2012

Finite element analysis of 2-D representative volume element

Mandar Kulkarni
Michigan Technological University

Finite Element Analysis of 2-D Representative Volume Element

By,

Mandar Kulkarni

Advisor

Dr. Gregory Odegard

A REPORT

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

(Mechanical Engineering)

MICHIGAN TECHNOLOGICAL UNIVERSITY

Spring 2012

Table of Contents

List of Variables

Γ_R – Boundary region over unit cell R

$u(x, y)$ – Displacement

$t(x, y)$ - Traction

Γ^u_R - Boundary where displacements in terms of $u(x, y)$ is specified over region R

Γ^t_R - Boundary where displacements in terms of $t(x, y)$ is specified over region R

E_{11} - Macroscopic strain rate

u_{ij} - Displacement vector for any material point on the corresponding boundary Γ_{ij}

u_{vi} - Displacement vector for each vertex v_i

1.   Objective and Motivation

With recent advances in computational speed and micromechanics, it become necessary to build models which will result in valuable computational savings that greatly simplifies the pre-processing stage of the analysis to find material properties, failure criteria, or constitutive laws for materials.

Microstructural modification of materials appears to be a promising route to improve mechanical properties. It is evident that knowledge of the relation between the microstructural phenomena and the macroscopic deformation behavior is indispensable when trying to predict macroscopic properties from the microstructure. A quantitative relation can be determined by using homogenization methods, where the heterogeneous material is replaced by an equivalent homogeneous continuum. Using representative volume element, it is possible to study behavior of material under complex loading condition.

The objective is to develop equivalent continuum in such a way that, in a certain sense, it has same average mechanical response as the actual homogenous material.

2.   Representative Volume Element (RVE)

It is argued in Nemat-Nasser and Hori, that Representative Volume Element (RVE) for a material point of a continuum is a material volume which is statistically representative for the infinitesimal material neighborhood of that material point[2]. It is regarded as a heterogeneous material with spatially varying, but known, constitutive properties. It is of importance that statistical properties of the state variables are considered invariant of the position in the material. This is called statistical homogeneity.

One of the key issues that arise when creating finite element micromechanical models is the size of the RVE necessary to capture all of the prominent features of the entire body under study[2]. Drugan and Willis state that the minimum size of the RVE is the smallest volume element of the composite that is ''statistically representative of the composite''[4]. They have shown that the minimum RVE size is at least twice the diameter of the reinforcement, citing a maximum error of five percent in elastic constants obtained with this RVE size. Gusev studied disordered periodic elastic composite unit cells composed of various numbers of identical spheres in order to determine the scatter in elastic constants obtained with different numbers of spheres and found that the scatter is small with only a few dozen spheres in the cell[3].

To model materials, a macroscopic array and a representative volume element (RVE) must be defined. The two macroscopic periodically-spaced arrays of composite material studied for this analysis are shown in Figure 1 and Figure 2. The first macroscopic array is a regular hexagonal array composed of triangular RVEs and the second macroscopic array is a square array composed of square RVEs. Composite arrays can be represented as a periodic assembly of
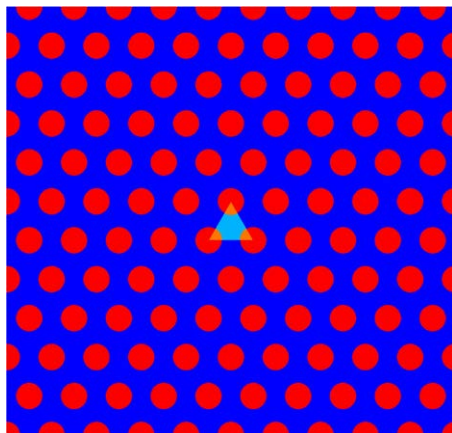
Figure 1: Hexagonal array [2]
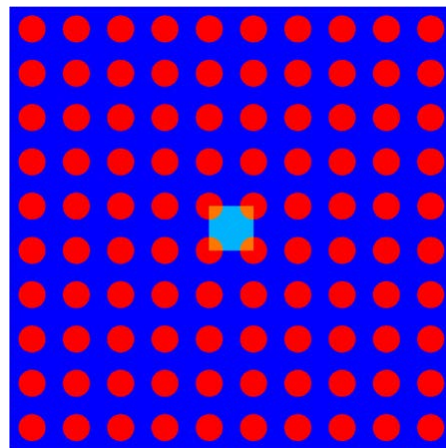
Figure 2: Square array [2]

RVEs. RVE phases are isotropic and linearly elastic and RVE is assumed to be small compared to the size of the macroscopic composite array. RVE is not unique with each RVE in the array exhibiting the same deformation to ensure periodicity as the composite array deforms. The average mechanical properties of the entire macroscopic composite array should be equal to the average mechanical properties of the RVE[2].

For this research work, material array shown in Figure 3 is considered. This consist with steel fabric (in blue color) stacked together and air (white color region) is considered in between stack of this steel fabric. This material array can be reduced to RVE shown in Figure 4 (with white color lines as boundary line). Main purpose to study this type of material array is to check stress wave propagation through stack of steel fabric, especially in area contact between neighboring steel fabrics.
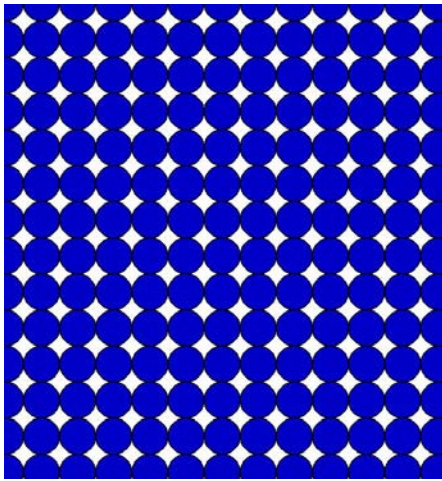
Figure 3: Material array with steel fabric (blue) and air (white) region
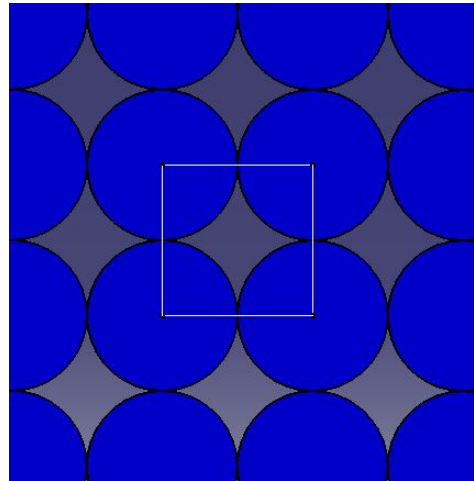
Figure 4: RVE

3.   Microscopic boundary conditions

An essential step in any numerical homogenization method is the choice of appropriate microscopic boundary conditions. Since these boundary conditions are applied on the RVE, and subsequently, the averaged response of the RVE is used to determine the effective properties, this choice requires a thorough motivation. The commonly used boundary conditions in micromechanics are the uniform ones, namely the kinematic uniform boundary conditions, also called Dirichlet or essential boundary conditions, where uniform displacements are applied to the boundary Γ_1 shown in Figure 5(a), and secondly, the static uniform or Neumann boundary conditions, where uniform tractions are prescribed on the edge of the sample Γ_2, shown in Figure 5(b). According to Hill, an RVE is well-defined when the responses under Dirichlet and Neumann boundary conditions coincide[5]. In the studies of Huet et al, the practically relevant case of samples smaller than the RVE is treated and the concept of apparent properties is introduced. In addition to the mentioned uniform boundary conditions, mixed boundary conditions have been proposed[1], inspired by the fact that in experimental set-ups, it is very difficult to realize uniform boundary conditions. In fact, the case of uniaxial tensile conditions, where in one part of the specimen, displacements are prescribed whereas on the remaining part of the sample, forces are prescribed, belongs to the family of mixed boundary conditions, shown in Figure 5(c). These works have shown that the mixed boundary conditions yield better approximations of the effective properties than the uniform ones. Finally, also periodic boundary conditions (PBC) can be formulated. Clearly, these boundary conditions should be applied on a unit cell when the heterogeneous body exhibits a periodic structure, but Terada et al. proved that also for general heterogeneous materials, periodic boundary conditions with relatively small unit cell size provide reasonable estimates of the effective properties, even if the medium does not have actual geometrical periodicity[6].
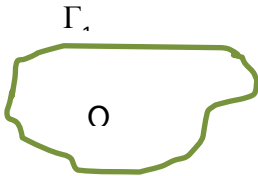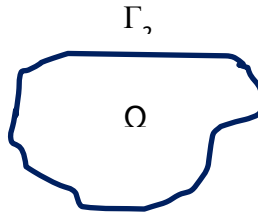

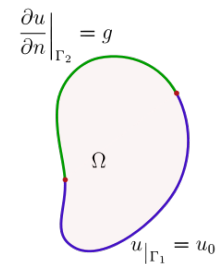**Figure 5(a) Dirichlet BC**   **Figure 5(b) Neumann BC**   **Figure 5(c) Mixed BC**

In this section, the effect of two types of boundary conditions will be studied, namely mixed and periodic boundary conditions. These boundary conditions provide better approximations of the effective properties than the strictly uniform Dirichlet and Neumann boundary conditions. Since the loading conditions on the unit cell are not acquired from any multilevel method, in which case simultaneous calculations are performed on micro and macro-level, and therefore, the boundary conditions on the unit cell are readily available, we prescribe the boundary conditions under tensile conditions.

3.1   Mixed Boundary Condition

In the case of mixed boundary conditions, the boundary Γ_R of the unit cell R is composed of two parts

$\Gamma_R = \Gamma^u_R \cup \Gamma^t_R$

in which, Γ^u_R is that part of the boundary where displacements u(x, y) are prescribed, whereas on Γ^t_R , tractions t(x, y) are applied. The boundary condition have been prescribed under uniaxial tensile conditions in the y1-direction, with a constant macroscopic strain rate $E_{11}= 1s^{-1}$. The boundary conditions now read

$$u1\left(\pm\frac{a}{2},y2\right) = \pm\frac{a}{2}E11 \; on \; \Gamma^u_R$$

$$t2\left(y1\pm\frac{a}{2}\right) = 0 \; on \;\; \Gamma^t_R$$

with E_{11} the total applied strain component. In addition, the edges on the boundary part Γ^t_R are forced to remain straight during loading by means of tying, since otherwise, artificial holes would be created in the heterogeneous specimen. These constraints can be easily implemented using a dependency matrix in the numerical treatment.

3.2 Periodic Boundary Condition

When analyzing a RVE, periodic boundary conditions must be applied to the RVE to ensure compatibility of deformation and correct computation of stress and strain. A displacement-based finite element method is employed in this analysis.

The macroscopic body exhibits a periodic structure, formed by the square unit cell with, 'a' the typical dimension of the cell. When we assume a two-dimensional orthonormal base [e1, e2] in the direction of the axes [x1, x2], then, due to the periodicity assumption, the state

variables are invariant with respect to any translation m1v1+ m2v2, where m1 and m2 are integers, v1 = ae_1, and v2 =ae_2[7]. In R2, the boundary ΓR of R is assumed to be enclosed by two boundary pairs ΓR = Γ1 ∪ Γ2, where both pairs are comprised of opposing edges. This is illustrated in Figure 6, where an arbitrary periodically deformed unit cell under uniaxial tensile conditions is shown.
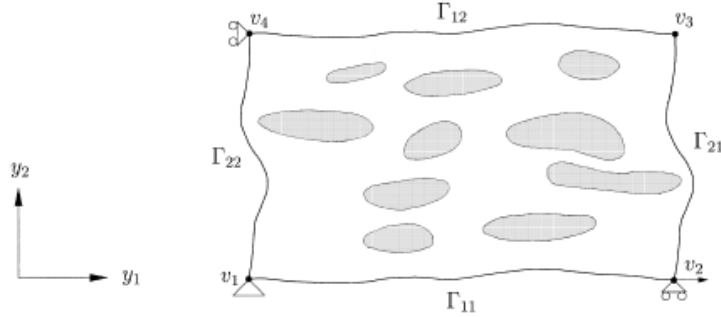
Figure 6: Periodically deformed unit cell with boundary ΓR and vertices v_i[1]

Each boundary pair Γp should, due to the periodicity assumption, be compatible, which means that the deformation of each boundary pair is equal and the stress vectors are opposite in sign on each pair[7]. In Smit et al., appropriate displacement boundary conditions are derived[8]. Here, only the final expressions are presented:

$u_{12} - u_{v4} = u_{11} - u_{v1}$

$u_{22} - u_{v1} = u_{21} - u_{v2}$

where u_{ij} is the displacement vector for any material point on the corresponding boundary Γ_{ij}, and u_{vi} the displacement vector for each vertex v_i.

Figure 8 illustrates the deformation of the square RVE of the composite macroscopic array shown in Figure 7, when shear loading is applied to RVE (shown in red color direction). It should be noted that this deformation is expected when PBC is imposed on RVE. But in actual, due to inconsistency in geometry and non-linearity in model, the deformed shape will not will be exactly as shown on Figure 7. The deformed edge may not follow the linear pattern and may appear in curvilinear shape along the edges of RVE.
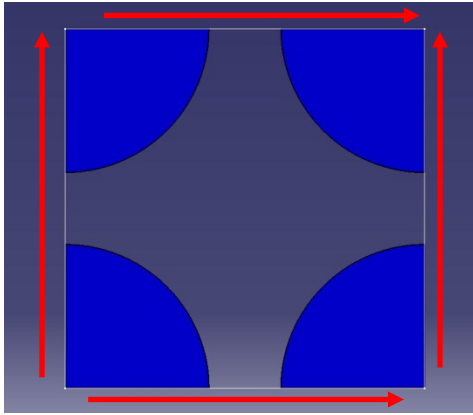
Figure 7: Square RVE
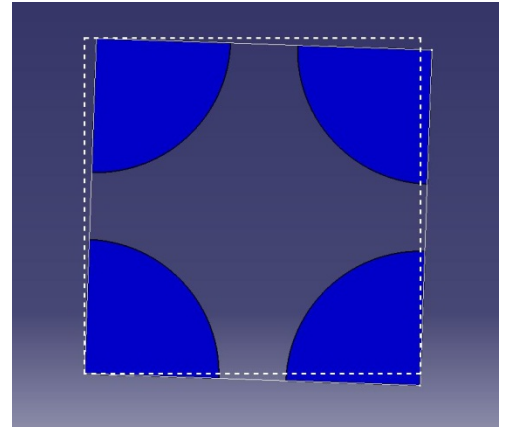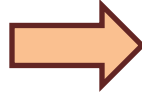
Figure 8: Deformed square RVE

4.   Python Scripting in Abaqus

The aim of this chapter is to introduce scripting in Abaqus. It will show bigger picture and introduce to idea of how a script can replace actions the user would otherwise perform in graphical user interface (GUI) Abaqus/CAE.

4.1   Introduction

When using the Abaqus/CAE graphical user interface (GUI) to create a model and to visualize the results, commands are issued internally by Abaqus/CAE after every operation. These commands reflect the geometry created along with the options and settings selected from each dialog box. The GUI generates commands in an object-oriented programming language called Python. The commands issued by the GUI are sent to the Abaqus/CAE kernel. The kernel interprets the commands and uses the options and settings to create an internal representation of model. The kernel is the brains behind Abaqus/CAE. The GUI is the interface between the user and the kernel.

4.2   Advantages of scripting

The Abaqus Scripting Interface allows bypassing the Abaqus/CAE GUI and communicate directly with the kernel. A file containing Abaqus Scripting Interface commands is called a script. Script are having following advantages:

- To automate repetitive tasks. For example, script can be created that executes when a user starts an Abaqus/CAE session. Such a script might be used to generate a library of standard materials. As a result, when the user enters the Property module, these materials will be available. Similarly, the script might be used to create remote queues for running analysis jobs, and these queues will be available in the Job module.

- To perform a parametric study. For example, script can be created that incrementally modifies the geometry of a part and analyzes the resulting model. The same script can read the resulting output databases, display the results, and generate annotated hard copies from each analysis.

- Create and modify the model databases and models that are created interactively when working with Abaqus/CAE. The Abaqus Scripting Interface is an application programming interface (API) to model databases and models.

- Access the data in an output database. For example, user may wish to do own post-processing of analysis results. User can write own data to an output database and use the Visualization module of Abaqus/CAE to view its contents.

The Abaqus Scripting Interface is an extension of the popular object-oriented language called Python. Any discussion of the Abaqus Scripting Interface applies equally to Python in general, and the Abaqus Scripting Interface uses the syntax and operators required by Python.

While one could write an entire script from scratch, ABAQUS provides several easier methods that auto-generate Python commands. When a model is created in CAE, two files are automatically created in the work directory. The replay files records every action that is performed in CAE including camera zoom/panning commands and also any mistakes that were made and then corrected. This file can be run to "replay" all the work that has been done on the model. The recover file records only the minimum necessary commands to recreate the model. If an error occurs and CAE closes without saving, the recover file can be run to recreate the model. When a model is saved, CAE uses the recover file to write a journal file. The journal file is a comprehensive script that shows all work saved on the model. Note that the recover file is deleted whenever the model is saved and all of its commands are transferred to the journal file. The recover and journal files provide an easy alternative to writing Python scripts from scratch. One can simply define a model in CAE and then save the Python commands from the recover or journal files as a separate script file[9].

An added advantage of a script is that user has entire simulation setup saved in the form of a small readable text file only a few kilobytes in size. It would redraw the part, apply the materials, loads, boundary conditions, create the steps, and even create and run the job if programmed to do so. It keeps things compact and easy to follow.

4.3   Python scripting for Periodic Boundary Condition

We used Macro Manager to record a sequence of the generated Abaqus Scripting Interface commands in a macro file. Using Macro manager tool, we obtained syntax for each command in Python and using this syntax entire script is build.

This script consists of recorded syntax for extracting edges of RVE and applying constraint equation to the nodes. The script is written from scratch for extracting nodes from edges of RVE and then separating each node from these extracted set of nodes. Several "for loop" are used to separate each node from set of nodes.

Because of this script application of Periodic Boundary Condition (PBC) is made very simple and time saving. Initially applying PBC to RVE was very time consuming and tedious task. The user has to select each and every node and assign constraint equation for two corresponding nodes. For a very fine mesh, the user need to apply this constraint equation more than 650 times since there were more than 650 nodes present on the each side of RVE.

Script made this task very simple. It takes few seconds to apply PBC. Even the mesh size is changed; it is very simple to apply PBC to modified mesh by re-running the script. So this CAE model is currently parametric in accordance with meshing. That means, because of this script it is very simple to adopt new mesh size for this RVE.

Script for PBC application is available in Appendix A for reference.

5.   Finite Element Modeling of RVE using ABAQUS

5.1   Introduction

To build RVE model Abaqus/CAE package is used. Abaqus/CAE is a complete Abaqus environment that provides a simple, consistent interface for creating, submitting, monitoring, and evaluating results from Abaqus/Standard and Abaqus/Explicit simulations. Abaqus/CAE is divided into modules, where each module defines a logical aspect of the modeling process; for example, defining the geometry, defining material properties, and generating a mesh. As we move from module to module, we can build the model from which Abaqus/CAE generates an input file that the user submits to the Abaqus/Standard or Abaqus/Explicit analysis product. The analysis product performs the analysis, sends information to Abaqus/CAE to allow monitoring the progress of the job, and generating an output database. Finally, the Visualization module of Abaqus/CAE (also licensed separately as Abaqus/Viewer) can be used to read the output database and view the results of analysis.

5.2   Modeling Approach

This model is built and analyzed in following steps:

5.2.1 Creating Part

Parts define the geometry of the individual components of model and, therefore, are the building blocks of an Abaqus/CAE model. It is possible to create parts that are native to Abaqus/CAE, or to import parts created by other applications either as a geometric representation or as a finite element mesh. For this case, the part is created in Abaqus/CAE.

Initially a square plate is created in Abaqus and it is partitioned into shape of a rectangular RVE as shown in Figure 9.

It is considered that RVE is homogenous and treated as one part.
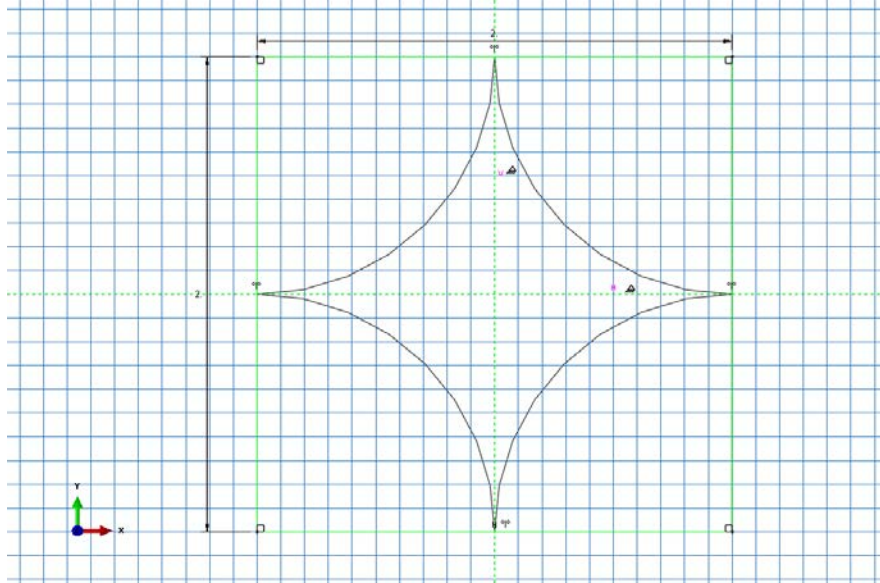
Figure 9: Final sketch of RVE

5.2.2 Creating material

RVE is made of steel and assumed to be linear elastic with Young's modulus of 30 X 10^6 psi and Poisson's ratio of 0.3. Thus, I have created a single linear elastic material with these properties.

5.2.3 Defining and assigning section properties

Properties of part can be defined through sections. A solid homogeneous section is created and this section is assigned to RVE.

5.2.4 Defining the assembly

Each part is oriented in its own coordinate system and is independent of the other parts in the model. Although a model may contain many parts, it contains only one assembly. It is possible to define the geometry of the assembly by creating instances of a part and then positioning the instances relative to each other in a global coordinate system. An instance can be classified as either independent or dependent. Independent part instances are meshed individually, while the mesh of a dependent part instance is associated with the mesh of the original part. By default, part instances are dependent and so for this part.

15

5.2.5 Configuring analysis

For this report work, behavior of RVE is studied under various loading condition: compression, shear and dynamic explicit loading. For the different loading condition, the analyses are configured differently. Abaqus/CAE generates the initial step automatically, but it is needed to create the analysis step. For compression and shear loading, single static, a general step is created. For dynamic analysis this step is replaced by a Dynamic\explicit step.

For Dynamic\explicit analysis, the time period is set to 0.2 ms and for other analyses, the time period is set to 1 second by default.

Finite element analyses can create very large amounts of output. Abaqus allows the user, to control and manage this output so that only data required to interpret the results of simulation are produced. By default, Abaqus/CAE writes the results of the analysis to the output database (.odb) file. When we create a step, Abaqus/CAE generates a default output request for the step. Field Output Requests Manager is used to request output of variables that should be written at relatively low frequencies to the output database from the entire model or from a large portion of the model. History Output Requests Manager is used to request output of variables that should be written to the output database at a high frequency from a small portion of the model; for example, the displacement of a single node. For this example I have examined the output requests to the .odb file and accepted the default configuration.

5.2.6 Applying boundary conditions and loads to the model

Prescribed conditions, such as loads and boundary conditions, are step dependent, which means that user must specify the step or steps in which they become active. Now steps are already defined in the analysis, this section defines prescribed conditions.

In structural analyses, boundary conditions are applied to those regions of the model where the displacements and/or rotations are known. Such regions may be constrained to remain fixed (have zero displacement and/or rotation) during the simulation or may have specified, nonzero displacements and/or rotations.

In Abaqus the term "load" (as in the Load module in Abaqus/CAE) generally refers to anything that induces a change in the response of a structure from its initial state, including:

- concentrated forces
- pressures

- non-zero boundary conditions

- body loads

- temperature (with thermal expansion of the material defined).

Sometimes the term load is used to refer specifically to force-type quantities (as in the Load Manager of the Load module); for example, concentrated forces, pressures, and body loads but not boundary conditions or temperature. The intended meaning of the term should be clear from the context of the discussion. The load will always be applied over some finite area. However, if the area being loaded is small, it is an appropriate idealization to treat the load as a concentrated load.

For this study, small deformation theory is considered. So, according to this theory, 2% deformation in compression and 0.1% deformation is assumed to be significant and same loading is applied to RVE respectively.

In this simulation three various types of loading were applied to RVE.

1. Compression: Compression loading is applied using Displacement\Rotation type Boundary Condition. RVE is compressed by 2%. This BC is applied during "Applied Load" step of Abaqus model. Compression loading is shown in Figure 10. In addition to Displacement boundary condition, bottom edges of RVE are applied with "Encastre Boundary Condition" which ensures that there is fixed edges at bottom of RVE.
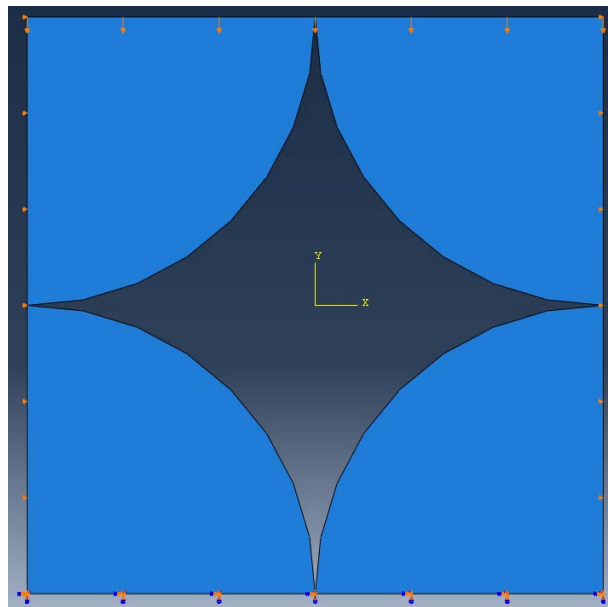
Figure 10: Boundary conditions for compression loading

In addition to Encastre and Displacement boundary condition for top and bottom edge respectively, side edges of RVE are also constrained in X-direction. This additional boundary condition ensures that the loading will be pure compression loading.

2) Shear Loading: Figure 11 describes boundary condition of RVE under pure shear loading. Similar to Compression loading, displacement boundary condition is applied to RVE. 0.1 % displacement is imposed on RVE under shear loading. Also bottom edges of RVE are fixed under shear loading.
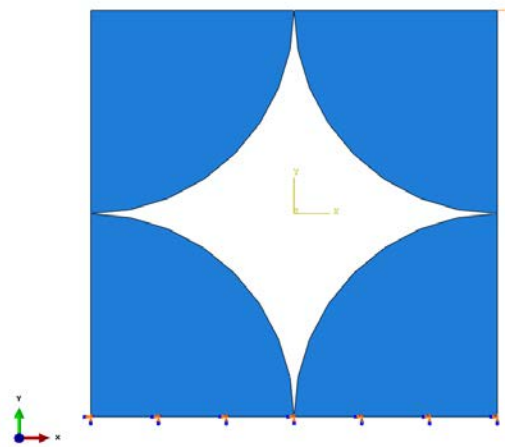

Figure 11: Boundary condition for Shear loading

3) Dynamic loading: Abaqus/Explicit solver can be used widely to solve High-speed dynamic events, Complex contact problems, Complex post-buckling problems, highly nonlinear quasi-static problems and Materials with degradation and failure.

For dynamic loading condition, Abaqus/Explicit is used. This is because; Abaqus/Explicit can readily analyze problems involving complex contact interaction between many independent bodies. Abaqus/Explicit is particularly well-suited for analyzing the transient dynamic response of structures that are subject to impact loads and subsequently undergo complex contact interaction within the structure. Contact conditions and other extremely discontinuous events are readily formulated in the explicit method and can be enforced on a node-by-node basis without iteration. The nodal accelerations can be adjusted to balance the external and internal forces during contact. The most striking feature of the explicit method is the absence of a global tangent

stiffness matrix, which is required with implicit methods. Since the state of the model is advanced explicitly, iterations and tolerances are not required.

The load of 0.5 X 10^6 psi is applied at time steps shown in Figure 12. Other boundary conditions for side edges and for bottom edges of RVE are maintained the same as in Compression loading.
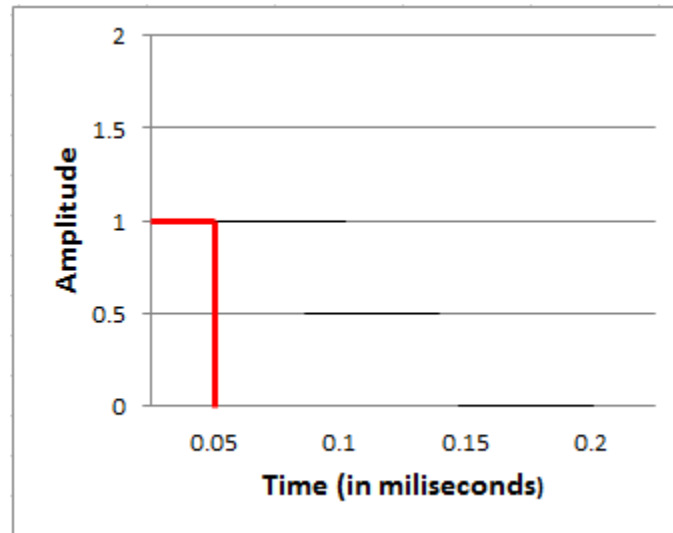
Figure 12: Amplitude for blast loading condition

5.2.7 Contact in Abaqus\CAE

Contact simulations in Abaqus/Standard can either be surface based or contact element based. Contact simulations in Abaqus/Explicit are surface based only.

Surface-based contact can utilize either the general ("automatic") contact algorithm or the contact pair algorithm. The general contact algorithm allows for a highly automated contact definition, where contact is based on an automatically generated all-inclusive surface definition. Conversely, the contact pair algorithm requires explicitly pair surfaces that may potentially come into contact. Both algorithms require specification of contact properties between surfaces (for example, friction).

This analysis used general contact method between surfaces of RVE. The contact interaction domain, contact properties, and surface attributes are specified independently for general contact, offering a more flexible way to add detail incrementally to a model. The simple

interface for specifying general contact allows for a highly automated contact definition; however, it is also possible to define contact with the general contact interface to mimic traditional contact pairs. Conversely, specifying self-contact of a surface spanning multiple bodies with the contact pair user interface (if the surface-to-surface formulation is used) mimics the highly automated approach often used for general contact.

When surfaces are in contact, they usually transmit shear as well as normal forces across their interface. Thus, the analysis may need to take frictional forces, which resist the relative sliding of the surfaces, into account. So, for this report analysis is performed with different coefficient of friction for compression loading (varied coefficient of friction value from 0.1 to 0.7). It is considered that, for the roughest surface, co-efficient of friction will be maximum 0.7.

5.2.8 Meshing the model

The Mesh module allows generating meshes on parts and assemblies created within Abaqus/CAE. As with creating parts and assemblies, the process of assigning mesh attributes to the mode such as seeds, mesh techniques, and element types—is feature based. As a result user can modify the parameters that define a part or an assembly, and the mesh attributes within the mesh module, that are regenerated, are automatically specified.

CPS8 element type is used for RVE. More details about this element are provided in next session.

Eight-node plane stress element (CPS8 and CPS8R)

The eight node plane stress element is a general purpose plane stress element. It is actually a special case of shell element: the structure is assumed to have a symmetry plane parallel to the x-y plane and the loading only acts in-plane. In general, the z-coordinates are zero. Just like in the case of the shell element, the plane stress element is expanded into a C3D20 (quadratic brick element) or C3D20R (quadratic brick element with reduced integration) element. From the above premises the following conclusions can be drawn:

- The displacement in z-direction of the mid-plane is zero.
- The displacements perpendicular to the z-direction of nodes not in the midplane is identical to the displacements of the corresponding nodes in the midplane.
- The normal is by default (0,0,1)

The use of plane stress elements can also lead to knots, namely, if the thickness varies in a discontinuous way, or if plane stress elements are combined with other 1D or 2D elements such as axisymmetric elements. The connection with the plane stress elements, however, is modeled as a hinge.

Distributed loading in plane stress elements is different from shell distributed loading: for the plane stress element it is in-plane, for the shell element it is out-of-plane. The number indicates the face as defined in Figure 13.
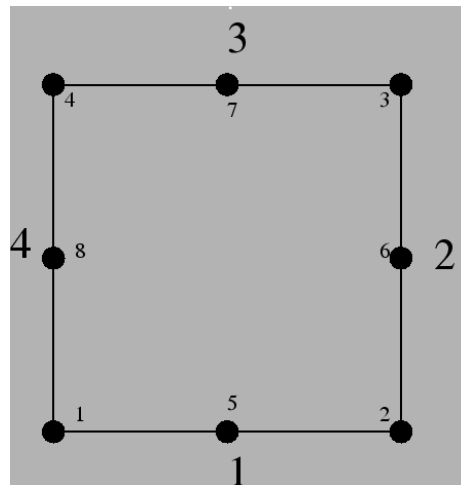
Figure 13: CPS8 Element type for meshing[11]

5.2.9 Creating an analysis job

Once all of the tasks involved in defining a model (such as defining the geometry of the model, assigning section properties, and defining contact) are done, then Job module can be used to analyze model. The Job module allows creating a job, to submit it for analysis, and to monitor its progress. Multiple models can be created and monitored simultaneously.

In addition, there is an option of creating only the analysis input file for model. This option allows viewing and editing the input file before submitting it for analysis. For an Abaqus/Standard or Abaqus/Explicit analysis, we can also view and edit the analysis keywords for a model by selecting Model⟶Edit Keywords⟶model name from the main menu bar.

Data check analysis is performed before running analysis to check for errors in model file.

5.2.10 Post-processing

Graphical postprocessing is important because of the great volume of data created during a simulation. The Visualization module of Abaqus/CAE (also licensed separately as Abaqus/Viewer) allows viewing the results graphically using a variety of methods, including deformed shape plots, contour plots, vector plots, animations, and X–Y plots. In addition, it allows creating tabular reports of the output data.

6. Results and Discussion

6.1 Compression loading

After successfully running the analysis for compression loading, following contour plot is obtained. From the contour plot, as shown in Figure 14, it has been observed that maximum stress experienced in RVE is 2.024 X 10^6 psi for seed size of 0.05.
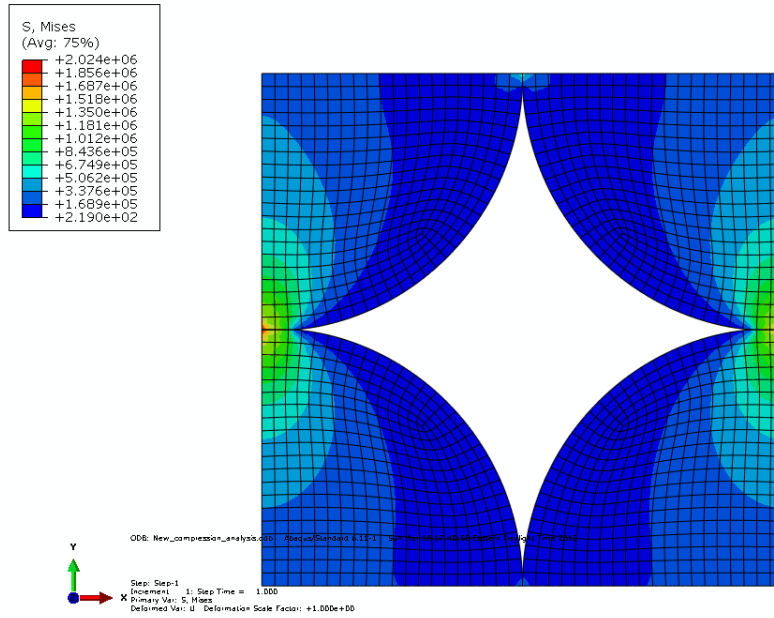
Figure 14: Contour plot for compression loading

According to contour plot, the side edges are deformed only in Y-direction and not in X-direction. This ensures that pure compression of RVE is obtained. Since Periodic boundary condition is maintained properly we can conclude that this RVE represents entire volume of solid material.

## 6.2 Mesh Sensitivity analysis

In order to make sure that the stresses are independent of element size, the mesh sensitivity study has been carried out. The mesh sensitivity study was carried for three element sizes as listed in Table 1. Only Von- mises stresses have been considered for the mesh sensitivity study. The mesh has to be refined globally and / or locally to improve the stress prediction in the critical regions.

Mesh type with Element number	Von mises stress (psi)
1500 (Coarse)	2.024×10^6
6076 (Fine)	2.034×10^6
24444 (Finer)	2.040×10^6
97500 (Finest)	2.045×10^6

Table 1: Mesh sensitivity analysis

From the above table it is observed that, only change of 0.3% in value of Von mises stress is observed. So it can be concluded that finest mesh type has reached convergent solution. So this mesh size is used in further work. Similar pattern of stress distribution is observed in all the contour plots with variation in stress values (contour plot for each mesh type is provided in Appendix B).

For finest mesh, the stress-strain curve is studied in order to validate the results. Plot of stress vs strain, as shown in Figure 15, is obtained. According to this plot, stress varies linearly with the true strain value. So RVE has shown linear elastic material behavior.
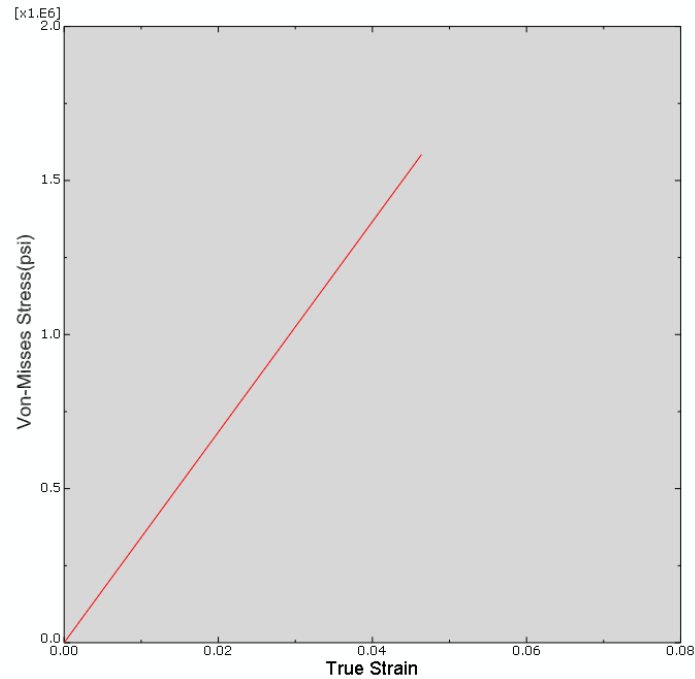

Figure 15: Stress Vs Strain curve for RVE under compression loading

6.3   Variation in Coefficient of Friction of RVE

Coefficient of friction is changed from interaction module of Abaqus/CAE package from 0.1 to 0.7. For these various coefficient of friction values, maximum stress and total reaction force values are recorded. Young's modulus value is calculated by below formula

$$Young's\ Modulus = \frac{\left(\dfrac{Total\ reaction\ forces}{area\ of\ RVE}\right)}{Strain} = 18.10\ X\ 10^6\ psi$$

Young's modulus value is calculated in each case of coefficient of friction value. It has been observed that the young's modulus value is same in each case. The plot shown in Figure 16 for Young's modulus vs Coefficient of friction, suggests that coefficient of friction value is not affecting Young's modulus value. Since there is point contact between two surfaces of RVE, it seems correct that coefficient of friction has little impact on Young's modulus values.
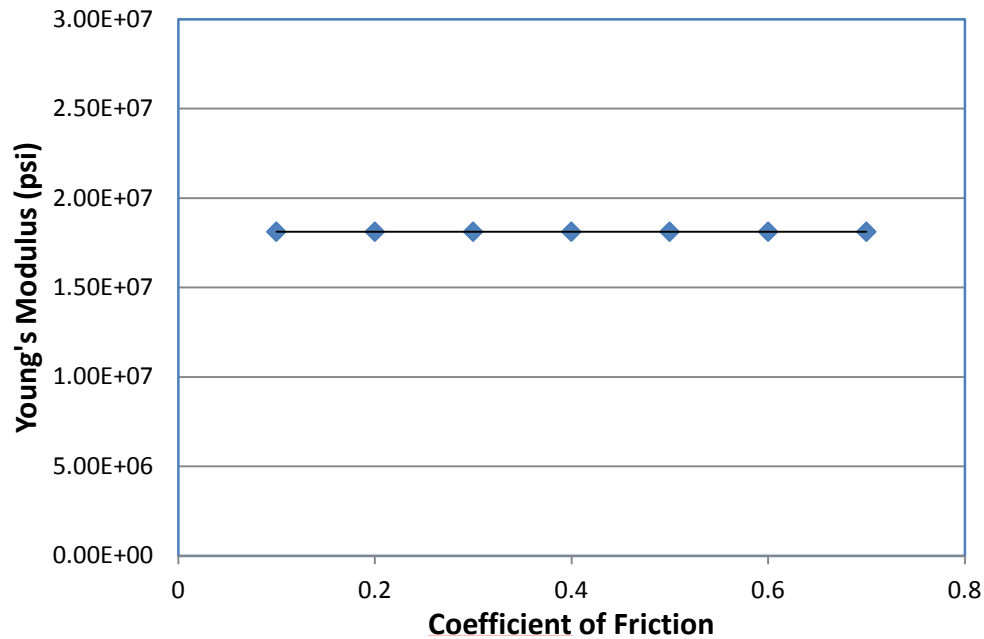
Figure 16: Young's modulus vs Coefficient of Friction

6.4 Shear Loading

After applying shear load with 0.1 % deformation, stress contour plot as shown in Figure 17 is obtained. According to this contour plot it has been observed that stress of 1.98 X 10^5 psi experienced in contact area. Because shear load applied is very small over most of the area of the RVE, stress value is near zero. However, for most of the cross section, stress is experienced at contact area and at the corners due to periodic boundary condition.
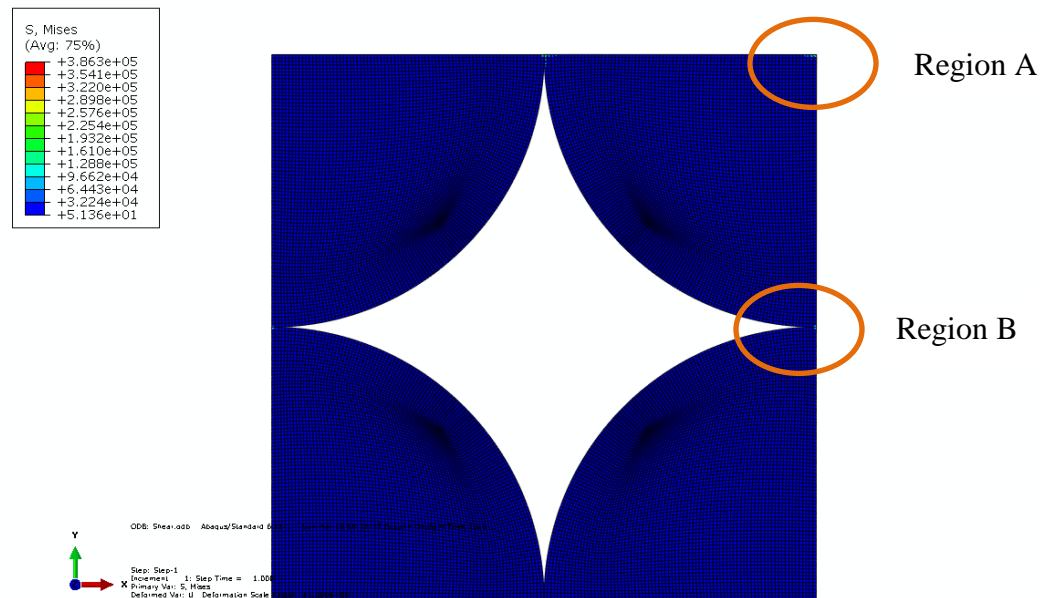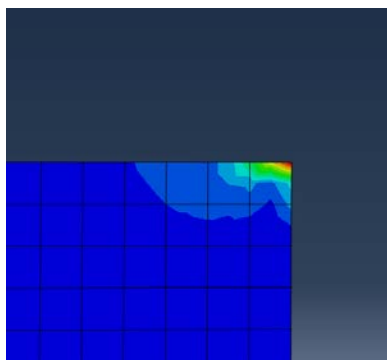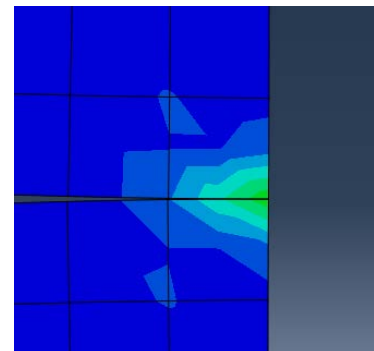

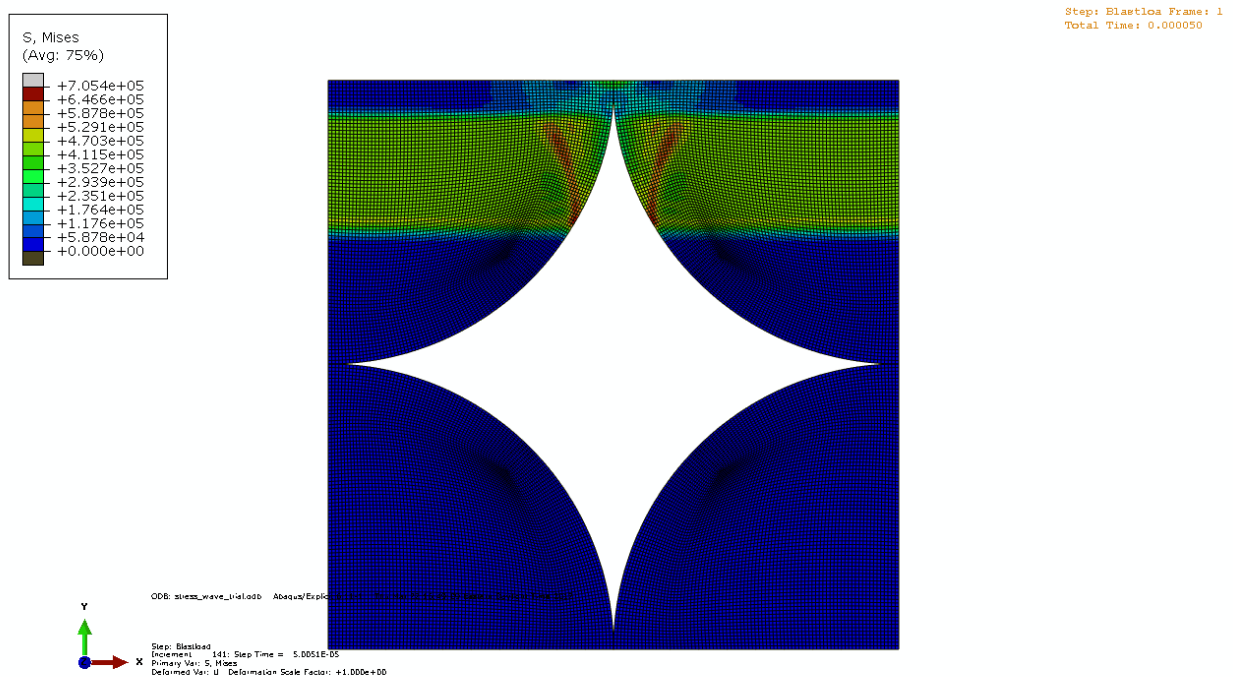**Figure 17: Stress contour plot of RVE under shear load**

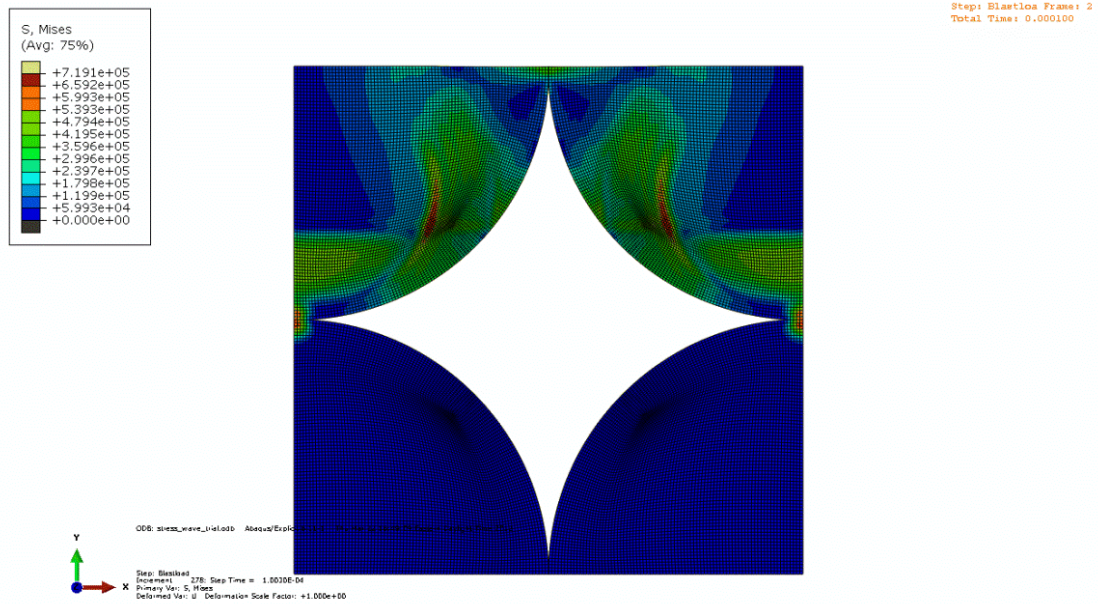**Detailed view of region A**   **Detailed view of region B**
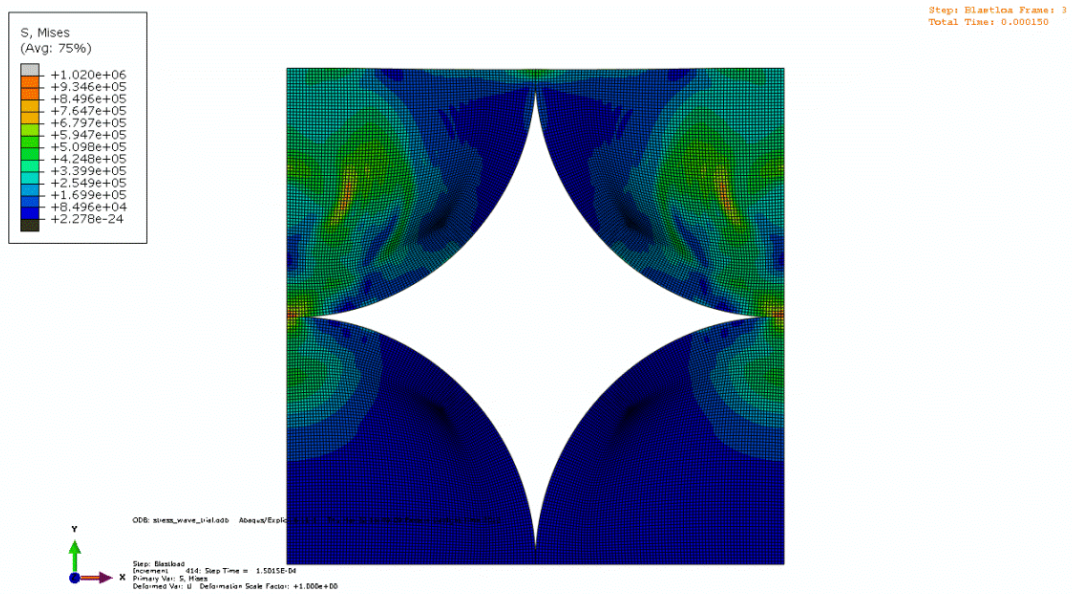
6.5   RVE under dynamic loading

As described in section 5.2, blast load is applied to RVE. The analysis is performed using Abaqus\Explicit package. Stress contour plot for five time steps are recorded and the behavior of RVE under blast load is studied. For each time step, mises stress contour plot is shown in below Figure 18-Figure 21. According to this plot it has been observed that stress wave mainly propagates in upper region of RVE and part of stress wave also get propagated in the lower region of RVE through contact surfaces.

Figure 18: Mises contour plot at t= 0.00005 sec

S, Mises
(Avg: 75%)

+7.191e+05
+6.592e+05
+5.993e+05
+5.393e+05
+4.794e+05
+4.195e+05
+3.596e+05
+2.996e+05
+2.397e+05
+1.798e+05
+1.199e+05
+5.993e+04
+0.000e+00

Figure 19: Mises contour plot at t= 0.00010sec

S, Mises
(Avg: 75%)

+1.020e+06
+9.346e+05
+8.496e+05
+7.647e+05
+6.797e+05
+5.947e+05
+5.098e+05
+4.248e+05
+3.399e+05
+2.549e+05
+1.699e+05
+8.496e+04
+2.278e-24

Figure 20: Mises contour plot at t= 0.000150sec

28
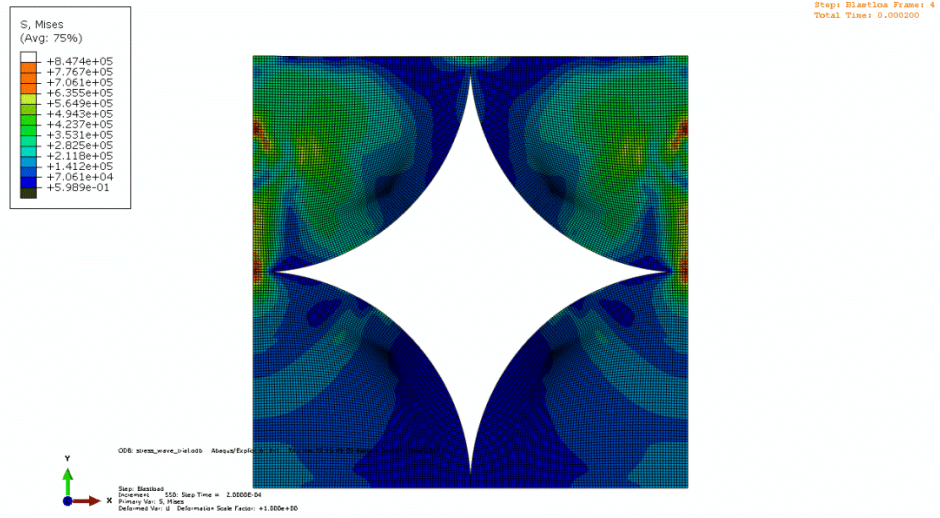
Figure 21: Mises contour plot at t= 0.0002sec

7.   Conclusion

Various array of RVE structure like square pack and hexagonal array are studied. Square array is chosen to study behavior of RVE. Periodic boundary condition is studied and applied to RVE using python scripting. It is found that python script saves more than an hour time for periodic boundary condition application to the RVE model.

Finite element model is built using Abaqus\CAE software package. Various modules- sketcher, property, step, assembly, interaction, load, mesh and visualization- from Abaqus\CAE package are studied and used to build finite element model of RVE.

Three different models under shear, compression and dynamic loading are submitted for post-processing of Abaqus\CAE solver and results are studied. Mesh sensitivity analysis is carried out for compression loading. After reaching to convergent solution, very fine mesh, with 24444 elements, is chosen for shear and dynamic loading. Behavior of RVE under different coefficient of friction between surfaces is also studied and found that coefficient of friction does not affect Young's modulus of RVE. Maximum mises stresses are observed for shear loading and for dynamic loading. For dynamic loading stress contour plot is taken for each time step of analysis. It is observed that wave propagates through upper region of RVE uniformly but in lower region, due to point contact between two regions, it propagates randomly towards fixed end of RVE.

In future, this model can be used to study behavior of material under complex loading like in the field of modal dynamics. Also non-linearity of model can be studied, in detail, using this model.

8.   References

[1] O. van der Sluis, P.J.G. Schreurs, W.A.M. Brekelmans, H.E.H. Meijer, Overall behaviour of heterogeneous elastoviscoplastic materials: effect of microstructural modeling, Mechanics of Materials 32 (2000) 449-462

[2] J.M. Tyrus, M. Gosz, E. DeSantiago, 2007, A local finite element implementation for imposing periodic boundary conditions on composite micromechanical models, International Journal of Solids and Structures 44, 2972–2989

[3] Gusev, A., 1997, Representative volume element size for elastic composites: A numerical study. Journal of the Mechanics and Physics of Solids 45, 1449–1459.

[4] Drugan, W., Willis, J., 1996, A micromechanics-based nonlocal constitutive equation and estimates of representative volume element size for elastic composites. Journal of the Mechanics and Physics of Solids 44, 497–524

[5] Hill, R., 1963, Elastic properties of reinforced solids: some theoretical principles. J. Mech. Phys. Solids 11, 357-372

[6] Terada K., Hori M., Kyoya T., Kikuchi N., 2000, Simulation of the multi-scale convergence in computational homogenization approaches. Int. J. Solids Struct. 37, 2285-2311

[7] Anthoine A., 1995, Derivation of the in-plane elastic characteristics of masonry through homogenization theory. Int. J. Solids Struct. 32 (2), 137-163

[8] Smit R., Brekelmans, W., Meijer, H, 1999, Prediction of the large-strain mechanical response of heterogeneous polymer systems: local and global deformation behavior of a representative volume element of voided polycarbonate. J. Mech. Phys. Solids 47, 201-221

[9] Abaqus Scripting User's Manual

[10] Abaqus/CAE User's Manual

[11] http://web.mit.edu/calculix_v2.0/CalculiX/ccx_2.0/doc/ccx/node26.html

[12] http://imechanica.org/node/9135

Appendix A: Python Script for Periodic Boundary Condition

Script for Periodic Boundary Condition:

```
import section

import regionToolset

import displayGroupMdbToolset as dgm

import part

import material

import assembly

import step

import interaction

import load

import mesh

import job

import sketch

import visualization

import xyPlot

import displayGroupOdbToolset as dgo

import connectorBehavior

# Creating nodes set at the edges

mdb.models['Model-1'].parts['Part-1'].Set(edges=

mdb.models['Model-1'].parts['Part-1'].edges.findAt((((-0.5,0.25,0),),), name='leftUpperEdge1')

mdb.models['Model-1'].parts['Part-1'].Set(name='LeftUpperNodes1',
nodes=mdb.models['Model-1'].parts['Part-1'].sets['leftUpperEdge1'].nodes)

mdb.models['Model-1'].parts['Part-1'].Set(edges=

mdb.models['Model-1'].parts['Part-1'].edges.findAt((((0.5,0.25,0),),),
name='RightUpperEdge1')
```

```
mdb.models['Model-1'].parts['Part-1'].Set(name='RightUpperNodes1',
nodes=mdb.models['Model-1'].parts['Part-1'].sets['RightUpperEdge1'].nodes)

mdb.models['Model-1'].parts['Part-1'].Set(edges=

mdb.models['Model-1'].parts['Part-1'].edges.findAt(((-0.5,-0.25,0),),),
name='LeftLowerEdge1')

mdb.models['Model-1'].parts['Part-1'].Set(name='LeftLowerNodes1',
nodes=mdb.models['Model-1'].parts['Part-1'].sets['LeftLowerEdge1'].nodes)

mdb.models['Model-1'].parts['Part-1'].Set(edges=

mdb.models['Model-1'].parts['Part-1'].edges.findAt(((0.5,-0.25,0),),),
name='RightLowerEdge1')

mdb.models['Model-1'].parts['Part-1'].Set(name='RightLowerNodes1',
nodes=mdb.models['Model-1'].parts['Part-1'].sets['RightLowerEdge1'].nodes)

# Creating individual nodes

# Nodes at left upper region[12]

a=[]

for i in mdb.models['Model-1'].parts['Part-1'].sets['LeftUpperNodes1'].nodes:

    a=a+[(i.coordinates[1],i.label)]

a.sort()

nodeno1=1

for i in a:

    mdb.models['Model-1'].parts['Part-1'].Set(name='LeftNode-'+str(nodeno1), nodes=

        mdb.models['Model-1'].parts['Part-1'].nodes[(i[1]-1):(i[1])])

    nodeno1=nodeno1+1

# Nodes at left lower region

b=[]

for i in mdb.models['Model-1'].parts['Part-3'].sets['LeftLowerNodes1'].nodes:

    b=b+[(i.coordinates[1],i.label)]
```

```
b.sort()

nodeno3=1

for i in b:

    mdb.models['Model-1'].parts['Part-3'].Set(name='LeftNode-'+str(nodeno3), nodes=

        mdb.models['Model-1'].parts['Part-3'].nodes[(i[1]-1):(i[1])])

    nodeno3=nodeno3+1

# Nodes at right upper region

c=[]

for i in mdb.models['Model-1'].parts['Part-2'].sets['RightUpperNodes1'].nodes:

    c=c+[(i.coordinates[1],i.label)]

c.sort()

nodeno2=1

for i in c:

    mdb.models['Model-1'].parts['Part-2'].Set(name='RightNode-'+str(nodeno2), nodes=

        mdb.models['Model-1'].parts['Part-2'].nodes[(i[1]-1):(i[1])])

    nodeno2=nodeno2+1

# Nodes at right lower region

d=[]

for i in mdb.models['Model-1'].parts['Part-4'].sets['RightLowerNodes1'].nodes:

    d=d+[(i.coordinates[1],i.label)]

d.sort()

nodeno4=1

for i in d:

    mdb.models['Model-1'].parts['Part-4'].Set(name='RightNode-'+str(nodeno4), nodes=

        mdb.models['Model-1'].parts['Part-4'].nodes[(i[1]-1):(i[1])])
```

```python
    nodeno4=nodeno4+1

#Applying constraint equation to each node for PBC

for num1 in range(1,nodeno1):

    leftnodename = 'Part-1-1.LeftNode-%d'%(num1)

    rightnodename = 'Part-2-1.RightNode-%d'%(num1)

    mdb.models['Model-1'].Equation(name="Constraint-%d"%(num1), terms=((1.0,

    leftnodename, 1), (-1.0, rightnodename, 1)))

for num2 in range(1,nodeno2):

    leftnodename = 'Part-3-1.LeftNode-%d'%(num2)

    rightnodename = 'Part-4-1.RightNode-%d'%(num2)

    mdb.models['Model-1'].Equation(name="Constraint-%d"%(nodeno1), terms=((1.0,

    leftnodename, 1), (-1.0, rightnodename, 1)))

    nodeno1=nodeno1+1
```
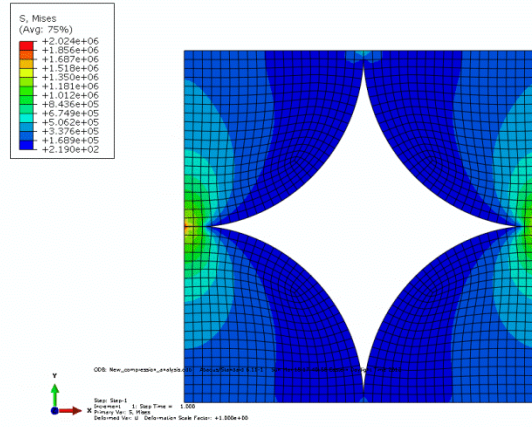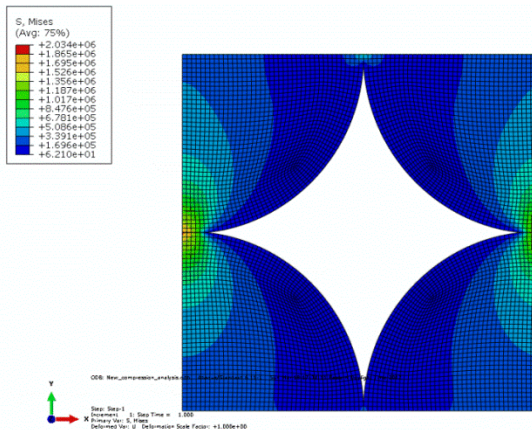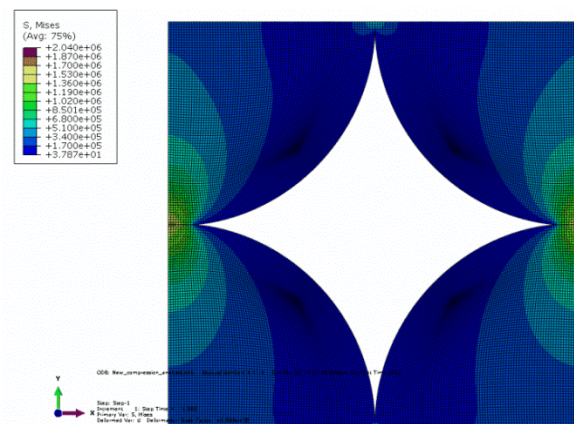
Appendix B: Mesh Sensitivity Analysis - Contour plot
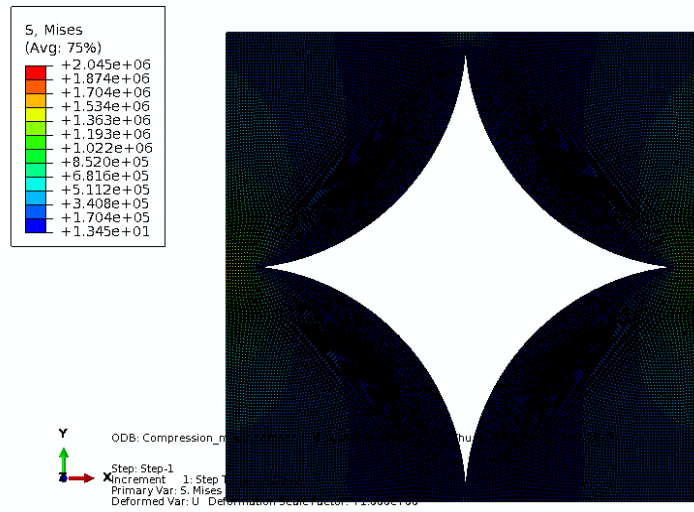
Contour plot for 1500 elements

Contour plot for 6076 elements

Contour plot for 24444 elements

Contour plot for 97500 elements