Micromechanics Plugin

For Abaqus/CAE

3DEXPERIENCE®

Version 1.18 – 4/26/2022

DASSAULT SYSTEMES | The 3DEXPERIENCE® Company

Table of contents

Overview

Abaqus provides powerful functionality for simulating the behavior of material systems that consist of distinct constituents and/or microstructure. The mean-field homogenization material type in Abaqus allows users to define a material system as a combination of constituents and uses analytical and quasi-analytical approaches to predict the aggregate (homogenized) response of the material system at larger scales. Additionally, a finite element model of a representative volume element (an FE-RVE) of a material's microstructure can be developed to study material systems in a way that is not subject to the assumptions inherent in the analytical approaches used in mean-field homogenization. Certain homogenized properties (e.g. elasticity, thermal expansion, and conductivity) can be directly obtained from FE-RVE models, and in other cases the homogenized response of an FE-RVE can be used in place of coupon test data to define phenomenological material models for use at larger scales (e.g. plasticity, viscoelasticity, and damage models). Additionally, helpful insight into a material system's behavior can be gained by studying the evolution of local fields in an FE-RVE that is experiencing a given far-field loading history.

This plugin provides helpful functionality for setting up and post-processing FE-RVE models. Model setup includes the imposition of far-field loading through periodic or non-periodic boundary conditions, and setting up the loads and analysis steps necessary to perform homogenization of properties or imposition of a far-field load history. Post processing includes determining the homogenized properties of the FE-RVE from the completed analysis as well as performing averaging and statistical analysis of the fields in the whole RVE and within individual RVE constituents.

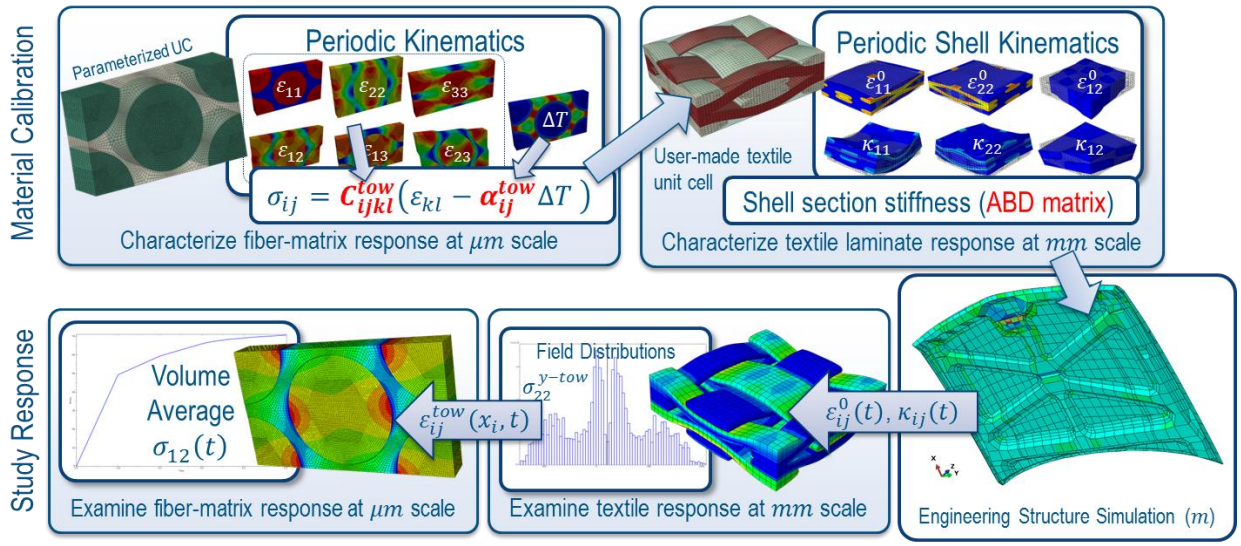


Figure 1: Example workflow facilitated by plugin

There are a broad variety of applications and workflows that these capabilities can be applied to. The preceding figure gives an example of a multi-scale workflow that can be facilitated by the plugin. In the first step, a fiber-matrix model is used to predict the transversely isotropic elastic response of a unidirectional composite material. This material response is then used for the tows in a textile unit cell model. This textile unit cell model is used to obtain the shell section behavior for a thin textile laminate. Note that this shell section stiffness is based directly on the response of the textile unit cell model with shell-like boundary conditions applied and does not come from laminated plate theory (which assumes that properties do not vary in the plane of the shell – an assumption which may be incongruous with thin textile laminates or honeycomb/lattice-core panels). This shell section stiffness can then be used to define a shell general section definition in an engineering structure.

The engineering structure can then be analyzed as normal. Once that analysis is complete, the shell deformation at some point of interest in the engineering structure can be used to drive the textile unit cell analysis. This permits the stresses in and behavior of the textile laminate to be examined in detail, permitting an analyst to predict whether any phenomena might occur such as interfacial failure between tows, matrix cracking in a tow, or plasticity in the neat matrix pocket which may not be accounted for in the material response used in the engineering analysis. Additionally, the strain history from a location in the tow can be used to drive a fiber-matrix unit cell, permitting more detailed analysis of the tow microstructure and prediction of phenomena within the tow such as matrix cracking, fiber failure, or plasticity, all based on the predicted response of the engineering structure.

Release Notes

V1.0 – 12/6/2016

- Original Release

V1.1 – 3/22/2017

- Handles coincident nodes with matching attached facets
- Enables user-defined environment variables for tolerance/etc.
- Adds simple ellipsoid array option
- Adds constituent strain concentration tensor output option
- Gives correct reference-point-derived average Cauchy stress under large deformation
- Improved diagnostic messages for problems with node pairing
- Fixes post-processing crash associated with sections assigned using internal element sets

V1.11 – 4/11/2017

- User-defined driven/ignored sets for Taylor-type BCs
- Separate tolerance for determining periodic node positioning
- Adds user-defined function handles for pairing nodes and calculating model volume/area
- Bugs fixed –
    - Incorrectly identified maximum coordinate for RVE bounding box when maximum was negative
- Plugin issued error when post-processing heat transfer model

V1.12 – 5/12/2017

- Bugs fixed –
    - Copied facets for non-periodic meshes now have correct element type assigned in Abaqus 2016
    - Fixed issue in which the order of copied facets doesn't match the order of the original facets

V1.13 – 11/2/2017

- Plugin now handles part names containing spaces
- Bugs fixed –
    - Anisotropic material definitions generated by the plugin had constants output in incorrect order
    - Jobs associated with deleted models could cause the plugin to crash

V1.14 – 12/1/2017

- Plugin periodically constrains rotations for beams and shells in mechanical and solid-to-shell scenario analyses.
- Adds library functionality to generate body-centered lattice RVEs

V1.15 – 12/11/2017

- Bugs fixed –
    - Plugin issued error for RVEs that did not contain rotational DOFs

V1.16 – 1/17/2020

- Add custom lattice generation tool (accessible through python scripting)
- Improve performance of facet copying
- Add checks for compatible Abaqus/CAE version with diagnostic messages
- Use native unsorted nodeset and equation objects in Abaqus/CAE for mechanical loading models
- Improved error messaging in situations where nodes on RVE boundaries have imprecise coordinates
- Bugs fixed –
    - Correct calculation of homogenized specific heat by using mass average instead of volume average
    - Improved handling of non-continuum elements in volume averaging routines
    - Fixed post-processing compatibility issue with Abaqus/CAE 2020

V1.17 – 12/9/2021

- Improved error messaging
- Output of full non-symmetric 1st Piola-Kirchoff stress

V1.18 – 4/26/2022

- Fixes issue with periodic boundary conditions for non-periodic meshes in with Abaqus/CAE version 2022
- Fixes issue that caused the 13 and 23 components of far-field stress to be swapped for periodic- and uniform surface gradient-type models (e.g., specifying 13 component of stress caused application of 23 component and vice-versa).

Installation

1. Obtaining the Plugin

The plugin is available on the Dassault Systemes Knowledge Base in QA article QA00000046185.

2. Installing the Plugin

Extract the contents of the zip folder into a subdirectory of the abaqus_plugins folder, which can either be located in the home directory (in which case the plugin will be available whenever Abaqus is run) or within the current working directory of your Abaqus/CAE session.

3. Compatibility

The FE-RVE features of the plugin are designed to be compatible with Abaqus/CAE 2016 and later. Earlier versions of Abaqus/CAE will not function correctly.

1. FE-RVE

1.1. Scenarios

The plugin can be used to perform unit cell analyses for several different types of FE-RVE models.

1.1.1. Mechanical

The mechanical scenario is for performing stress-displacement analysis on unit cells that are embedded in an effectively infinite medium in all three coordinate directions. Homogenization can be performed with these models to obtain effective elastic properties, thermal expansion, and density.

1.1.2. Thermal

The thermal scenario is for performing steady-state thermal transport analysis on unit cells that are embedded in an effectively infinite medium in all three coordinate directions. Homogenization can be performed with these models to obtain effective conductivity and heat capacity.

1.1.3. Coupled Temperature-Displacement

The coupled temperature-displacement scenario is for performing fully coupled steady-state temperature-displacement analysis on unit cells that are embedded in an effectively infinite medium in all three coordinate directions. Homogenization can be performed with these models to obtain effective elastic properties, thermal conductivity, and the fully coupled 9x9 effective constitutive matrix relating strain and temperature gradient with stress and heat flux.

1.1.4. Solid-To-Shell

The solid-to-shell scenario is designed to perform stress-displacement analysis on unit cells that form a plate-like structures undergoing bending and membrane loading and which have traction free surfaces on the top and bottom z surfaces. Homogenization can be performed with these models to obtain the effective shell section stiffness matrix (commonly referred to as the ABD matrix) of the unit cell for use in defining an equivalent shell section. This is particularly useful for thin material systems that exhibit periodic in-plane variation of their structure (which is incongruous with the assumptions of laminated plate theory). Examples of such structures are honeycomb or corrugated core sandwich structures and thin textile structures.

1.2. Creating an FE-RVE Model

1.2.1. Automatic Specification of FE-RVE boundaries and sets

If the RVE is a hexahedron with flat surfaces that are aligned with the global coordinate axes, then the plugin will automatically create the node sets needed to impose the requested boundary conditions. Meshes with nodal coordinates that have roundoff may cause problems which can sometimes be alleviated by relaxing RELATIVE_DIMENSION_TOLERANCE as described in Section **5 User Variables and Functions**.

1.2.2. User-Defined Face Pairing for Periodic Boundary Conditions

If an RVE is not a rectangle with axis-aligned faces, it may be necessary for the user to define the pairs of faces manually for imposing periodic boundary conditions. The plugin searches the the model's assembly for the existence of node sets with names of the form *RVE_USER_PAIR_<Pair#>_<M/S>,* where *<Pair#>* is an integer denoting the number of the pair, and *<M/S>* denotes whether this should be the master (*M*) or slave (*S*) in the constraints. If found, these sets will be used in lieu of the automatically-generated pairs of surface nodes. Note that these sets must be defined at the assembly level, not at the part level.

NOTE – For unit-cells that are not hexahedra with flat, axis-aligned faces, the plugin cannot calculate the correct unit cell volume from the unit-cell bounding box, which will lead to far-field stress/flux values being calculated incorrectly. This issue can be addressed using user-defined functions described in Section 5 *User Variables and Functions****.***

1.2.3. User-Defined Nodeset for Uniform Surface Gradient (Taylor) Boundary Conditions

If a user is specifying Uniform Surface Gradient boundary conditions (described in Section **1.3.2 Uniform Surface Gradient (Taylor)**), the user can manually define the nodes that will be constrained to the far-field gradient by putting these nodes into an assembly-level node set named *RVE_USER_TAYLOR_DRIVEN*. When the plugin encounters a node set with this name, it will be used rather automatically finding nodes using the bounding box of the FE-RVE model. This set must be defined at the assembly level rather than at the part level.

NOTE – For unit-cells that are not hexahedra with flat, axis-aligned faces, the plugin cannot calculate the correct unit cell volume from the unit-cell bounding box, which will lead to far-field stress/flux values being calculated incorrectly. This issue can be addressed using user-defined functions described in Section 5 *User Variables and Functions****.***

1.2.4. RVEs with Constraints Applied on External Faces

Because of the way that the plugin specifies boundary conditions and equations on the surface nodes of the FE-RVE, applying constraints to these nodes which deactivate degrees of freedom (e.g., tie constraints) can cause the analysis to fail because nodes will be slaves in multiple constraints. An example of this would be tying two instances together where the boundary of the instances crosses the RVE boundary. In cases when nodes on the surface must be constrained, the slaved nodes should be added to an assembly-level nodeset called *RVE_USER_IGNORE_FOR_PAIRING*. The nodes in this nodeset will not have boundary conditions applied to them.

1.2.5. Coincident Surface Nodes

For cases in which there are multiple nodes at a location on the surface of the RVE, facet matching is used to determine how to pair nodes between pairs of periodic faces.

1.2.6. Reserved Nodeset Names

Do not create sets/surfaces/materials/sections/parts/instances/constraints/amplitudes containing the substring *RVE_AG* in their name. The plugin automatically creates a number of assembly-level sets and other objects. These sets typically contain *RVE_AG* as a substring to denote that they were auto-generated by the plugin. The cleanup routine called when setting up an FE-RVE analysis searches the model for various objects containing this substring and deletes them.

1.2.7. Steps, Boundary Conditions, and Loads

The plugin assumes that it is responsible for creating all steps, loads, and boundary conditions in the analysis. The cleanup routine deletes all step, BC, and load objects from the model before setting up the analysis.

1.2.8. Incompatible Abaqus/CAE Features

The following is a non-comprehensive list of features and characteristics in Abaqus and CAE which are not supported when using the Micromechanics plugin:

- Model attribute “Do not use parts and assemblies in input files” is not supported and must be unchecked.
- Volume-Averaged quantities will not be correct when using structural element types such as shells and beams

1.3. Types of Boundary Conditions

The plugin is able to specify boundary conditions which are appropriate for a number of different types of RVE.

1.3.1. Periodic

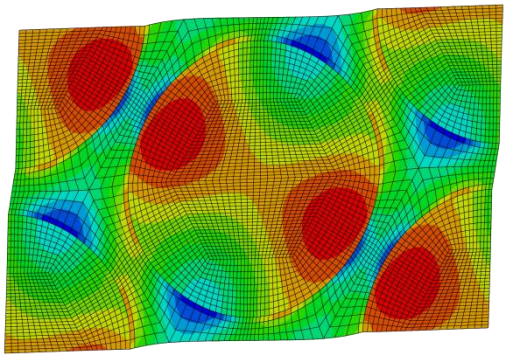


Figure 2: Stress in an RVE with periodic boundary conditions

A periodic response assumes that the solution field φ (e.g. displacement u_i, temperature θ, etc.) exhibits the following form:

$$\varphi(x_j + p_j^\alpha) = \varphi(x_j) + \langle \frac{\partial \varphi}{\partial x_j} \rangle p_j^\alpha$$

x_j is the coordinate, p_j^α is the α^{th} vector of periodicity, and $\langle \frac{\partial \varphi}{\partial x_j} \rangle$ is the far-field gradient of the solution field (for instance, the far-field displacement gradient or temperature gradient).

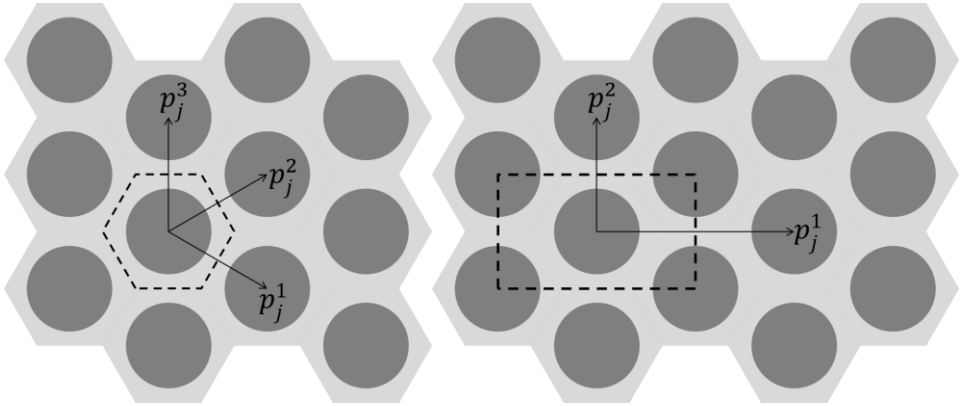


Figure 3: Vectors of periodicity for different unit cell definitions

The plugin imposes this relationship on the boundary nodes of the RVE through the use of equation constraints. The far-field gradient is introduced through the degrees of freedom of assembly-level nodes, called far-field reference nodes, which are added to the analysis and are not attached to any elements. The far-field gradient can be specified by applying boundary conditions to these far-field reference nodes. Alternatively, the far-field flux (e.g. volume-average stress, volume average heat flux vector) is related to the conjugate flux terms for these far-field reference nodes through energy equivalence, and therefore can be specified by applying a concentrated flux or force to the far-field reference node degrees of freedom.

Imposing periodic boundary conditions requires nodes to be positioned periodically. If nodes are not positioned periodically, a novel approach is used to copy and tie facets from one side of the RVE to the other to facilitate the imposition of the periodic equations.

NOTE – Applying periodic boundary conditions to meshes that are not periodic can lead to considerable errors in the solution field near the non-periodic RVE boundaries, particularly when the facets of the slaved surface are larger than or of similar size to the facets of the master surface. Results should be examined carefully in these situations, particularly if there are nonlinear phenomena such as damage in the analysis, as these can lead to non-realistic predictions that can significantly affect the homogenized response of the RVE.

1.3.2. Uniform Surface Gradient (Taylor)

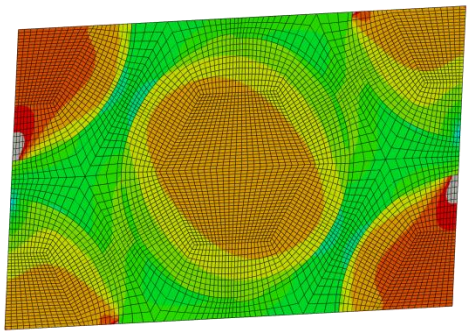


Figure 4: Stress contours in an RVE with uniform surface gradient boundary conditions

The uniform surface gradient boundary condition, often referred to as Taylor boundary conditions, specifies that the solution field value of the RVE's boundary nodes to conform to the far-field solution gradient by imposing the following relationship on the boundary nodes:

$$\varphi(x_j)\big|_\Gamma = \left\langle \frac{\partial \varphi}{\partial x_j} \right\rangle x_j$$

One benefit of this boundary condition is that it does not require that the FE-RVE have periodic geometry. Note that this differs from Voigt conditions in that only the boundary of the RVE is constrained to the far-field gradient. The solution gradient within the RVE is generally not uniform (unless the RVE itself is uniform). For periodic RVEs, applying the uniform surface gradient boundary condition will yield a stiffer response than periodic boundary conditions. The boundary conditions introduce a boundary effect. As an RVE becomes larger, this boundary effect will have less influence on the overall RVE response.

The uniform surface gradient boundary condition is imposed in a similar manner to periodic boundary conditions, in that the far-field solution gradient is introduced through far-field reference nodes and equations are defined relating the degrees of freedom of the boundary nodes to the far-field reference node degrees of freedom.

1.3.3. Uniform Surface Flux (Neumann)

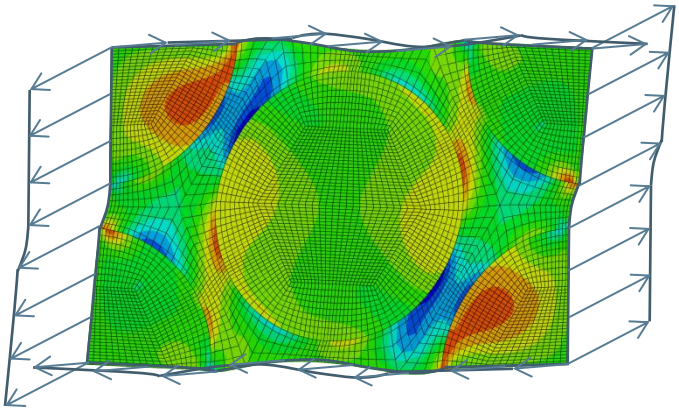


Figure 5: Stress contours and applied tractions on RVE with uniform surface flux boundary conditions

The uniform surface flux, or Neumann boundary condition applies the far-field flux as a distributed surface flux to the outer boundary of the RVE. For instance, for mechanical analysis, the following equation relates the applied surface traction t_i to the far-field stress $\langle \sigma_{ij} \rangle$.

$$t_i(x_k)|_\Gamma = \langle \sigma_{ij} \rangle \cdot n_j(x_k)$$

Like the uniform surface gradient boundary condition, this boundary condition is also appropriate for FE-RVEs that are not periodic. Also like the uniform surface gradient condition, there is a boundary effect inherent with the uniform surface flux boundary condition that will have less effect on RVE response as the RVE increases in size. For periodic RVEs, the uniform surface flux boundary condition will yield more compliant response than periodic boundary conditions.

Note – Presently, when applying this boundary condition, the plugin assumes that faces are flat and aligned with the global axes, so using it with user-defined RVEs that are not hexahedra with flat, axis-aligned boundaries will yield incorrect results.

Note – For a number of reasons related to both imposing correct loads and obtaining correct volume average response from the RVE, it is not recommended to use the uniform surface flux boundary condition with RVEs that contain unmeshed voids. If the RVE does contain voids and it is required to use the uniform surface flux boundary condition, the voids should be meshed using a material with negligible stiffness.

1.3.4. Periodic Shell Boundary Conditions

The plugin is able to impose shell kinematics in a periodic manner to an in-plane unit cells of thin, shell-like structures. Such structures have vectors of periodicity running the x-y plane and have free top and bottom z surfaces.

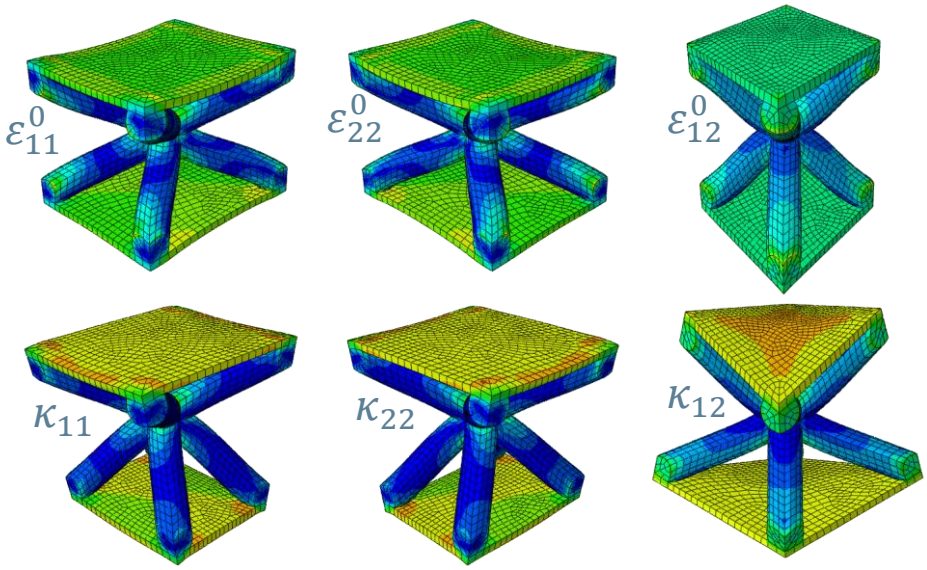


Figure 6: Periodic shell deformation of a unit cell of a lattice-core panel

The kinematics of periodically-related nodes to the far-field shell membrane strain and curvature are given by the following equations:

$$u_\alpha\left(x_i + p_i^m\right) = u_\alpha\left(x_i\right) + \varepsilon_{\alpha\beta}^0 p_\beta^m + x_3 \kappa_{\alpha\beta} p_\beta^m$$

$$u_3\left(x_i + p_i^m\right) = u_3\left(x_i\right) + \frac{x_\alpha x_\beta - \left(x_\alpha + p_\alpha^m\right)\left(x_\beta + p_\beta^m\right)}{2}\kappa_{\alpha\beta}$$

$$\alpha, \beta \text{ have values } 1,2 \quad p_3 = 0 \quad \varepsilon_{\alpha\beta}^0 = \varepsilon_{\beta\alpha}^0 \quad \kappa_{\alpha\beta} = \kappa_{\beta\alpha}$$

Note that the the above constraints are imposed on the unit cell about it's centroidal x_1 and x_2 coordinates, but x_3 is not modified as this would alter the bending stiffness and bending-extension coupling of the shell-like structure. The global coordinate x_3 is assumed to be the reference plane of the shell. Further information on periodic shell boundary conditions can be found in the following references:

Karkkainen, R. L., & Sankar, B. V. (2006). A direct micromechanics method for analysis of failure initiation of plain weave textile composites. *Composites Science and Technology*, *66*(1), 137–150.

Gigliotti, L., & Pinho, S. T. (2015). Exploiting symmetries in solid-to-shell homogenization, with application to periodic pin-reinforced sandwich structures. *Composite Structures*, *132*, 995–1005.

Note that for models using shell periodicity that have elements with rotational degrees of freedom on the boundary, the relationship between the periodic rotational degrees of freedom

depends on the far-field curvature of the unit cell and its current in-plane dimension. The plugin currently imposes these constraints in a linear fashion that does not account for changes to the in-plane dimensions of the unit cell. This will lead to inaccuracies in the rotational degrees of freedom at the unit cell boundary for geometrically nonlinear models which experience finite rotations and appreciable change of in-plane dimension.

2. Mean-Field

Mean field homogenization is a material type implemented in Abaqus 2017. It allows a material to be defined as a combination of multiple constituent materials. For a discussion of mean field homogenization, please refer to Abaqus (version 2017 or later) documentation under Abaqus → Materials → Multiscale Properties → Mean-field homogenization.

3. Plugin GUI

3.1. Overview

The Plugin GUI is set up using a tree structure to facilitate different workflows. Currently the FE-RVE part of the tree can be used to set up FE-RVE analyses.

3.2. FE-RVE

The FE-RVE part of the plugin is separated into several different “leafs” that provide distinct functionality, described in the following sections.

3.2.1. Library Leaf

The plugin provides functionality for defining certain types of RVE parametrically. Users also have the option of creating their own FE-RVE following the conventions described in section **1.2 Creating an FE-RVE Model**.

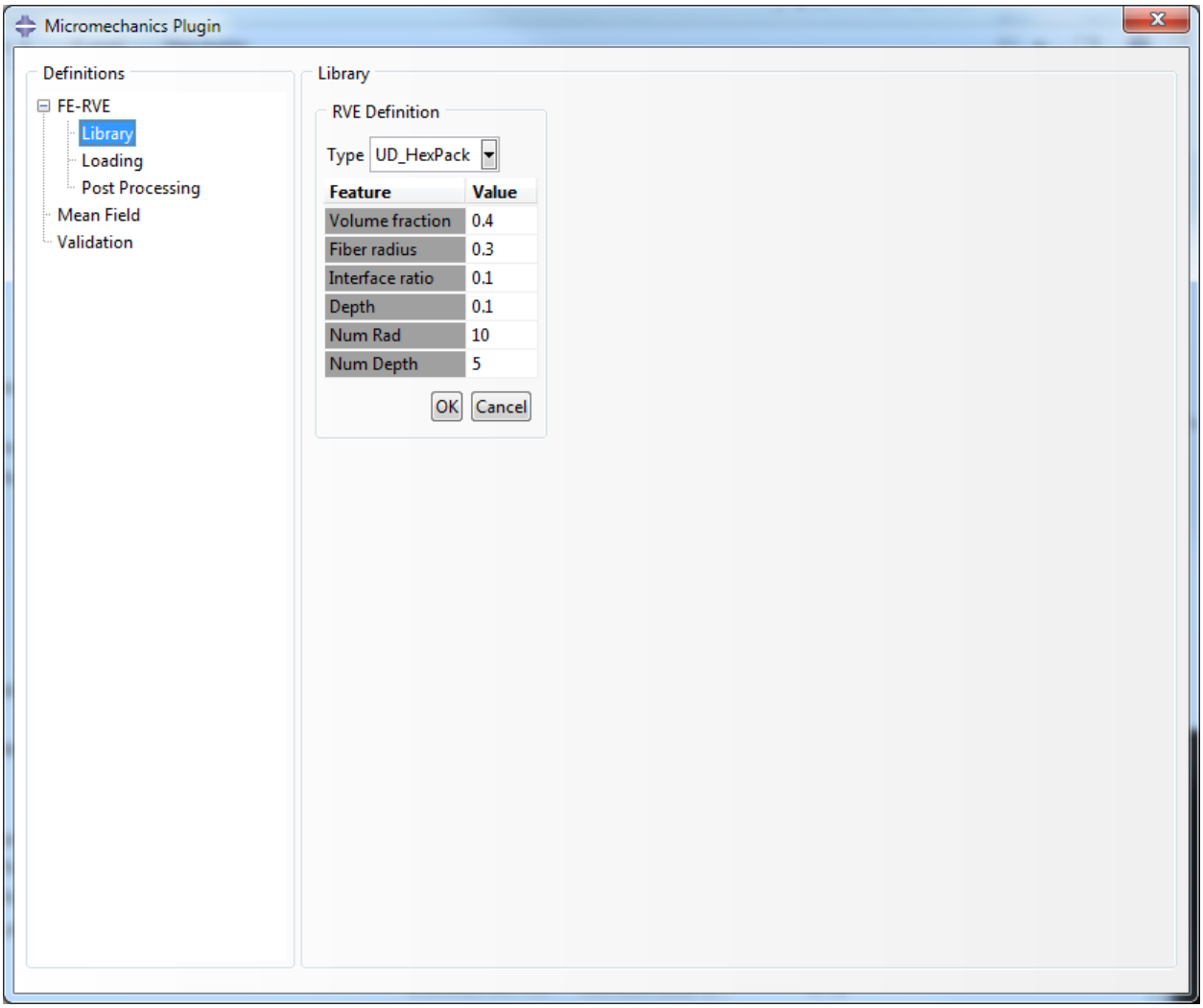


Figure 7: Library leaf

3.2.1.1. Unidirectional Composite with Hexagonally Arrayed Fibers

The plugin provides functionality to create a fiber-matrix unit cell for a unidirectional composite with hexagonally arrayed fibers by specifying a limited number of parameters.

The RVE will be created with placeholder materials. These materials should be modified to have appropriate values for the user's material system and choice of units after the RVE has been created.

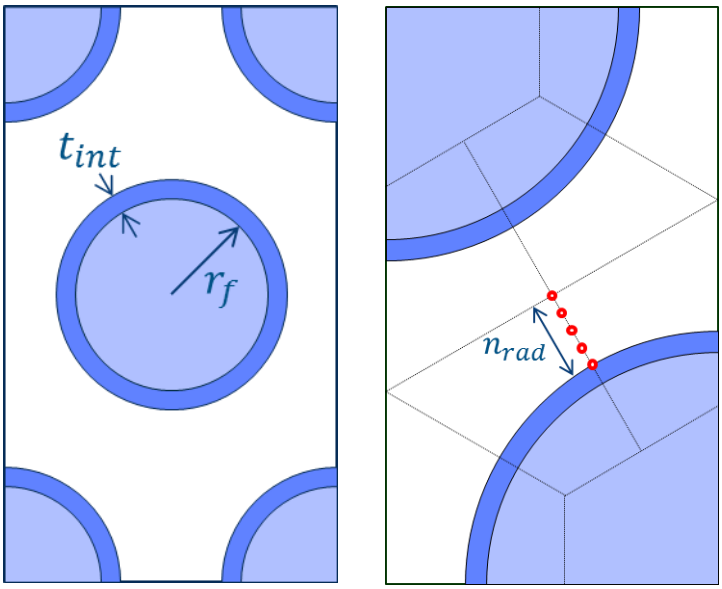


Figure 8: UD composite with hexagonally-arrayed fibers showing various parameters

To create the RVE, do the following:

1. Choose *UD_HexPack* from the **Type** drop-down menu
2. Specify the parameters to define the unit cell as follows:
   - **Volume Fraction** - Fraction of fiber's volume compared to whole RVE
   - **Fiber Radius** - Radius of the fiber
   - **Interface Ratio** - Thickness of interface as fraction of fiber radius; $t_r = t_{int}/r_f$. Specifying a value of zero will result in no interface being created.
   - **Depth** - Dimension of RVE in fiber-direction
   - **Num Rad** - Number of elements radially from fiber or interface to mid-point between adjacent fiber.
   - **Num Depth** - Number of elements along the fiber direction.
3. Click OK
4. Modify the placeholder materials to appropriate values.

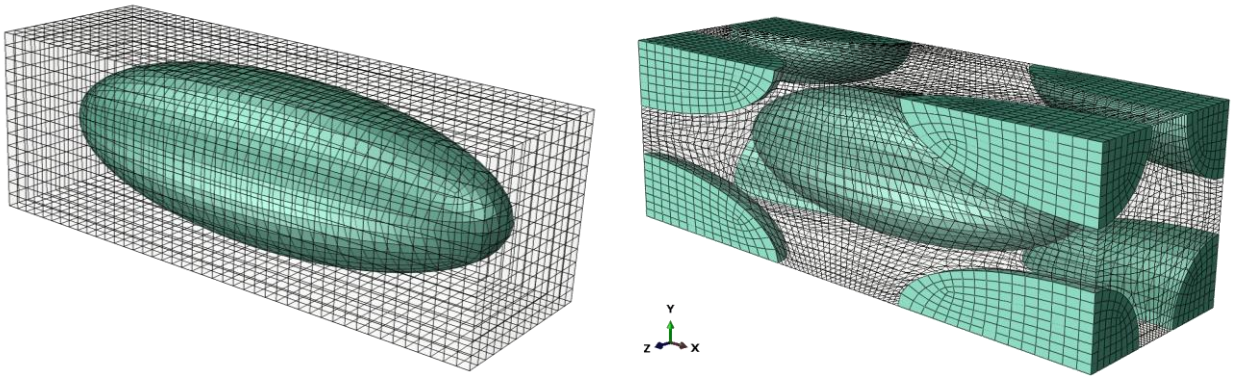
A script will be run to generate the fiber-matrix unit cell geometry and mesh based on the input parameters. Due to close packing of circles, the volume fraction and interface ratio must satisfy to the following inequality: $v_f(1 + t_r)^2 < \sqrt{3}\pi/6$.

If using periodic boundary conditions, the strain field will not vary in the fiber direction, and response and accuracy will not depend on mesh refinement in the fiber direction. In this situation, it is recommended to use Num Depth = 2 and Depth approximately 1/10 the fiber radius.

3.2.1.2. Ellipsoid Array

The plugin provides functionality to create a unit cell of an array of ellipsoidal inclusions. This can be convenient for comparison to the mean-field homogenization material model, since Eshelby-based formulations (like Mori-Tanaka) assume ellipsoidal inclusions.

The RVE will be created with placeholder materials. These materials should be modified to have appropriate values for the user's material system and choice of units after the RVE has been created. Additionally, it is possible that meshing of the matrix will fail for higher volume fractions and/or aspect ratios significantly different than 1.0. In such situations, the user must manually mesh the RVE.



(a) Simple **(b) Body-Centered**

Figure 9: Ellipsoid Array

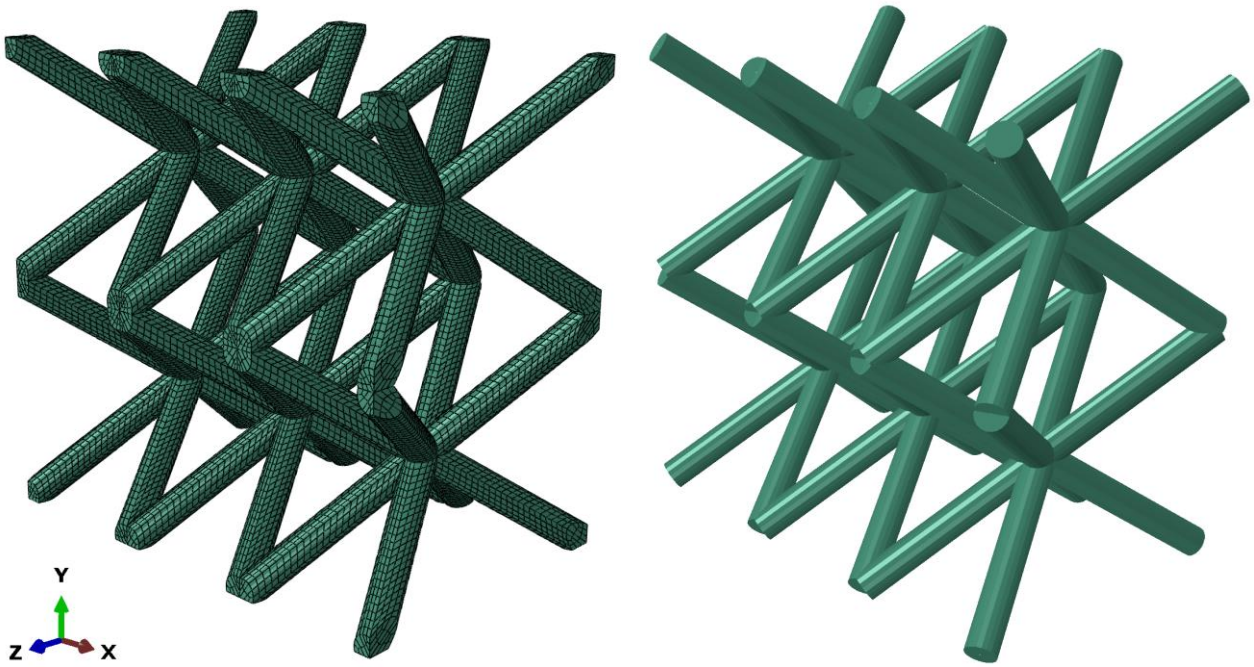
To create the RVE, do the following:

1. Choose *Ellipsoid* from the **Type** drop-down menu
2. Specify the parameters to define the unit cell as follows:
    - **Array** – Simple or Body-Centered, defines how ellipsoids are positioned
    - **Volume Fraction** – Fraction of fiber's volume compared to whole RVE
    - **Aspect Ratio** – the aspect ratio of the axisymmetric ellipsoidal inclusion
    - **Axial Radius** – one-half the total dimension of the inclusion along its axis of revolution
3. Click OK
4. Modify the placeholder materials to appropriate values.

Note – The script that generates this unit cell does not result in a periodic mesh, which can lead to errors in the solution near the unit-cell boundary where nodes do not line up.

3.2.1.3. Body-Centered Lattice

The plugin provides functionality to create a unit cell of a body-centered lattice. The unit cell can be generated using solid or beam elements, and can have different dimensions and can be repeated multiple times in each of the x, y, and z directions, and can be created with solid or beam elements.



(a) Solid Elements **(b) Beam Elements**

Figure 10: Lattice Unit Cell

To create a lattice RVE, do the following:

1. Choose *BC Lattice* from the **Type** drop-down menu
2. Specify the parameters to define the unit cell as follows:
    - **Radius** – the radius of the lattice members
    - **X/Y/Z Dim** – the dimension of a single lattice unit cell in the X/Y/Z direction
    - **X/Y/Z Cells** – the number of individual cells to array in the X/Y/Z direction
    - **Element Type** – Choose whether to make the RVE with solid or beam elements
3. Click OK
4. By default, the model is created with a material definition for Ti-6Al-4V alloy based on sources from academic literature.

Note – The script that generates this unit cell does not result in a periodic mesh for solid elements, which can lead to errors in the solution near the unit-cell boundary where nodes do not line up.

3.2.1.4. General Lattice with Solid Elements (No GUI)

Through the Python scripting interface, the plugin includes the ability to define general lattice unit cells through the specification of the unit cell dimension and each lattice member's end coordinates and radius. As an example, by defining the appropriate vertices, one could create a lattice with a rhombic dodecahedral shape. The only requirement is that the lattice be defined within a hexahedral unit cell with boundaries aligned with the global axes.

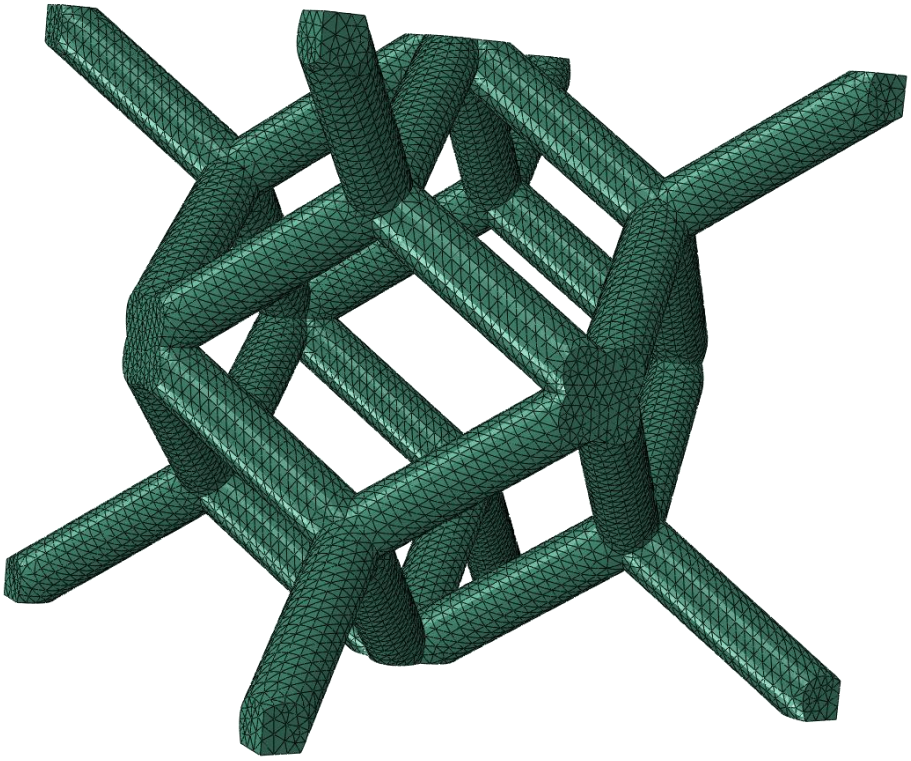


Figure 11: Rhombic Dodecahedral Lattice

This functionality is accessed by directly calling a Python method within the plugin. To call this method, the kernel module of the plugin must be included in the Python search path. Starting at the root path of the plugin, this module is located in the following directory:

/MicroMechanics/microMechanics/mmpBackend

There are numerous ways to add directories to the python search path. One convenient way is to append to the *sys.path* variable in Python. An example of this is below:

```
import sys
import os
sys.path.append(os.path.join('<...PathToPlugin...>',
                             'MicroMechanics',
                             'microMechanics',
                             'mmpBackend'))
```

Once this directory is included in the python search path, it is possible to import the *mmpKernel* module and call the method as given in the example at the end of this section. The method is in the *Library.Lattice* submodule and is named *generateGeneralLattice*. It includes the following arguments:

Argument	Description
modelName	Name of the model to be created in the current mdb
cellDims	Tuple of form (dimX, dimY, dimZ) giving overall dimensions of the unit cell
memberData	Sequence of tuples defining each member in the unit cell, with each tuple formatted (startX, startY, startZ, endX, endY, endZ, radius)
repeatedCells	Tuple defining how many repetitions of the lattice unit cell to generate in each direction, formatted (nX, nY, nZ)
seedRadiusFraction (optional)	Global seed size to assign as a fraction of the minimum member radius. The default value is 0.5

By default, the lattice will be generated using quadratic tetrahedral elements. The unit cell will be centered on the centroid of the bounding box for all the given vertices

To generate a single body-centered lattice unit cell with dimensions 1 x 2 x 3 using this approach, the following Python script could be used:

```
import mmpKernel as Kernel
if "Lattice" in mdb.models.keys():
    print "Deleting current Lattice model"
    del mdb.models["Lattice"]
generatedModel = Kernel.Library.Lattice. generateGeneralLattice(
        "Lattice",
        (1.0, 2.0, 3.0),
        [ (0.0, 0.0, 0.0, 1.0, 2.0, 3.0, 0.1),
          (1.0, 0.0, 0.0, 0.0, 2.0, 3.0, 0.1),
          (0.0, 2.0, 0.0, 1.0, 0.0, 3.0, 0.1),
          (0.0, 0.0, 3.0, 1.0, 2.0, 0.0, 0.1) ]
        (1, 1, 1),
        0.5 )
```

Running this script will create a new model in the current model database.

3.2.2. Loading Leaf

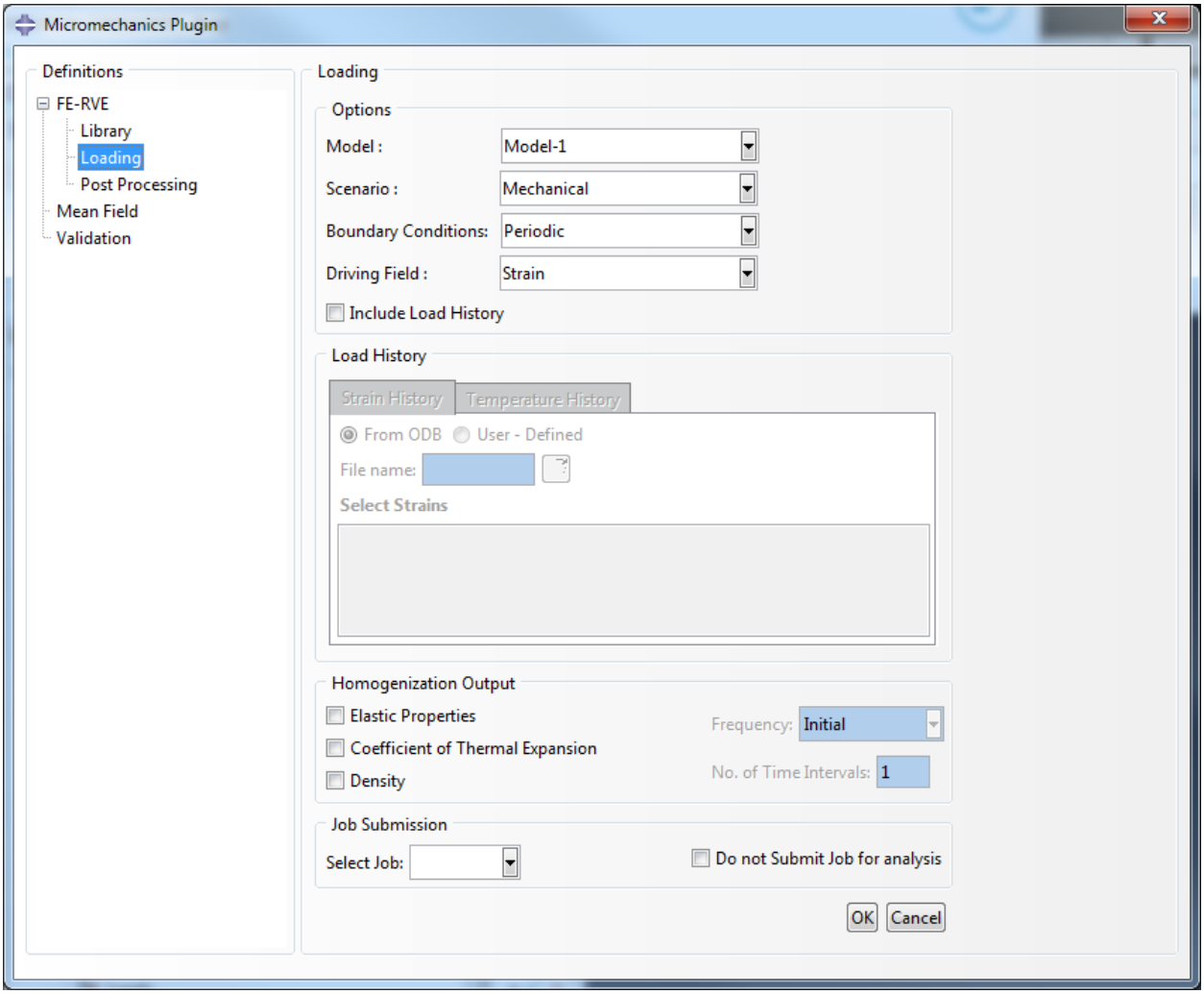


Figure 12: Loading leaf

The loading leaf is used to set up a model for analysis and run it. It assumes that the RVE model's geometry and mesh have already been created along with materials, sections, and section assignments. On clicking OK in the loading leaf, the plugin will collect the specified analysis options and then call the kernel functions required to create the sets, steps, boundary conditions, and loads required to perform the desired FE-RVE analysis. The selected job will then be submitted via Abaqus/CAE.

3.2.2.1. Options Panel

The options panel provides overall options that determine how to set up the analysis. Not all options are available for all scenarios.

1. **Model** – The model in the current Abaqus/CAE session that will be set up for analysis
2. **Scenario** – The type of analysis to be performed – see section **1.1 Scenarios**.
3. **Boundary Conditions** – The applied boundary conditions – see section **1.3 Types of Boundary Conditions**.

4. **Driven Field** – The type of driven field. This is only available for the Mechanical scenario. This in combination with the *Boundary Conditions* option determines what specific boundary condition scheme is applied when *Non-Periodic* is selected under *Boundary Conditions*
    - *Strain* – Uniform Surface Gradient (Taylor) is specified
    - *Stress* – Uniform Surface Flux (Neumann) is specified
5. **Include Load History** – species whether to apply a load history to the RVE. Leaving this unselected means that analysis will only consist of the steps needed to perform the requested homogenization about the RVE's initial state.

3.2.2.2. Load History Panel

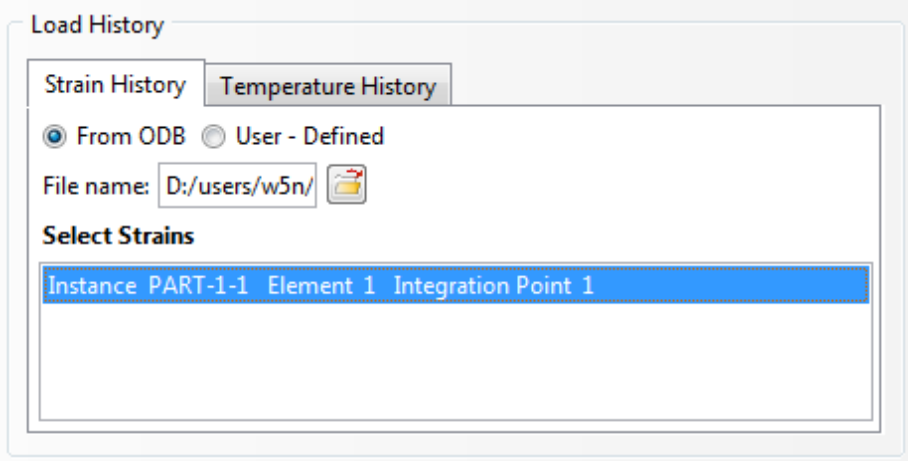


Figure 13: Load history panel – From ODB option

The Load History panel is where the time history of the far-field solution is defined. For most fields, it is possible to either read the far-field history from an odb file (e.g., read the strain at an integration point from a larger-scale analysis) or to manually specify the far-field history. If reading from an .odb, the driven field quantity must be available either as history data at an integration point or as XY-Data that has been extracted from odb field data. Locations where the necessary data is available are listed in the Load History panel.

Note – All far-field boundary conditions are imposed through the boundary constraints as nominal values. The plugin will convert far-field log strain read from an odb to the equivalent nominal strain for imposition of strain boundary conditions. However, far-field stress quantities are always treated as the 1st Piola Kirchoff stress (and are not converted from Cauchy stress when read from an odb). Under large deformation in which the 1st PK stress may be unsymmetric, the shear value specified to the plugin should be the average of the 1st PK shears.

Note – If large time increments are taken in a macro-scale model used to drive an RVE, it is possible that the RVE results will not be representative of realistic behavior at time points in the RVE analysis that are not close to increment times from the macro-scale analysis. This is because the multiaxial strain or stress state driving the RVE is simply the component-wise linear interpolation of the strains or stresses from the macro-scale

analysis. This may not accurately represent a converged state for nonlinear material behavior when large increments are taken.

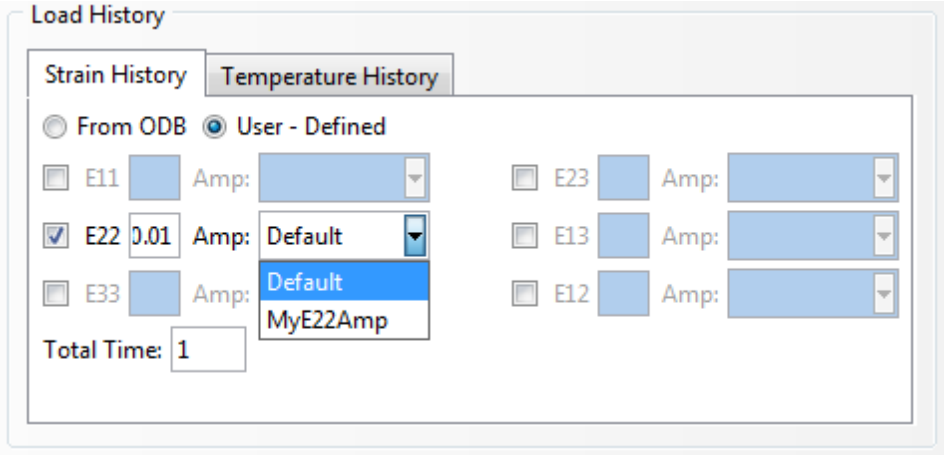


Figure 14: Load history panel - User-Defined option

Manually specifying the far-field history is accomplished by selecting the *User-Defined* radio button. This will bring up a menu that permits any combination of components of the far-field gradient to be specified. Any left unspecified are unconstrained. Amplitudes in the model can be assigned to drive each component individually. When defining a user-defined load history, it is also necessary to specify the total duration of the analysis in the *Total Time* text box. User-defined far-field strain and stress values should be given as nominal strains and stresses. Shear strain values should be expressed as engineering shear in keeping with Abaqus convention.

Note that in most cases, it is possible to specify multiple fields simultaneously using the different tabs in the Load History panel. For instance, in the mechanical scenario with *Strain* as the driving field, it is possible to specify a far-field strain as well as a uniform temperature that will be applied to the RVE (which could cause thermal strain or affect temperature-dependent properties). For all scenarios, the first tab determines the total time of the analysis. When reading results from an ODB, the total time (and overall step structure) will mirror the analysis from which the field history is being read.

3.2.2.1. Homogenization Output Panel

Figure 15: Homogenization Output Panel

The Homogenization Output panel is where the required homogenization is specified. Steps will be set up based on the options specified in this panel so that the required homogenized response can be measured. The Frequency box allows specification of whether

homogenization will only be performed once at the beginning of the analysis or multiple times throughout the load history. In this manner, the change in the response can be observed as nonlinear behavior such as damage occurs. If *Time Intervals* is requested, each load history step will be subdivided into however many steps are defined in *No. of Time Intervals* and homogenization will be performed between each subdivided step.

3.2.2.2. Job Submission Panel

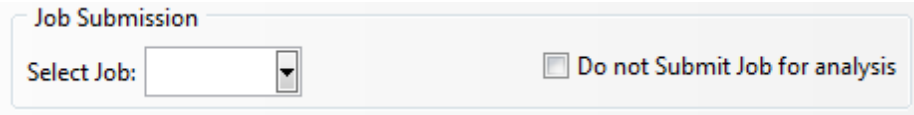


Figure 16: Job Submission panel

The job submission panel is where the Abaqus/CAE job is selected for the analysis that will be run. The drop-down will include all jobs in the current Abaqus/CAE session that are associated with the selected model. On clicking OK, the plugin will create all the necessary steps, loads, and other features needed to perform the analysis and will submit the job in Abaqus/CAE, unless the “Do not submit job for analysis” checkbox is selected, in which case the analysis is set up but not submitted. This can be useful if it is necessary to modify the analysis in some way – for instance, to change the incrementation controls in the load history steps to facilitate modeling non-linear behavior.

3.2.3. Post-processing Leaf

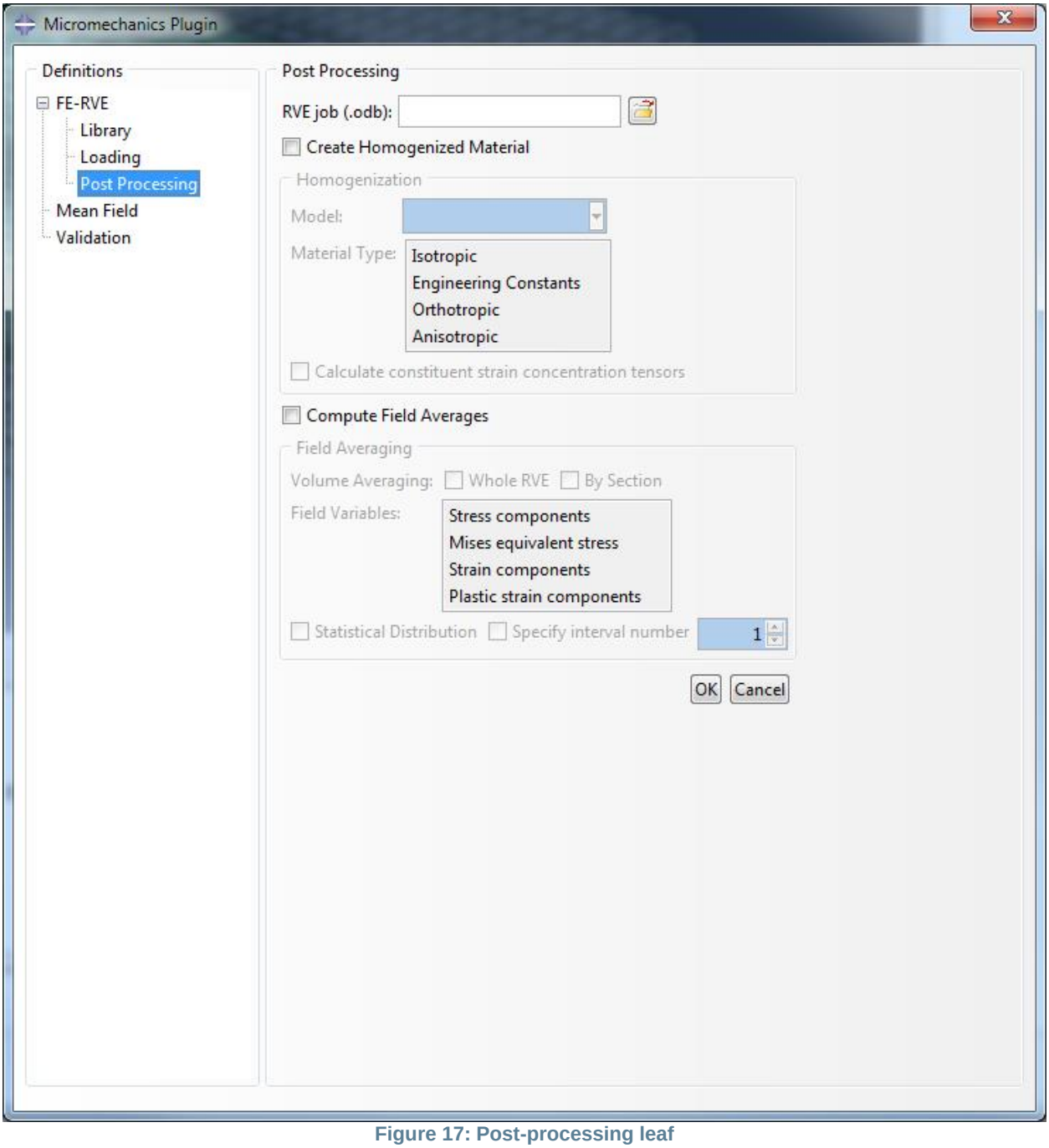


Figure 17: Post-processing leaf

The post-processing leaf is where homogenization calculations and field post-processing is performed after an FE-RVE analysis has been run. Start by selecting the FE-RVE analysis .odb file in the *RVE job* field.

3.2.3.1. Homogenization Panel

The homogenization panel directs the calculation of the homogenized material response. From the *Model* drop-down, select the model in the current Abaqus/CAE session where the plugin should write the homogenized properties. From the *Material Type* field, select the types of definitions that should be created. On clicking OK, the homogenized response will be obtained from the FE-RVE .odb for each time point at which homogenization was requested, and a material object will be written to the selected model object for each item selected in the *Material Type* field. Solid-to-shell scenario analyses result in section definitions being written to the model object. These material and section definitions can then be conveniently used in analyses at larger scales. Note that it is possible to request homogenized response types that are not in good agreement with the actual response of the FE-RVE. For instance, a unidirectional composite will not exhibit an isotropic response even though an isotropic material definition can be requested. Some of the generated material definitions include relative error norms in their descriptions indicating how much the requested material response differs from the fully anisotropic response of the FE-RVE.

Selecting the *Calculate Constituent Strain Concentration Tensors* checkbox causes files to be written to the working directory for each section in the model at each homogenization time point relating the far-field average strain to the average strain within a section according to:

$$\left\langle \varepsilon_\alpha \right\rangle^{\text{section}} = A_{\alpha\beta}^{\text{section}} \left\langle \varepsilon_\beta \right\rangle^{\text{whole model}}$$

3.2.3.2. Field Averaging Panel

The Field Averaging panel allows the creation of plots of the time history of certain fields on a whole-RVE and/or per-phase basis. It also allows the statistical distribution of a field within the whole RVE and/or a given phase at a point in time to be visualized in a histogram. The number of bins in the histograms can be automatically specified by the plugin or can be manually specified by the user. Fields are transformed into the RVE's global coordinate system for averaging and histogram creation. The current volumes in the model are used whenever possible, which will impact volume averaging for models experiencing large deformations.

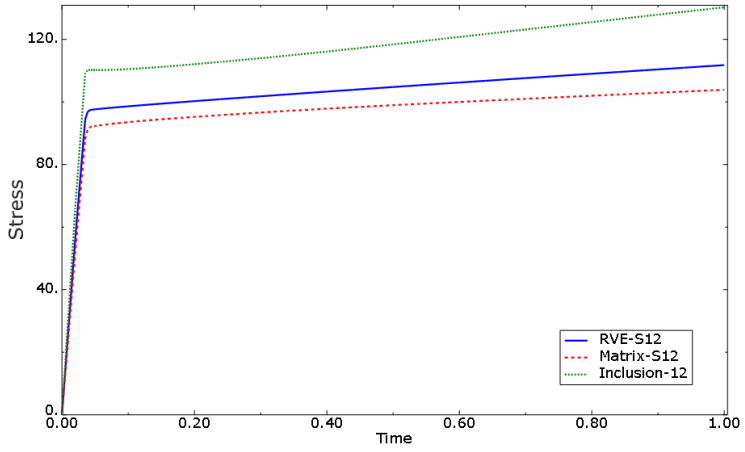


Figure 18: Time history of stress

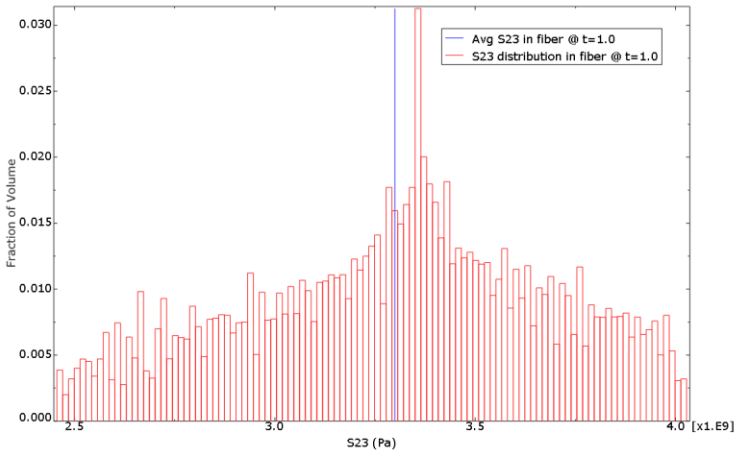


Figure 19: Distribution of stress in a given phase

In addition to outputting field quantities averaged by looping over elements in the RVE, when outputting Whole-RVE field data for mechanical models that make use of far-field reference nodes (i.e., models with periodic and uniform surface gradient BCs), several whole-RVE field quantities are calculated from the reference nodes' degrees of freedom and their conjugates:

XY-Data Name	Description
VolAvgRP-WholeRVE-NE##	The volume-average nominal strain of the RVE calculated from the degrees of freedom of the far-field reference nodes
VolAvgRP-WholeRVE-LE##	The volume-average logarithmic strain of the RVE, calculated from the volume-average nominal strain. Note that due to the nonlinear nature of the logarithmic operator, this quantity can differ from the volume-average logarithmic strain *VolAvg-WholeRVE-LE##* which is the volume-average of the logarithmic strain in each element of the RVE.
VolAvgRP-WholeRVE-1PKS##	The volume-average 1st Piola-Kirchoff stress of the RVE calculated from the conjugate values of the far-field reference nodes' degrees of freedom. All nine components are given as this stress is generally unsymmetric under large deformation.
VolAvgRP-WholeRVE-S##	The volume-average Cauchy stress of the RVE calculated from the volume average nominal strain and 1st Piola-Kirchoff stress.

Note – If an RVE contains voids or cracks, beam, shell or other structural elements, or other volume that is not accounted for by 3D continuum elements in the RVE, any field quantity which is not zero in these unmeshed regions or structural elements will have an incorrect volume average value when calculated by looping over elements in the RVE. Examples of this would be strains in void regions or stresses in structural elements. For RVEs driven using far-field reference nodes (i.e. periodic or uniform surface gradient boundary conditions) it is possible to obtain certain whole-RVE average field values from the reference nodes. Whole-RVE quantities obtained in this manner will be given with XYData objects prepended with the string "VolAvgRP".

Note – If an RVE under mechanical loading does not use reference points (i.e. uses uniform-surface-flux-type boundary conditions) and contains voids, the whole-RVE volume cannot be properly updated to account for the presence of the voids, which will result in inaccuracies in all volume-averaged quantities.

3.3. Mean-Field

This part of the GUI is not yet implemented.

3.4. Validation

This part of the GUI is not yet implemented.

4. Plugin Kernel

The plugin's back-end functionality is implemented using kernel functions that can be directly accessed with the Python scripting interface and leveraged to implement more advanced complex workflows than can be facilitated using the GUI. Documentation of these kernel functions may be provided at a future time. If the user wishes to script workflows involving the plugin, they may find it helpful to examine the Abaqus/CAE replay file (abaqus.rpy) where high-level calls to the plugin's Interface module are recorded.

5. User Variables and Functions

Users can override certain quantities such a tolerance values used within the plugin by defining non-default variables. This is done by including a file named **MicroMechanicsUserVars.py** in the PYTHONPATH used by Abaqus/CAE. A convenient choice of location is the current working directory for a model. A message will be issued to the message area in Abaqus/CAE upon opening the plugin when non-default variables are defined.

Environment variables can be specified in **MicroMechanicsUserVars.py** as follows:

```
RELATIVE_DIMENSION_TOLERANCE = 1E-4
PERIODICITY_VEC_ZERO_TOLERANCE = 1E-4
def myVolumeFunc(modelOdb):
    return 42
USER_ODB_VOLUME_CALCULATOR = myVolumeFunc
```

There are several variables that may need to be redefined by a user in certain situations:

Variable	Default	Description
RELATIVE_DIMENSION_TOLERANCE	1E-6	The distance relative to the model dimension that will be searched to automatically identify nodes on the x, y, and z faces
RELATIVE_PERIODIC_POSITION_TOLERANCE	1E-6	Tolerance relative to the bounding box diagonal of the slave nodeset to determine if nodes are positioned periodically
PERIODICITY_VEC_ZERO_TOLERANCE	1E-6	If a component of the periodicity vector divided by the vector's total length is less than this, equation terms associated with this component of the periodicity vector are assumed zero and are not written to the periodic equation
PERT_STEP_TIME_FRACTION	1E-8	The relative duration of steps (compared to the total load history time or 1.0 if no load history is included) used for homogenization-type steps in procedures that do not support linear perturbation-type steps (i.e. Thermal, Coupled-temperature-displacement)
COUPLED_PERT_TEMP_DIFF	1.0	The amount that the uniform temperature is perturbed in coupled temperature-displacement analyses in the CTE homogenization step
COUPLED_PERT_STRAIN	1E-3	The amount that the applied strain is perturbed about its current state in stiffness homogenization steps for coupled temperature-displacement analyses.

In addition to these variables, there are several handles to certain functions that the user can define which will override functions used internally in the plugin. These can be particularly useful in situations where the user is analyzing unit cells that are not hexahedra with flat axis-aligned faces.

Variable	**Description**	**Function Inputs**	**Function Outputs**
USER_NODE_PAIRING_FUNCTION	Sorts pairs of periodically positioned nodes so that the i^{th} nodes in each list are periodically positioned. Returns Python *None* if nodes are not periodic.	*masterNodes* (list of MeshNodes), *slaveNodes* (list of MeshNodes), *vectorOfPeriodicity* (list of 3 floats), *relTolerance* (float)	*masterNodes_ordered*, *slaveNodes_ordered* (lists of MeshNodes) **or** *None* (if nodes are not periodically positioned)
USER_MDB_VOLUME_CALCULATOR	Returns the volume (including voids) of a unit cell from a model database (used in model setup)	*modelMdb* (mdb model object)	*volume* (float)
USER_ODB_VOLUME_CALCULATOR	Returns the undeformed volume (including voids) of a unit cell from an output database (used in model post-processing and homogenization)	*modelOdb* (odb object)	*volume* (float)
USER_ODB_AREA_CALCULATOR	Returns the undeformed in-plane area (including voids) of a shell unit cell from an output database (used in model post-processing and homogenization)	*modelOdb* (odb object)	*area* (float)