref
inement=(COMPONENT, c), )
myViewport.odbDisplay.commonOptions.setValues(deformationScaling=UNIFORM,
uniformScaleFactor=1.0)
myViewport.odbDisplay.contourOptions.setValues(contourStyle=CONTINUOUS,
maxAutoCompute=OFF,       maxValue=-2443,       minAutoCompute=OFF,       minValue=-5500,
outsideLimitsMode=SPECIFY, outsideLimitsBelowColor='DeepPink')
path_filename = subPath +'job_2DVoidPlate_S11'
try:
   myViewport.view.fitView()
   session.printToFile(path_filename+'.png', PNG, (myViewport,))
except:
   pass
# save stress plots zoom
o = 'S' ; c = 'S22' ; s = 0 ; f = -1
myViewport.view.fitView()
myViewport.view.zoomRectangle(point1=(0.25-0.05, 0.5+0.1), point2=(0.25+0.1, 0.5-0.1) )
myViewport.odbDisplay.display.setValues(plotState=(CONTOURS_ON_DEF, ))
myViewport.odbDisplay.setFrame(step=s, frame=f)
myViewport.odbDisplay.setPrimaryVariable(variableLabel=o,outputPosition=INTEGRATION_POINT,ref
inement=(COMPONENT, c), )
myViewport.odbDisplay.commonOptions.setValues(deformationScaling=UNIFORM,
uniformScaleFactor=1.0)
myViewport.odbDisplay.contourOptions.setValues(contourStyle=CONTINUOUS,
maxAutoCompute=OFF,       maxValue=-2443,       minAutoCompute=OFF,       minValue=-5500,
outsideLimitsMode=SPECIFY, outsideLimitsBelowColor='DeepPink')
path_filename = subPath +'_S22_zoom'
try:
        myViewport.view.fitView()
        myViewport.view.zoomRectangle(point1=(0.25-0.1, 0.5+0.1), point2=(0.25+0.1, 0.5-0.1) )
        session.printToFile(path_filename+'.png', PNG, (myViewport,))
except:
   pass

myViewport.view.fitView()
```

```
# save displacement plots
o = 'U' ; c = 'U1'
myViewport.odbDisplay.display.setValues(plotState=(CONTOURS_ON_DEF, ))
myViewport.odbDisplay.setFrame(step=s, frame=f)
myViewport.odbDisplay.setPrimaryVariable(variableLabel=o,outputPosition=NODAL,refinement=(COM
PONENT, c), )
myViewport.odbDisplay.commonOptions.setValues(deformationScaling=UNIFORM,uniformScaleFactor=
1.0)
myViewport.odbDisplay.contourOptions.setValues(contourStyle=CONTINUOUS,
maxAutoCompute=OFF, maxValue=0.038, minAutoCompute=OFF, minValue=-0.068)
path_filename = subPath +'job_2DVoidPlate_U1'
try:
   myViewport.view.fitView()
   session.printToFile(path_filename+'.png', PNG, (myViewport,))
except:
   pass


o = 'U' ; c = 'U2'
myViewport.odbDisplay.display.setValues(plotState=(CONTOURS_ON_DEF, ))
myViewport.odbDisplay.setFrame(step=s, frame=f)
myViewport.odbDisplay.setPrimaryVariable(variableLabel=o,outputPosition=NODAL,refinement=(COM
PONENT, c), )
myViewport.odbDisplay.commonOptions.setValues(deformationScaling=UNIFORM,
uniformScaleFactor=1)
myViewport.odbDisplay.contourOptions.setValues(contourStyle=CONTINUOUS,
maxAutoCompute=OFF, maxValue=0.028, minAutoCompute=OFF, minValue=-0.089)
path_filename = subPath +'job_2DVoidPlate_U2'
try:
   myViewport.view.fitView()
   session.printToFile(path_filename+'.png', PNG, (myViewport,))
except:
   pass



##-----------


session.fieldReportOptions.setValues(printTotal=OFF,                                    printMinMax=OFF,
reportFormat=COMMA_SEPARATED_VALUES)
session.writeFieldReport(fileName='job_2DVoidPlate_stress.csv', append=ON,
           sortItem='Element Label', odb=odb, step=0, frame=1, outputPosition=INTEGRATION_POINT,
           variable=(('S', INTEGRATION_POINT, ((COMPONENT, 'S11'), (COMPONENT, 'S22'),)), ))
session.writeFieldReport(fileName='job_2DVoidPlate.csv', append=ON,
   sortItem='Node Label', odb=odb, step=0, frame=1, outputPosition=NODAL,
   variable=(('RF', NODAL, ((COMPONENT, 'RF1'), (COMPONENT, 'RF2'), )), ('U',
   NODAL, ((COMPONENT, 'U1'), (COMPONENT, 'U2'), )), ('S', NODAL, ((COMPONENT, 'S11'),
(COMPONENT, 'S22'),)), ))
import csv
csv_name='Interest_Point.csv'
```

```
area_plate = width_plate*height_plate;
with open(csv_name, 'wb') as myfile:
    wr = csv.writer(myfile, quoting=csv.QUOTE_NONE)
    wr.writerow(interestPoint)
A=mdb.models[modelName].parts[partName].faces[0].getSize()
Area=[A, ]
por=(1-(A/area_plate))*100
Poros=[por, ]
import csv
csv_name='Interest_Point.csv'
with open(csv_name, 'wb') as myfile:
    wr = csv.writer(myfile, quoting=csv.QUOTE_NONE)
    wr.writerow(interestPoint)
csv_name='Area.csv'
with open(csv_name, 'wb') as myfile:
    wr = csv.writer(myfile, quoting=csv.QUOTE_NONE)
    wr.writerow(Area)
csv_name='Porosity.csv'
with open(csv_name, 'wb') as myfile:
    wr = csv.writer(myfile, quoting=csv.QUOTE_NONE)
    wr.writerow(Poros)
ElemNum=(ElemNum, )
csv_name='ElemNum.csv'
with open(csv_name, 'wb') as myfile:
    wr = csv.writer(myfile, quoting=csv.QUOTE_NONE)
    wr.writerow(ElemNum)

# ------------------------------------------------------------------------------------------------------------------------
```

Appendix C MATLAB-python-ABAQUS interaction script

```
function
[Por,nu,I_point,S11_max,S22_max,S11_min,S22_min,ElemNum]=funct(n1,n2,n3,mM,a,b,B,seed_mesh,ed
ge_seed_mesh,i)
%This MATLAB Script comunicates with ABAQUS by editing a python script
%using cmd commands and powershell commands, then using cmd comands to run
%the python script on ABAQUS

%This first line sets a character line with the general setup of lines to
%be changed in the python script by a powershell command in the windows cmd
%the # characters represent blank spaces to be filled

%Will require a base txt file containg the base python code and the blank
%spaces in this location
\\samba1\mbarilla\dcengr\Downloads\Auxetic_Stopholes\Auxetic_with_super_formula_stopholes_csv_2_d
c.txt
    command=char(['powershell -command "(Get-Content \\samba1\mbarilla\dcengr\Downloads\'...
        'Auxetic_Stopholes\Auxetic_with_super_formula_stopholes_csv_2_dc.txt )'...
        '.Replace("seed_mesh =#      # seed-mesh (in mm)","seed_mesh =0.15      # seed-mesh (in mm)")'...
        '.Replace("B=#  #scale in y","B=0.07  #scale in y")'...
        '.Replace("edge_seed_mesh =#         # edge-seed-mesh (in mm)","edge_seed_mesh =0.05         # edge-
seed-mesh (in mm)")'...
        '.Replace("n1=#  #exponent","n1=1.0  #exponent")'...
        '.Replace("n2=#  #exponent","n2=4.0  #exponent")'...
        '.Replace("n3=#  #exponent","n3=8.0  #exponent")'...
        '.Replace("m=#   #frequency","m=2.0   #frequency")'...
        '.Replace("a=#   #symmetry","a=1.0   #symmetry")'...
        '.Replace("b=#   #symmetry","b=1.0   #symmetry")'...
        '|                                    Set-Content                                    -Path
\\samba1\mbarilla\dcengr\Downloads\Auxetic_Stopholes\Auxetic_with_super_formula_stopholes_csv_2_d
c']...
        +string(i)+'.txt'");
    %Each of the following sets one of the previous blank spaces found in
    %the above character line
    B=round(B,6);
    B=fix(B*100000)/100000;
    command=replace(command,'B=0.07 ','B=' + string(num2str(B,6)) + ' ');
    command=replace(command,'edge_seed_mesh =0.05 ','edge_seed_mesh  =' + string(edge_seed_mesh) + '
');
    command=replace(command,'b=1.0','b=' + string(b) + ' ');
    command=replace(command,'a=1.0','a=' + string(a) + ' ');
    command=replace(command,'m=2.0','m=' + string(mM) + ' ');
    command=replace(command,'n3=8.0','n3=' + string(n3) + ' ');
    command=replace(command,'n2=4.0','n2=' + string(n2) + ' ');
    command=replace(command,'n1=1.0','n1=' + string(n1) + ' ');
    command=replace(command,'seed_mesh =0.15','seed_mesh =' + string(seed_mesh) + ' ');
    %Display point to review results
    disp(command)
    %Run on Windows cmd
    [b1,b2]=dos(command);
    %Display control variables (0 or 1) for the command runned correctly
    %and any resulting lines recieved by the cmd
    disp(b1)
    disp(b2)

    %renaming as python script
    a1=dos(char('rename
\\samba1\mbarilla\dcengr\Downloads\Auxetic_Stopholes\Auxetic_with_super_formula_stopholes_csv_2_d
c'...
        +string(i)+'.txt Auxetic_with_super_formula_stopholes_csv_2_dc'+string(i)+'.py'));
    disp(a1)

    %Setting path to ABAQUS files to run the abaqus cae command properly
    %(the path may vary depending on the actual computer path)
    dos(char('path ;'));
    com1=char(['cd          C:\Temp\SIMULIA\Commands          &&          abaqus          cae
noGUI=\\samba1\mbarilla\dcengr\Downloads\'...
        'Auxetic_Stopholes\Auxetic_with_super_formula_stopholes_csv_2_dc']+string(i)+'.py');
    disp(com1)
    a2=system(com1);
    disp(a2)
    %------------------------Data acquisition---------------------

    %Depending on the python script several reports can be exported by
    %ABAQUS in different formats. Now the main format used is CSV. Also
    %some variables generated in the python SCRIPT can be exported directly
    %by python in the same format

    %The file location is also defined in the python script, the path is
    %automatically defined by the parameters inserted, therefore it is
    %computed as a combinations of string versions of the variables
    %inserted

    %Important note: since the axis ratio and minor axis are not input
    %variables for this script, if they get changed in the source python
    %(.txt) script they have to be changed manually in the following
    %commands.

    %readtable command to read the file job_2DVoidPlate.csv



    data=readtable(char("\\samba1\mbarilla\dcengr\Downloads\Auxetic_Stopholes\"
+"_mA0.45_AR11.0_sS"+string(seed_mesh*100)...
        + "B" + string((B*1000000)) + "n1" + string(n1*100) + "n2" + string(n2*100) + "n3" + string(n3*100)
+ "m" + string(mM*100)...
        + "a" + string(a*100) + "b" + string(b*100) + "eSs" + string(round(edge_seed_mesh*100,0)) + "\" +
"job_2DVoidPlate.csv"));
    [m,~]=size(data);
    data=table2array(data(1:m,14:15));

    %obtaining displacements of the vitual points as strain (may be
    %subject to a different tranformation to represent actual strain)
    Strain=data([m-1,m],[1,2]);
    Strain=str2double(string(Strain));
    %Poison's ratio computation
    nu=-(Strain(1,1)/Strain(2,2));

    %Reading the variable generated by python Interest_Point.csv, this
    %variable correspond to the tip of the stophole geometry
    I_point=csvread(char("\\samba1\mbarilla\dcengr\Downloads\Auxetic_Stopholes\"
+"_mA0.45_AR11.0_sS"+string(seed_mesh*100)...
        + "B" + string((B*1000000)) + "n1" + string(n1*100) + "n2" + string(n2*100) + "n3" + string(n3*100)
+ "m" + string(mM*100)...
        + "a" + string(a*100) + "b" + string(b*100) + "eSs" + string(round(edge_seed_mesh*100,0)) + "\" +
"Interest_Point.csv"));

    %Reading the variable generated by python ElemNum.csv, this
    ElemNum=csvread(char("\\samba1\mbarilla\dcengr\Downloads\Auxetic_Stopholes\"
+"_mA0.45_AR11.0_sS"+string(seed_mesh*100)...
        + "B" + string((B*1000000)) + "n1" + string(n1*100) + "n2" + string(n2*100) + "n3" + string(n3*100)
+ "m" + string(mM*100)...
        + "a" + string(a*100) + "b" + string(b*100) + "eSs" + string(round(edge_seed_mesh*100,0)) + "\" +
"ElemNum.csv"));


    %Reading the variable generated by python Porosity.csv,
    Por=csvread(char("\\samba1\mbarilla\dcengr\Downloads\Auxetic_Stopholes\"
+"_mA0.45_AR11.0_sS"+string(seed_mesh*100)...
```

```
        + "B" + string((B*1000000)) + "n1" + string(n1*100) + "n2" + string(n2*100) + "n3" + string(n3*100)
+ "m" + string(mM*100)...
        + "a" + string(a*100) + "b" + string(b*100) + "eSs" + string(round(edge_seed_mesh*100,0)) + "\" +
"Porosity.csv"));
    %reading the Stress report for all the integration points generated by
    %ABAQUS as job_2DVoidPlate_stress.csv
    data=readtable(char("\\samba1\mbarilla\dcengr\Downloads\Auxetic_Stopholes\"
+"_mA0.45_AR11.0_sS"+string(seed_mesh*100)...
        + "B" + string((B*1000000)) + "n1" + string(n1*100) + "n2" + string(n2*100) + "n3" + string(n3*100)
+ "m" + string(mM*100)...
        + "a" + string(a*100) + "b" + string(b*100) + "eSs" + string(round(edge_seed_mesh*100,0)) + "\" +
"job_2DVoidPlate_stress.csv"));
    [m,~]=size(data);
    data1=table2array(data(1:m,5:8));
    data2=table2array(data(1:m,13:14));
    data=[data1,data2];
    data=str2double(string(data));

    %reducing the stress points acquired from all the integration ponts to
    %integration points in a region around the interest point
    j=1;
    k=0;
    l=0;
    for i=2:m
        if (data(i,3)-I_point(1,1))<2.5 %size of the region considered
            if ((data(i,4)-I_point(1,2))<2.5) %size of the region considered
                k=[k,data(i,5)];
                l=[l,data(i,6)] ;
                j=j+1;
            end
         end

    end
     %The minimum stress (applied compression strain in the 22 direction)
     %on the region is stored as the stress at the
     %interest point, given that it should be the point of maximum stress
     %concentration. Maximums are also computed for the region but may be
     %not representative of the stress at the interest point
     S11_max=max(k);
     S22_max=max(l);
     S11_min=min(k);
     S22_min=min(l);
end
```

Appendix D MATLAB script: mesh study

```
%requires update to the generalize script that allows to define all the
%parameters of the superformula and to the new data acquisition for the
%S11_min and S22_min
n1=15;
n2=25;
n3=60;
mM=2;
```

```
a=10;
b=5;
B=0.00187;
seed_mesh=0.25;
Range0=0.01;
Range1=0.14;
x=Range0:0.01:Range1;
[~,n]=size(x);
nu_mat=zeros(n,1);
S11_max_mat=zeros(n,1);
S22_max_mat=zeros(n,1);
S11_min_mat=zeros(n,1);
S22_min_mat=zeros(n,1);
Porosity=zeros(n,1);
ElemNumMat=zeros(n,1);
time=zeros(n,1);
j=1;
for x=Range0:0.01:Range1
    xreverse=Range1+Range0-x;
    tic

[Por,nu,I_point,S11_max,S22_max,S11_min,S22_min,ElemNum]=funct(n1,n2,n3,mM,a,b,B,seed_mesh,xr
everse,j) ;
    nu_mat(j)=nu;
    S11_max_mat(j)=S11_max;
    S22_max_mat(j)=S22_max;
    S11_min_mat(j)=S11_min;
    S22_min_mat(j)=S22_min;
    Porosity(j)=Por;
    ElemNumMat(j)=ElemNum;
    time(j)=toc;
    j=j+1;
end

figure
ax1 = subplot(1,3,1);
ax2 = subplot(1,3,2);
ax3 = subplot(1,3,3);
plot(ax1,ElemNumMat,nu_mat)
title(ax1,'nu')
plot(ax2,ElemNumMat,S22_min_mat)
title(ax2,'S22_min')
plot(ax3,ElemNumMat,time)
title(ax3,'time')
```

Appendix E MATLAB script: Batch simulation

```
function Porosity_array(shapes)
A=shapes(1,:);
ll=length(A);
B=shapes(2,:);
M=shapes(3,:);
N1=shapes(4,:);
```

```
N2=shapes(5,:);
N3=shapes(6,:);
S=shapes(7:9,:);
[ii,~]=size(S);
Porosity=zeros(ii,ll);
seed_mesh=0.25;
edge_seed_mesh=[0.05,0.1,0.15];
for l=1:ll
   for i=2:2
       tic
       [Por,~,~,~,~,~,~]=funct(N1(l),N2(l),N3(l),M(l),A(l),B(l),S(i,l),seed_mesh,edge_seed_mesh(i),(ii*l-
ii)+i) ;
       Porosity(i,l)=Por;
       hhh=toc
   end
end
end
```

Appendix F MATLAB script: Total search of scale for a batch of geometries

```
function [nu_f,S22_min_f,Porosity_f,Scale_f]=Scale_total_search_array(shapes)
A=shapes(1,:);
ll=length(A);
B=shapes(2,:);
M=shapes(3,:);
N1=shapes(4,:);
N2=shapes(5,:);
N3=shapes(6,:);
S0=shapes(7,:);
S1=shapes(8,:);
for l=1:ll
       n1=N1(l);
       n2=N2(l);
       n3=N3(l);
       mM=M(l);
       a=A(l);
       b=B(l);
       seed_mesh=0.25;
       Range0=S0(l);
       Range1=S1(l);
       Step=(Range1-Range0)/4;
       x=Range0:Step:Range1;
       [~,n]=size(x);
       edge_seed_mesh=linspace(0.15,0.2,n);
       nu_mat=zeros(n,1);
       S11_max_mat=zeros(n,1);
       S22_max_mat=zeros(n,1);
       S11_min_mat=zeros(n,1);
       S22_min_mat=zeros(n,1);
       Porosity=zeros(n,1);
       q=1;
       for x=Range0:Step:Range1
```

```
[Por,nu,~,S11_max,S22_max,S11_min,S22_min]=funct(n1,n2,n3,mM,a,b,x,seed_mesh,edge_seed_mesh(q
),(16*l-16)+q) ;
            nu_mat(q)=nu;
            S11_max_mat(q)=S11_max;
            S22_max_mat(q)=S22_max;
            S11_min_mat(q)=S11_min;
            S22_min_mat(q)=S22_min;
            Porosity(q)=Por;
            q=q+1;
        end
        x=Range0:Step:Range1;
        % figure
        % ax1 = subplot(4,1,1);
        % ax2 = subplot(4,1,2);
        % ax3 = subplot(4,1,3);
        % ax4 = subplot(4,1,4);
        % plot(ax1,x,nu_mat)
        % title(ax1,'nu')
        % plot(ax2,x,S11_mat)
        % title(ax2,'S11_min')
        % plot(ax3,x,S22_mat)
        % title(ax3,'S22_min')
        % plot(ax4,x,Por_mat)
        % title(ax4,'Por')
        title_f= 'a-'+string(a)+'-b-'+string(b)+'-m-'+string(mM)+'-n1-'+string(n1)...
            +'-n2-'+string(n2)+'-n3-'+string(n3)+'-S0-'+string(S0(l))+'-S1-'+string(S1(l));
        save(char(title_f + '.mat'))
        title_g= 'a:'+string(a)+'-b:'+string(b)+'-m:'+string(mM)+'-n1:'+string(n1)...
            +'-n2:'+string(n2)+'-n3:'+string(n3)+'-S0:'+string(S0(l))+'-S1:'+string(S1(l));
        figure('Name',title_g,'NumberTitle','off');
        ax1 = subplot(2,2,1);
        ax2 = subplot(2,2,2);
        ax3 = subplot(2,2,3);
        ax4 = subplot(2,2,4);
        plot(ax1,x,nu_mat)
        title(ax1,'nu')
        plot(ax2,x,S11_max_mat)
        title(ax2,'S11_max')
        plot(ax3,x,S22_min_mat)
        title(ax3,'S22_min')
        plot(ax4,x,Porosity)
        title(ax4,'Porosity')

        saveas(gcf,char(title_f +'.png'))
        nu_f(:,l)=nu_mat;
        S22_min_f(:,l)=S22_min_mat;
        Porosity_f(:,l)=Porosity;
        Scale_f(:,l)=x';

end
title_comp='nu vs porosity';
figure('Name',title_comp,'NumberTitle','off');
```

```
ax1 = subplot(1,1,1);
for l=1:ll
        plot(ax1,x,nu_mat)
end
end
```