

# LA-UR-15-25143

Approved for public release; distribution is unlimited.

Title: MCNP6 Unstructured Mesh Tutorial Using Abaqus/CAE 6.12-1

Author(s): Joel A. Kulesza  
Roger L. Martz

Intended For: General Reference / MCNP Website

Issued: July 9, 2015



**Disclaimer:** Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

## Abstract

The purpose of this report is to provide a tutorial for nuclear engineers, who have some experience with 3-D design software, in how to create an MCNP6 unstructured mesh model. This is not an exhaustive review of 3-D CAD/CAE design tools and techniques, but rather is intended to provide sufficient instruction for an engineer to proceed through the steps necessary to create an unstructured mesh model using the current Abaqus CAE tool. This report is suitable for users who have Abaqus available to create an unstructured mesh input file from scratch and for users who have an Abaqus-formatted mesh file and need to perform the final steps of preparation prior to MCNP6 execution.

The reader is taken through the process of creating Abaqus Parts (the fundamental geometric building block within Abaqus), assigning element sets to the Parts, defining materials, meshing the Parts, and combining the Parts into an Assembly for use in MCNP6. This document has screenshots to accompany the discussion and the major steps are also captured in Python scripts (which can be directly used within Abaqus) to reproduce the work described herein. Utilities provided with MCNP6 are used to build a skeleton input file from the Abaqus mesh file and an example, completed, input file is provided. Finally, the calculation is executed and the results are examined (briefly).

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Abaqus Operations</b>	<b>5</b>
2.1	Running Abaqus Python Macros . . . . .	5
2.2	Geometry Creation . . . . .	5
2.3	Part Cutting Operations . . . . .	10
2.4	Assignment of Element Sets (elsets) . . . . .	14
2.5	Definition of Materials . . . . .	15
2.6	Mesh Parts . . . . .	17
2.7	Create Assembly . . . . .	21
2.8	Write Abaqus Mesh File . . . . .	22
<b>3</b>	<b>Pre-MCNP6 Execution Operations</b>	<b>23</b>
<b>4</b>	<b>MCNP6 Execution Notes</b>	<b>24</b>
<b>5</b>	<b>Results Processing</b>	<b>24</b>
5.1	Geometry Visualization . . . . .	24
5.2	Unstructured Mesh Results Visualization . . . . .	25
<b>6</b>	<b>Conclusions</b>	<b>29</b>
	<b>References</b>	<b>30</b>
	<b>Appendix A Abaqus Python Macros</b>	<b>31</b>
A.1	01_Rename_Model.py — Rename Model For Convenience . . . . .	32
A.2	02_Create_Paraffin_Block.py — Create Cube to Form Basis of Paraffin Block . . . . .	33
A.3	03_Create_Cutter.py — Create Conic Volume to Subtract from Paraffin Block . . . . .	34
A.4	04_Cut_Paraffin.py — Perform ‘Cut’ Operation on Paraffin Block with Cubic Volume . . . . .	35
A.5	05_Create_Graphite.py — Create 20 cm Graphite Shield . . . . .	36
A.6	06_Create_Air.py — Create 5 × 5 × 5 cm Air Detector Region . . . . .	37
A.7	07_Assign_Elsets.py — Assign Element Set Identifiers to Parts . . . . .	38
A.8	08_Create_Materials.py — Create Material Names & Assign Densities . . . . .	39

A.9	09_Partition_Paraffin.py — Partition Paraffin to Permit Hexahedral Meshing . . . . .	40
A.10	10_Mesh_Paraffin.py — Seed and Mesh Paraffin with Hexahedral Elements . . . . .	41
A.11	11_Mesh_Graphite_Air.py — Seed and Mesh Graphite & Air with Hexahedral Elements . .	42
A.12	12_Create_Assembly.py — Create Assembly from Individual Parts . . . . .	43
A.13	13_Create_Job.py — Create Abaqus Mesh Input File from Assembly . . . . .	44
<b>Appendix B MCNP6 Input File</b>		<b>45</b>

# 1 Introduction

The purpose of this report is to provide a tutorial for nuclear engineers, who have some experience with 3-D design software, in how to create an MCNP6 unstructured mesh model. This is not an exhaustive review of 3-D CAD/CAE design tools and techniques, but rather is intended to provide sufficient instruction for an engineer to proceed through the steps necessary to create an unstructured mesh model suitable for use in MCNP6. Furthermore, this report is intended to complement Refs. 1, 2; if the reader is not familiar with those documents, he or she is strongly encouraged to review them to better understand the unstructured mesh capabilities (and limitations) within MCNP6.

In order to motivate this tutorial, one of the Ueki fixed-source shielding benchmark experiments discussed in Ref. 3 (which is subtly different than the benchmark experiments described in Ref. 4) is modeled. This benchmark was selected for use herein because it is characterized by simple but non-trivial geometry and continues to be used to validate both transport methods (Ref. 5) and nuclear data (Ref. 6), and is thus well-studied and understood. For the purposes of this document, a 20 cm thick graphite shield is required. To the left, a paraffin block with a conical cutout houses an isotropic point Cf-252 source at (approximately) the center. To the right, two tallies are used: a volumetric track-length (F4 type) tally with a point flux detector (F5 type) type centered within. The geometric arrangement of these components is illustrated in Figure 1.

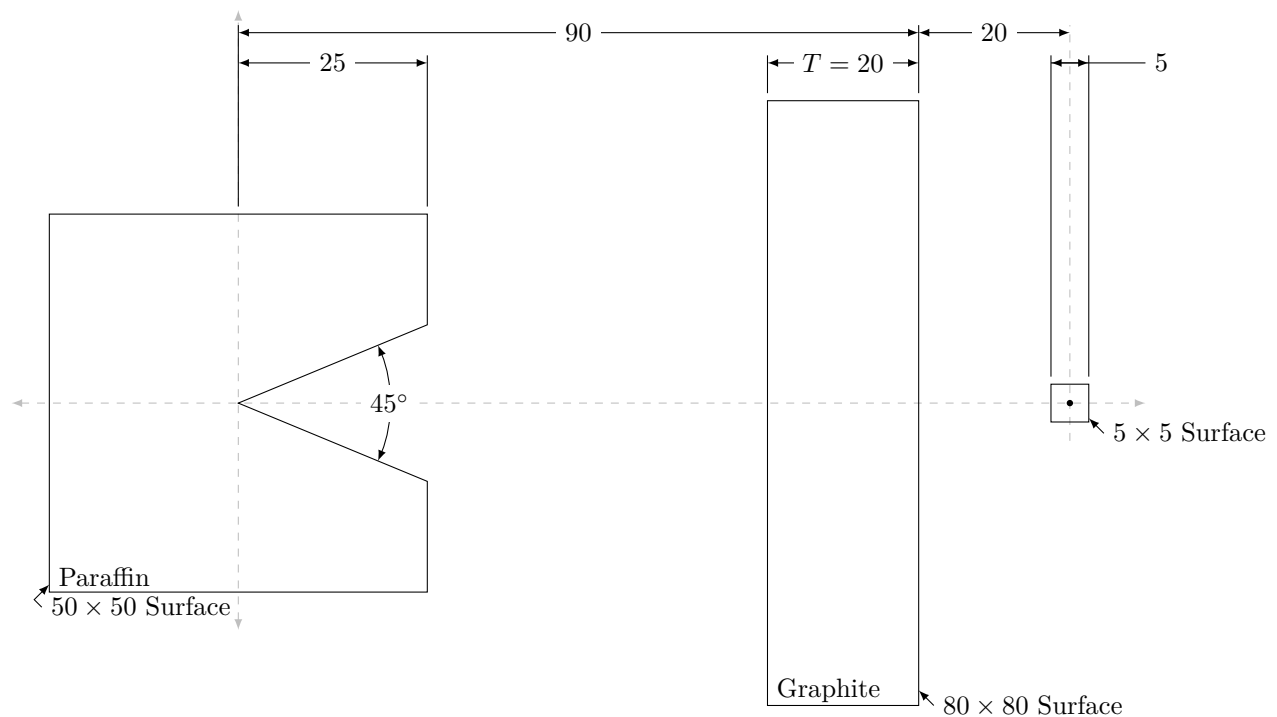


Figure 1: Schematic of Ueki Benchmark Geometry (Dimensions are Centimeters)

The reader is taken through the process of creating Abaqus Parts<sup>1</sup> (the fundamental geometric building block within Abaqus), assigning element sets to the Parts, defining materials, meshing the Parts, and combining the Parts into an Assembly for use in MCNP6. Screenshots accompany the discussion and the major steps are also captured in Python scripts (which can be directly used within Abaqus) to reproduce the work described herein. More information on these scripts is available in Appendix A. Finally, utilities provided with MCNP6 are used to build a skeleton input file from the Abaqus mesh (.inp) file, the calculation is executed, and the results examined (briefly).

## 2 Abaqus Operations

### 2.1 Running Abaqus Python Macros

Instead of manually working through all aspects of this section, we could instead use the Abaqus Python macros listed in Appendix A (concatenated together into a single file as described in Appendix A). Macros are accessed with File → Macro Manager which will display a dialog box listing macros in the Abaqus home directory and user-specified Work Directory. Macros are run by selecting the desired macro and clicking “Run.” Note that these macros depend on specific Part names, so the reader is encouraged to follow the conventions suggested herein.

### 2.2 Geometry Creation

Upon launching Abaqus/CAE, we should see a screen similar to that shown in Figure 2.

---

<sup>1</sup>Words such as Parts and Assemblies are capitalized throughout to reflect how they appear within Abaqus and to differentiate them from more generic uses of the same terms.

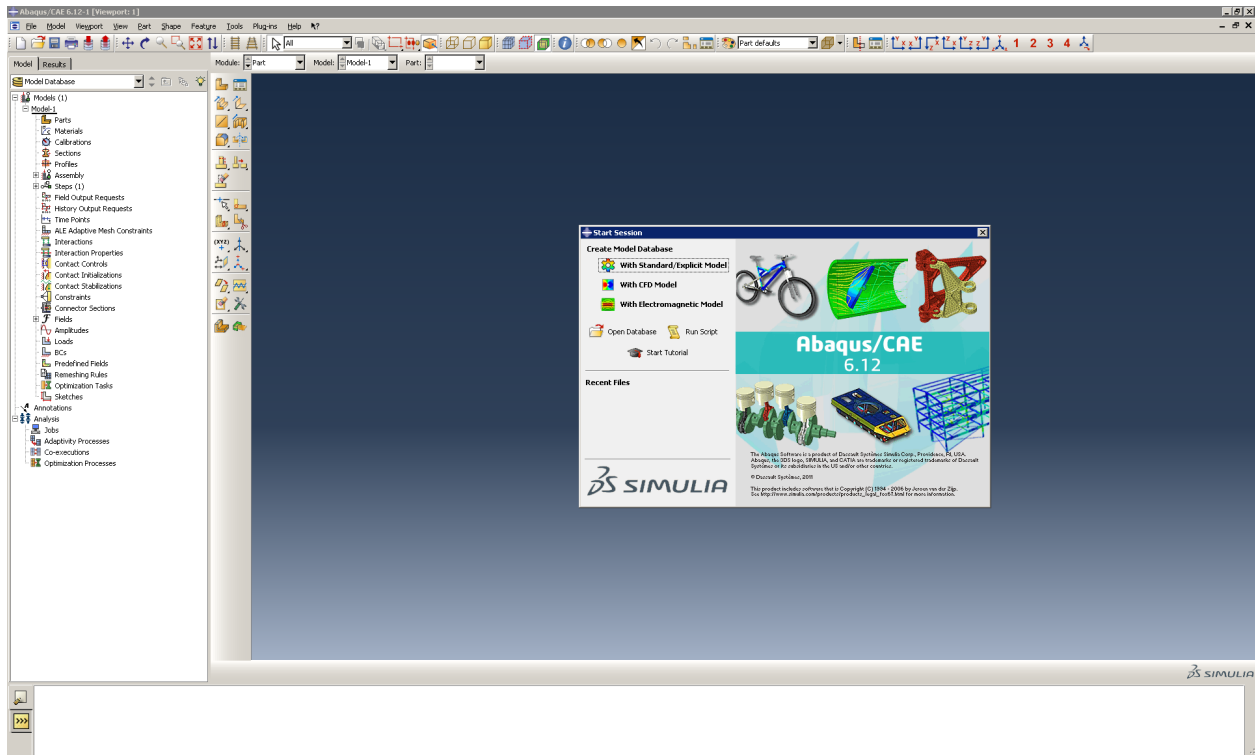



Figure 2: Abaqus/CAE 6.12 Launch Screen

Here, a “Standard/Explicit” model is used. We then save the model in a working directory and set this directory as the Abaqus Work Directory with File → Set Work Directory. Note that setting the working directory accomplishes several objectives:

- The Abaqus Macro Manager, which records and executes Python-based macros, will look in the Abaqus home directory (e.g., C:\Abaqus) and the user-specified working directory for macro definitions. As such, it is important that any macros intended for use are stored in either location, and
- When Abaqus executes a job to write the mesh input file intended for use with MCNP6, it writes it in the Abaqus home directory if no working directory is specified.

Next, we rename our model to something more descriptive (e.g., “Ueki\_20cm”) by clicking on the model name “Model-1” on the left-hand part of the screen in the model database browser, right-clicking, and selecting Rename.

We now create our first Part: the Paraffin block. We do this by clicking “Create Part” (  , on the toolbar left of the main viewport) and entering reasonable information in the dialog box that comes up (see Figure 3 for suggested parameters). For this type of Part (an extruded Part), most of the defaults are acceptable.

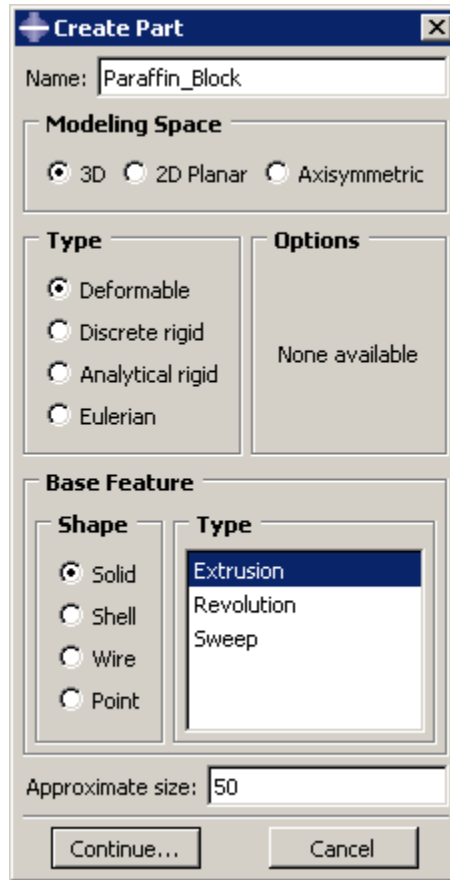


Figure 3: Create Part Dialog for Paraffin Block

From here, we sketch the Part's profile to extrude. We sketch a  $50 \times 50$  cm square centered about the origin so that it extends in both directions by 25 cm as shown in Figure 4.



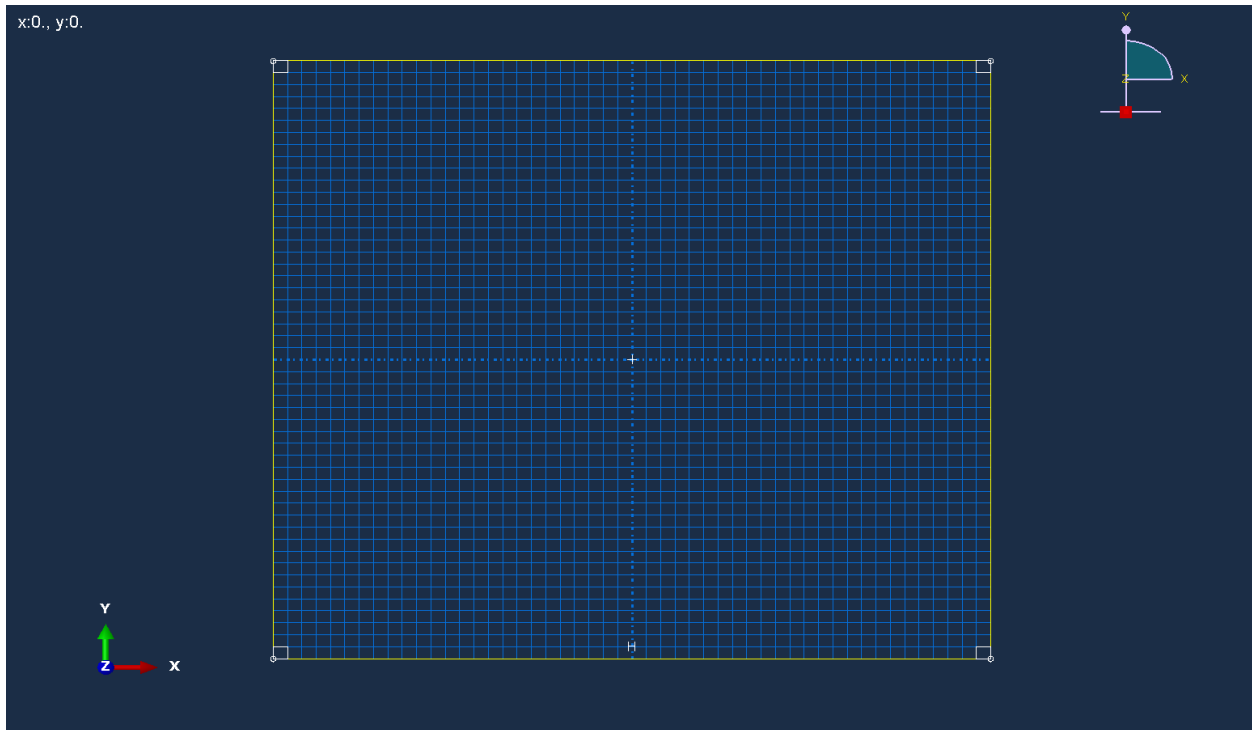


Figure 4: Layout Sketch for Paraffin Block

Once the sketch is satisfactory, click “Done” at the bottom and enter the extrusion length in the dialog box that appears (shown in Figure 5).

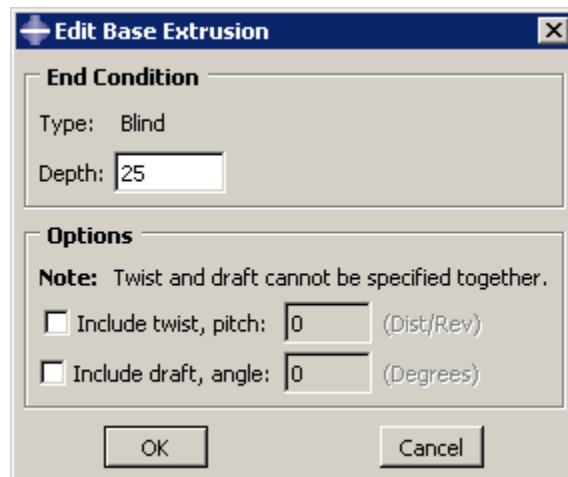


Figure 5: Create Part Dialog for Paraffin Block

Note that in Abaqus, instead of “Done” we are sometimes presented with a red “X” (✗) that, despite its ominous appearance, can produce the desired result — don’t be afraid to use it. Nevertheless, we now have a Part created that represents half of our cube because we could not extrude bi-directionally. As such, we

now extrude the face that lies along the origin with the Create Solid: Extrude (👉) function. We select the face that we want to extrude and a line that will orient the rotation (which, for our purposes here, is irrelevant). We then have a sketch appear with the existing Part geometry projected on it. As such, we create another  $50 \times 50$  square sketch using the projected geometry of the previously-created Part to snap two opposing corner points. After finishing the sketch, we can enter the extrusion distance (25 cm) in the dialog box as shown in Figure 6 and should see a  $50 \times 50 \times 50$  cube as shown in Figure 7.

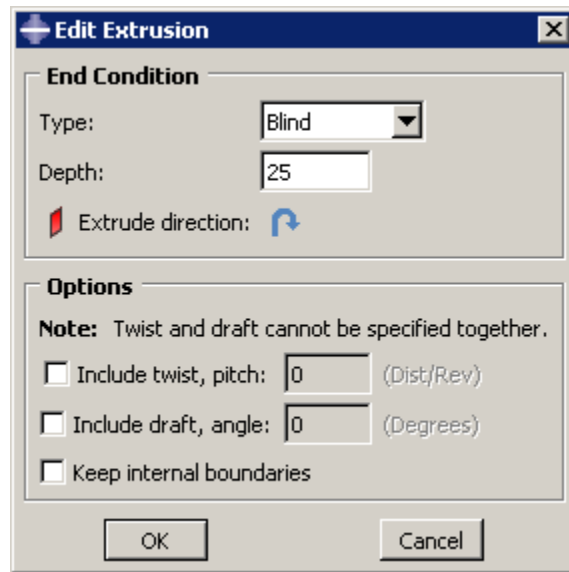


Figure 6: Create Part Dialog for Paraffin Block

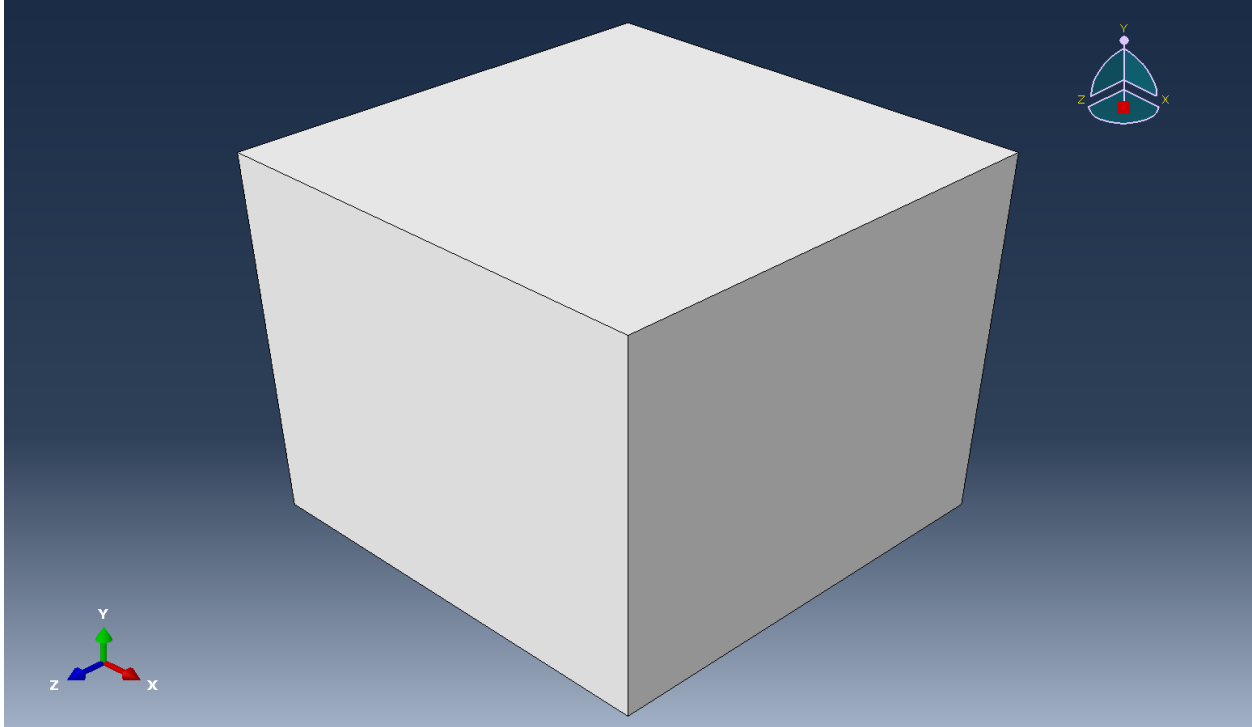


Figure 7: Final Extruded Paraffin Cube

These same operations can be performed with the macro `O2_Create_Paraffin_Block`. Note that dimensions can be verified by using `Tools → Query → Distance` and selecting corner vertices where the coordinates and distances will be printed in the status window at the bottom of Abaqus.

Alternatively, we could have extruded by 50 cm, when the part was created initially, and then translated the Part by 25 cm when instanced into the Assembly to center it at the origin.

### 2.3 Part Cutting Operations

We have now created our first Part. It is left as an exercise to create a cutting Part to remove a cone of material from the paraffin block (hint: use a revolve operation to create the cutter Part by revolving a triangle). We use the two Parts to cut out the cone with the "Cut geometry" operation. To do this, we right-click on `Assembly → Instances` in our model tree and select "Create" and select the two Parts that we just made; these appear in the construction area. We then choose "OK" to place the Instances. See Figure 8.

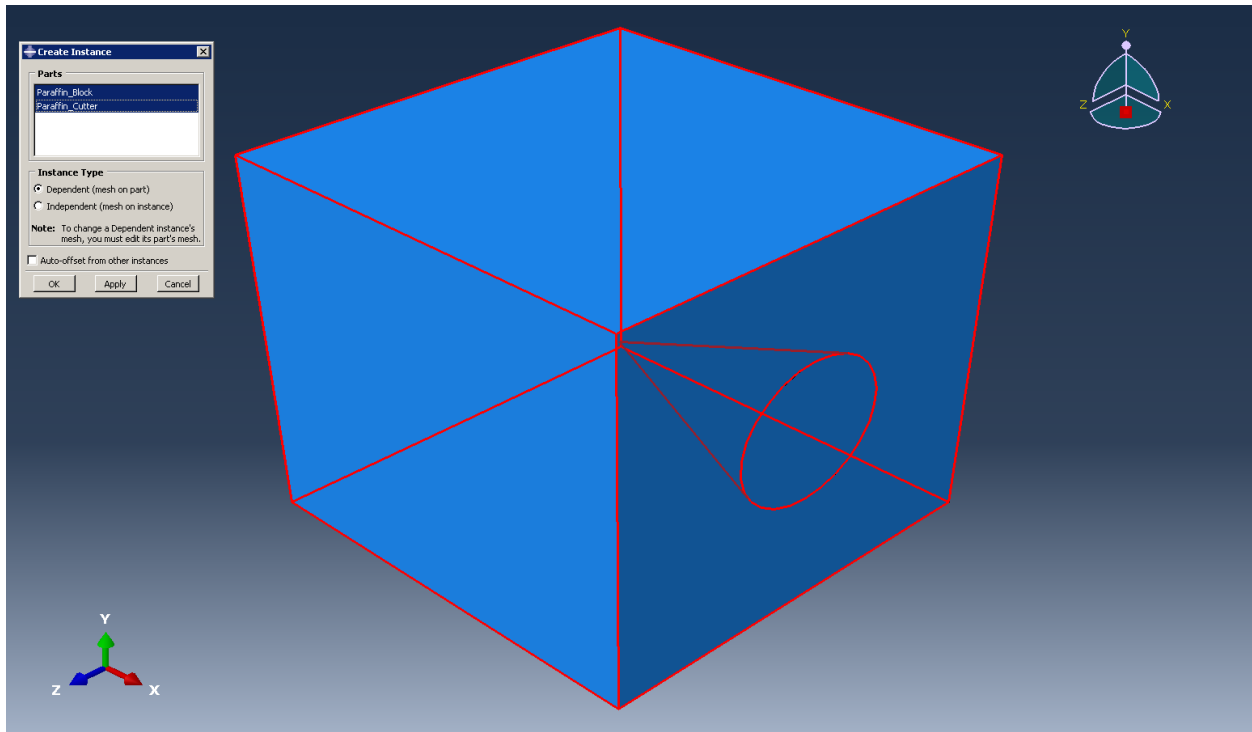



Figure 8: Placed Instances of Parts in an Assembly

We now select Merge/Cut Instances () , name the new Instance, choose to perform a “Cut” and to delete the original Instances, and click OK (see Figure 9).

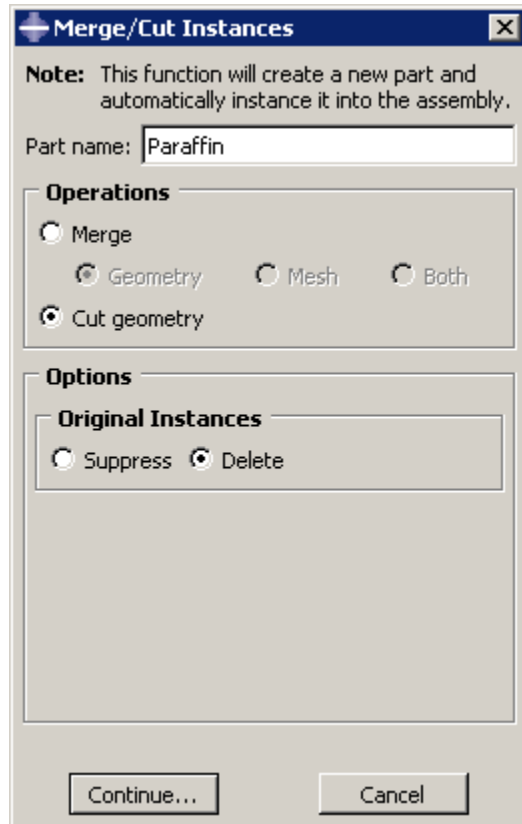


Figure 9: Merge/Cut Dialog for Paraffin Block

We then select the Instance to be cut (i.e., the block) and the Instance(s) to perform the cutting (i.e., the cone). After clicking “Done,” we are left with our cut block which has become a new Part with the name we gave it (see Figure 10).

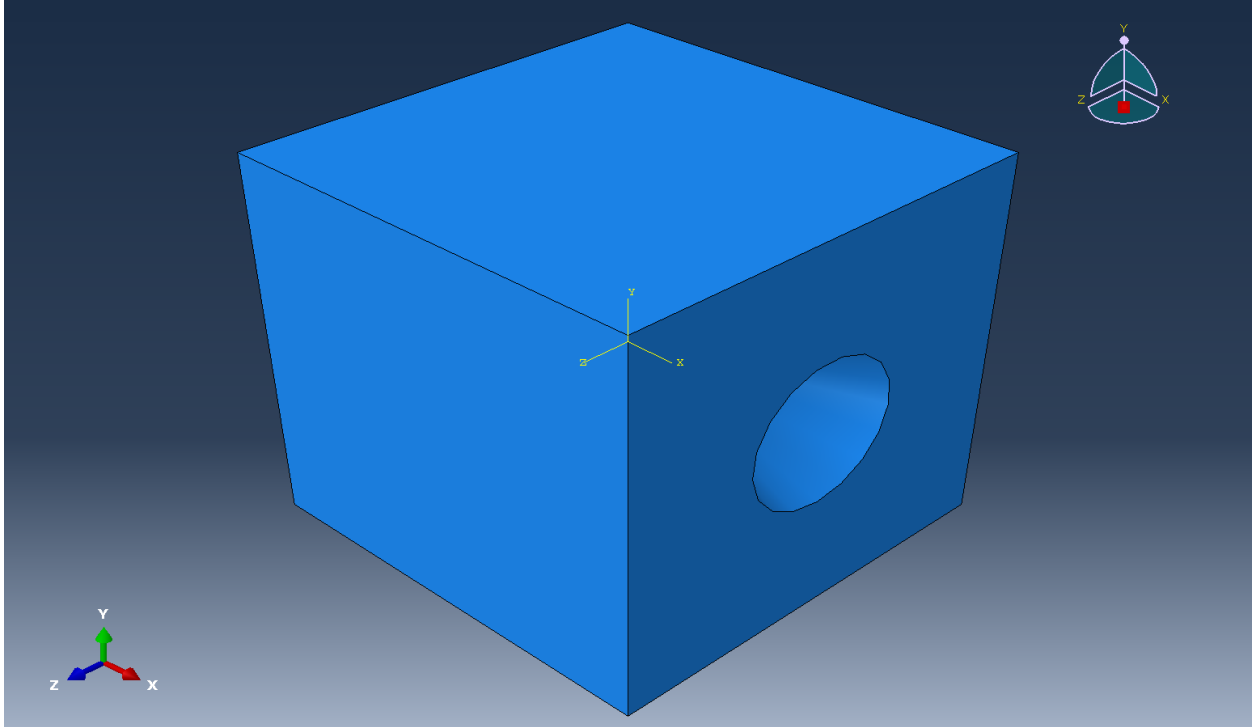


Figure 10: Final Paraffin Part

The creation of the cut action can be performed with the macros `03_Create_Cutter` and `04_Cut_Paraffin`, respectively.

It is left as an exercise to create the graphite shield and an air (detector) cube using similar operations so that there are three Parts (named: Paraffin, Graphite, Air). When creating these Parts (which are not centered about the origin in the final arrangement), we have two options:

- Create the sketch and extruded geometries centered about the origin, or
- Create the sketch and extruded geometries in their final position.

Both of these approaches have benefits and drawbacks, but this document assumes all geometry is made using the latter approach (i.e., with Parts created in their final position relative to the global origin). The graphite and air Parts can be created using the macros `05_Create_Graphite` and `06_Create_Air`, respectively.

## 2.4 Assignment of Element Sets (elsets)

We now need to associate element sets with the geometry so that MCNP6 understands which elements correspond to which materials, tallies, and sources. To do this, we begin with the Paraffin, expand it in the tree, right click “Sets,” and select “Create.” Assign a set named “Set\_tally\_material\_1” and click “Continue...” to select the (entire) Part with a click-and-drag “lasso”; finally, click “Done.” We just assigned a tally and material, both numbered 1, to the Paraffin Part. We perform similar operations with the Graphite and Air Parts, but increment the number assigned for both the tally and material. This should give us the following three elsets:

1. Set\_material\_tally\_1 (assigned to the Paraffin),
2. Set\_material\_tally\_2 (assigned to the Graphite), and
3. Set\_material\_tally\_3 (assigned to the Air).

Some notes regarding elsets:

1. Care should be taken when assigning elset names. For the material number, use the material number that is used in the MCNP6 data cards. If there is only one tally region in a part, its number is immaterial. However, if there is more than one tally region, make the tally numbers unique. This process implies the need for multiple elsets. More details regarding elset naming and assignment are available in Ref. 2.
2. The elsets assigned within this section were assigned before the Part was meshed. As such, we had a relatively simple Set assignment dialog box to use which only required us to enter a name and click “Continue...” to select the appropriate geometry. However, if the Part was already meshed, a more complicated Set assignment dialog box is presented as shown in Figure 11. The correct option to use is “Geometry” when assigning elsets to a Part after it has been meshed.

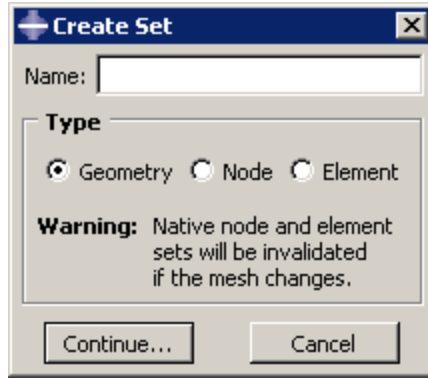


Figure 11: Post-Meshing Set Assignment Dialog Box

The macro `07_Assign_Elsets` can be used to assign the appropriate element sets for all three Parts.

## 2.5 Definition of Materials

With our element sets assigned, the next step is to create materials that correspond to the Paraffin, Graphite, and Air. For the Paraffin, we right-click on “Materials” in the Model Tree and click “Create.” We name the material “Material\_Paraffin\_1” (to tie the material number 1 to the element set assigned previously). We also set the density with General → Density. Note that when entering the density, one should use the MCNP6 sign conventions (i.e., a negative number represents a mass density and a positive number represents an atom density)<sup>2</sup>.

---

<sup>2</sup>If the user intends to perform subsequent thermal-mechanical calculations, then Abaqus requires material definitions with positive densities.



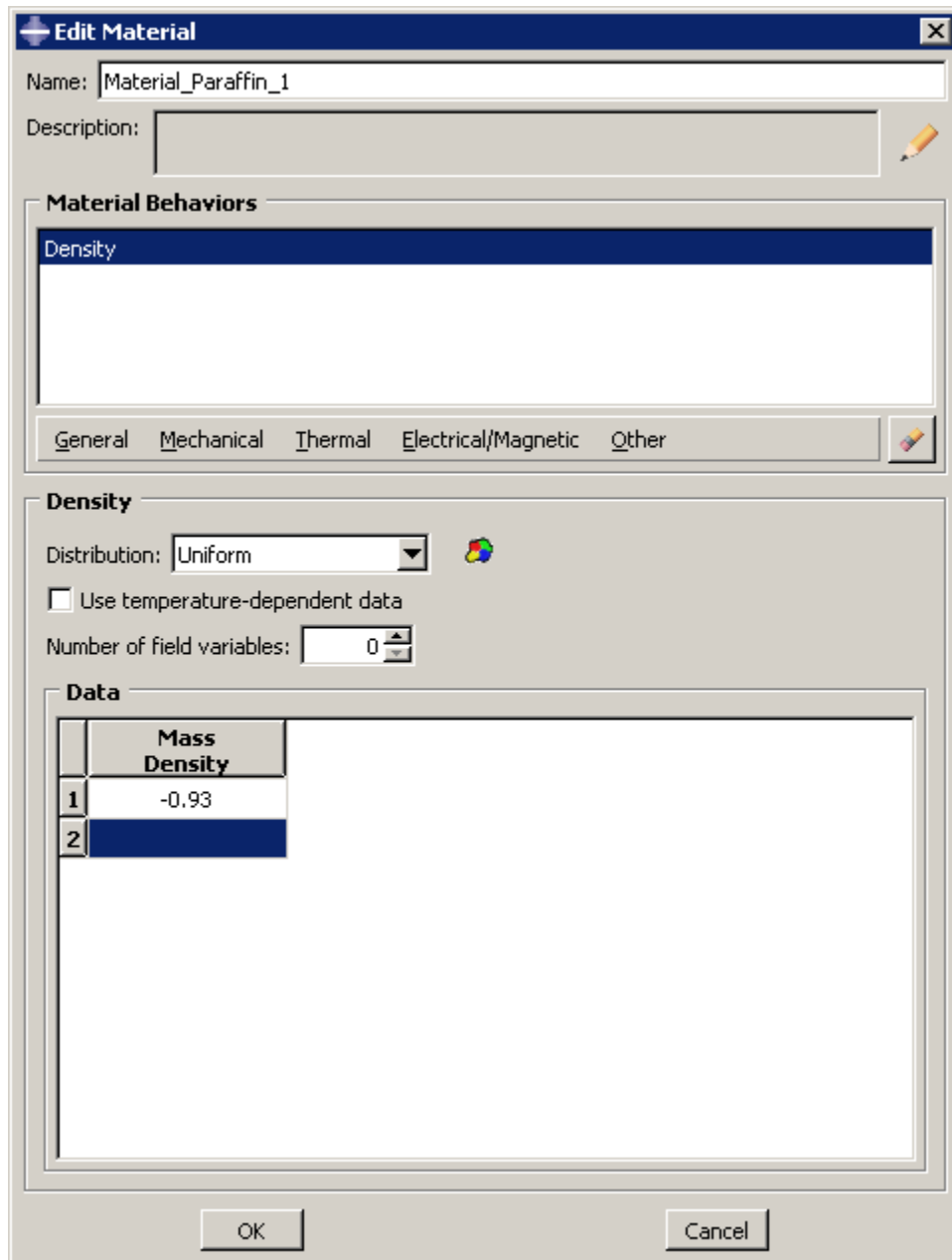


Figure 12: Paraffin Material Definition

See Figure 12. The material densities in use are:

- Paraffin: (-)0.93 g/cc,
- Graphite: (-)1.7 g/cc, and
- Air: (-)0.001205 g/cc.

When finished, click OK and create material definitions for the Graphite and Air in a similar way. Instead, the macro `08_Create_Materials` can be used to create the materials with the appropriate density values assigned.

## 2.6 Mesh Parts

Next, each part needs to be meshed. Because the Parts generally fit neatly on a Cartesian grid, we choose to mesh them with hexahedral elements. To do this, we can right-click on the Paraffin and choose “Make Current.” Then, using the Module drop-down, choose “Mesh.” We should see the Paraffin change color to become yellow (see Figure 13).

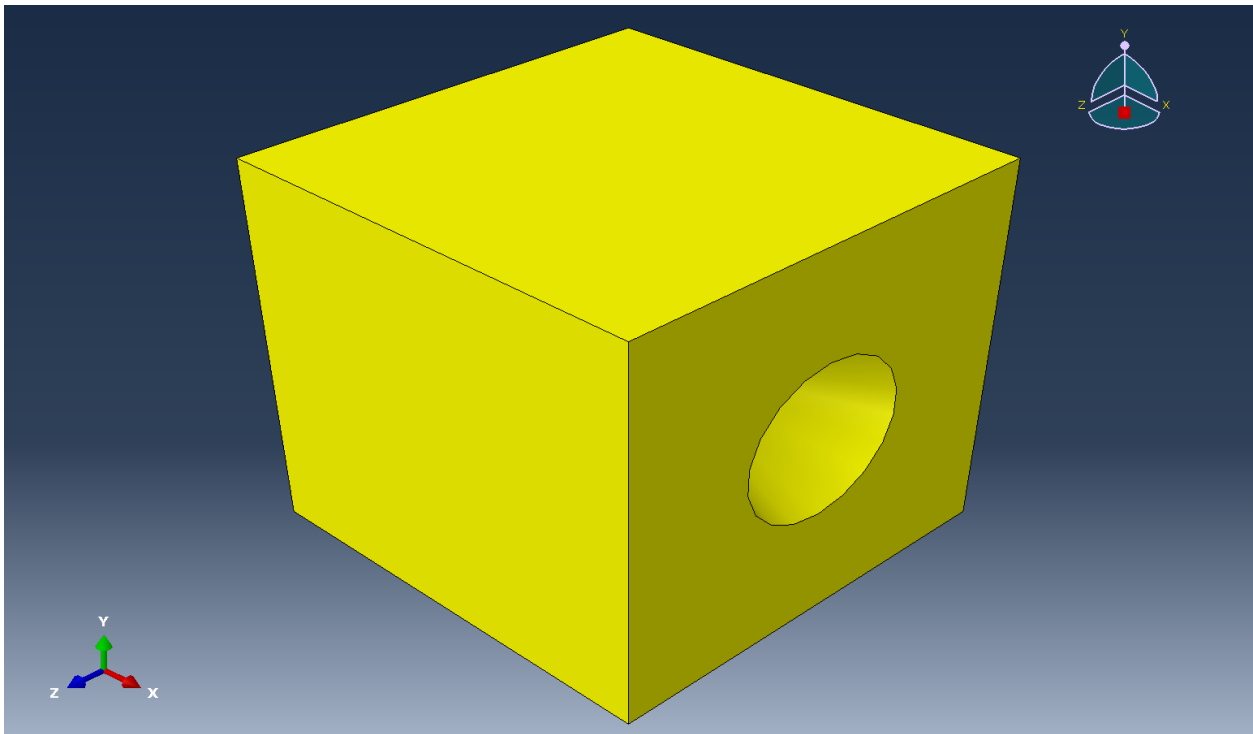




Figure 13: Paraffin Meshability: Yellow

Abaqus uses colors to indicate what ability it has to mesh the Part given the current options (where the options are available in Mesh → Controls). To mesh the Part, we want it to display “green” and not “yellow.” Yellow often works but may not produce the desired result. Orange is bad — a sign that partitioning is needed. We create a better meshing configuration by partitioning the Part using datum planes.

We will create two datum planes using the Create Datum Plane function (). Click the button, select the XY Plane, and enter a distance of zero. A plane should appear bisecting the paraffin. Repeat the operation

for the XZ plane and click the red “X” () to end the creation of cutting planes.

Next, we will partition the cell using these new datum planes. Use the “Partition Cell: Use Datum Plane” function accessed by left-clicking and holding on “Partition Cell: Define Cutting Plane” and moving to the right to select the appropriate operation (, also accessible by left-clicking on the black triangle in the lower right-hand corner). When selected, because Abaqus is already aware of the Part selected, we need only select the appropriate datum plane. Then immediately re-select the entire Part and the next datum plane. When finished partitioning, a green Part should appear (see Figure 14).

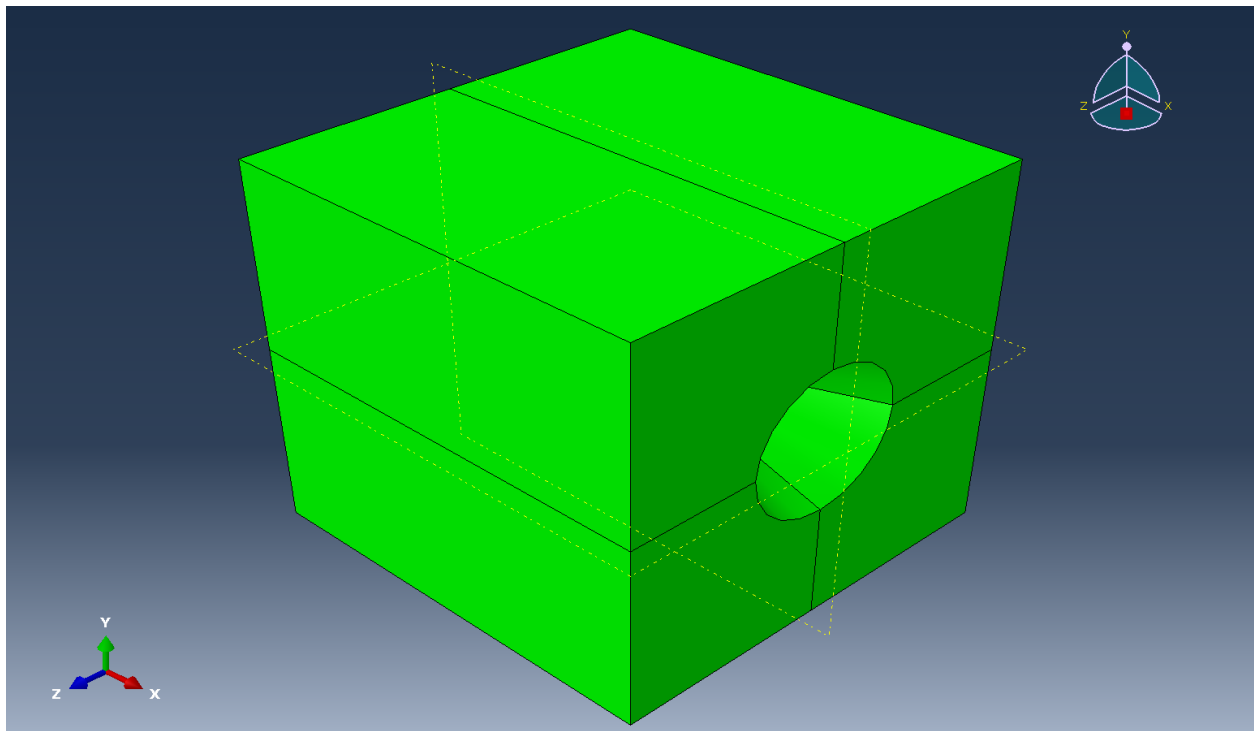


Figure 14: Paraffin Meshability: Green

The datum plane creation and partition operations can also be performed using the macro `09_Partition_Paraffin`.

To mesh the Part, first seed it with mesh points using the Seed → Part dialog; choose a global size of  $5 \text{ (cm)}^3$ . See the dialog box in Figure 15 and the seeded Part in Figure 16.

---

<sup>3</sup>The Abaqus dimensions are effectively unitless — they are what the user intends them to be. As such, we specify everything in centimeters for MCNP6.

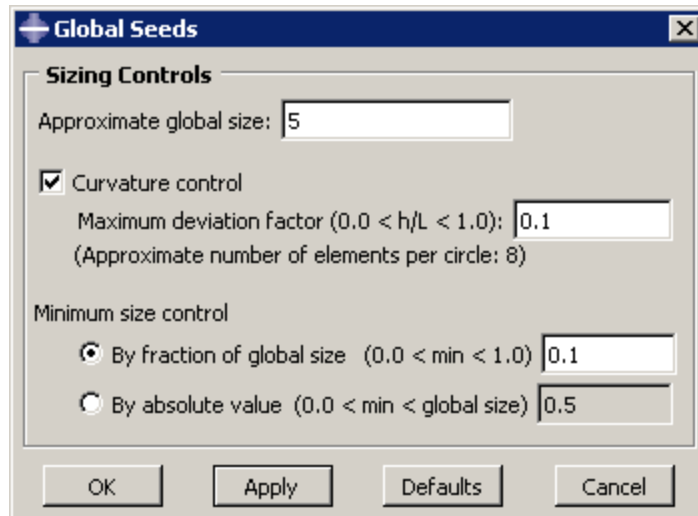


Figure 15: Seeding Parameters

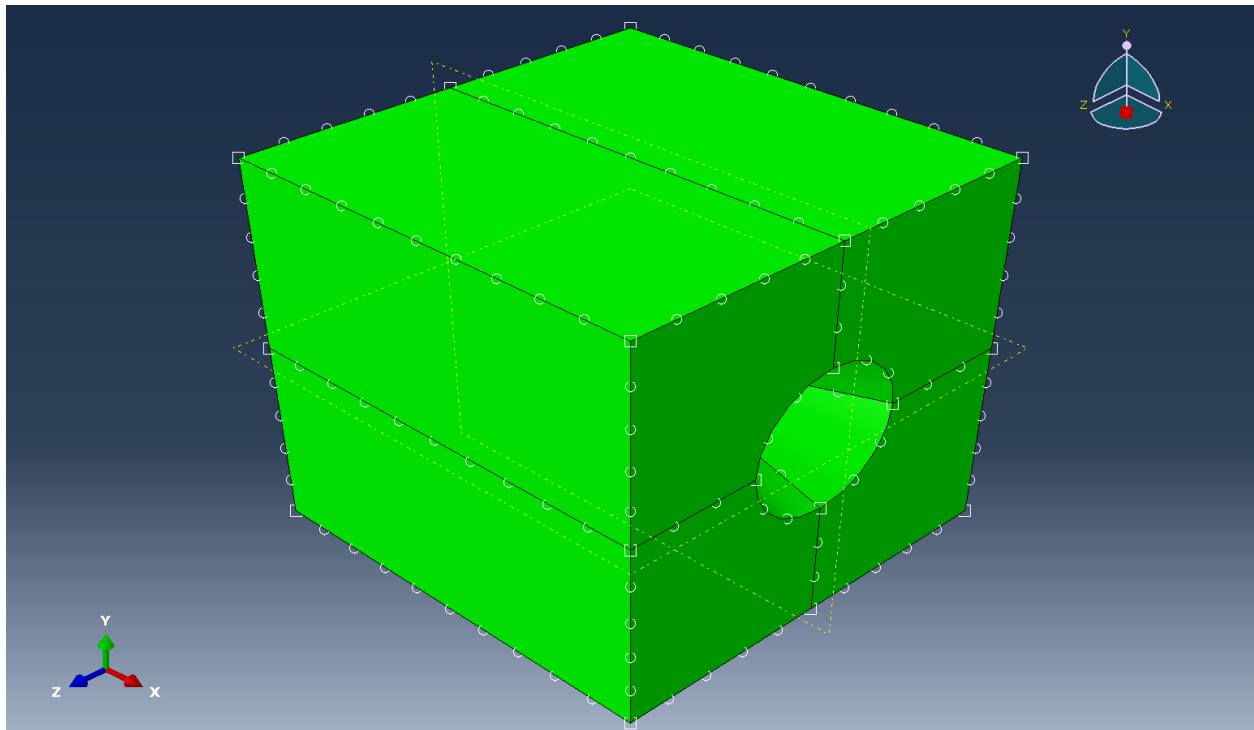


Figure 16: Seeded Paraffin

Finally, select Mesh → Part and confirm “Yes” that we want to mesh it. The Part meshed using hexahedral elements (see Figure 17) appears.

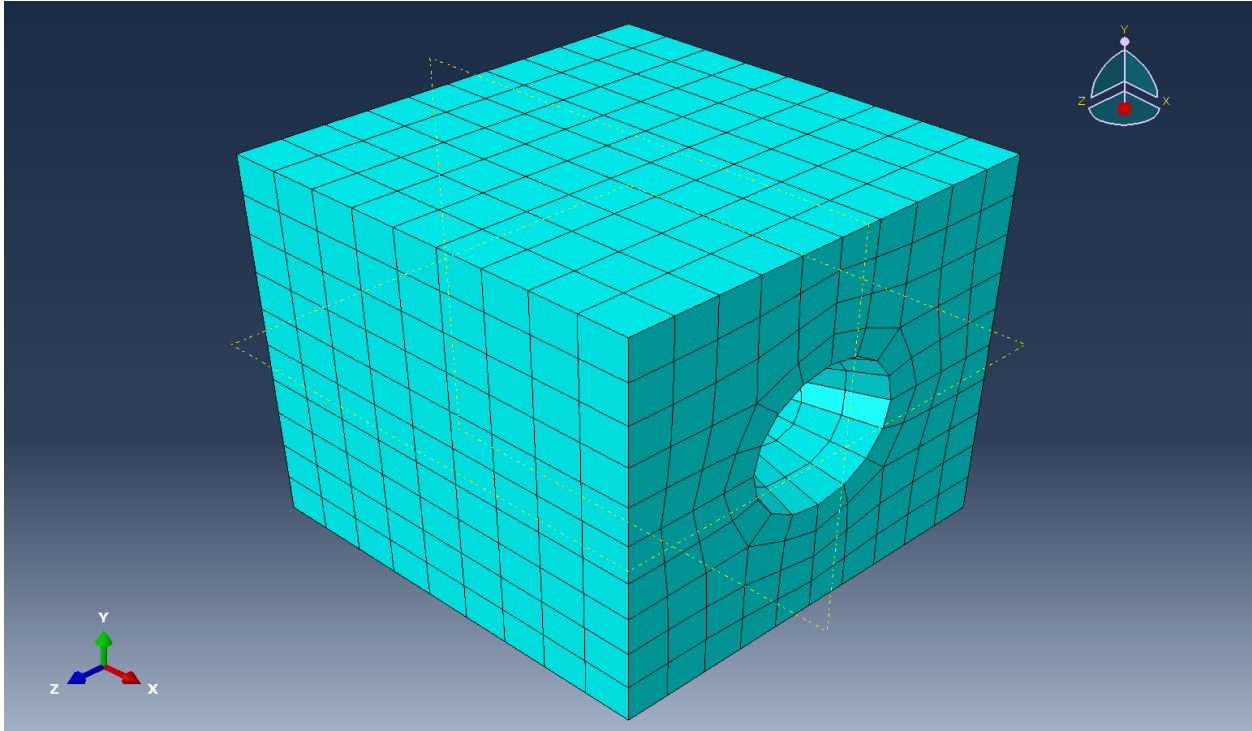


Figure 17: Meshed Paraffin

Seeding and meshing can also be performed using the macro `10_Mesh_Paraffin`.

The same seeding and meshing operations for the graphite and air are left as exercises to the user noting that as purely Cartesian elements, no partitioning is necessary. These components can be meshed with the macro `11_Mesh_Graphite_Air`.

Some notes regarding meshing:

1. When a part is meshed, a lasso operation over some or all of the elements will highlight the vertices and edges lassoed. This is a good way to view and/or confirm edge-center vertices associated with quadratic elements. These vertices are not otherwise visible.
2. Meshing serves multiple purposes for a MCNP6 practitioner. First, the mesh is used to (accurately) represent the geometry of interest. Because the mesh is constructed using linear edges, curved surfaces are approximated and it is up to the user to determine what degree of approximation is appropriate. Next, results on the mesh can be visualized. As such, the user might wish to make the mesh finer than would otherwise be necessary to get a more pleasing and/or informative visualization of the edit information, particularly where high flux gradients are expected.

## 2.7 Create Assembly

Next, we insert Instances, one of each, of the three Parts that were created. It may be necessary to delete (or suppress) any other Instances already present. Regardless, once inserted (and perhaps repositioned depending on whether absolute or relative coordinates were used in Part creation — see note at end of Subsection 2.2), one should see the Parts arranged as shown in Figure 18. Furthermore, one can change the palette to be based on “Sets,” rather than “Assembly defaults,” and change the Module dropdown to “Mesh” to see something similar to Figure 19.

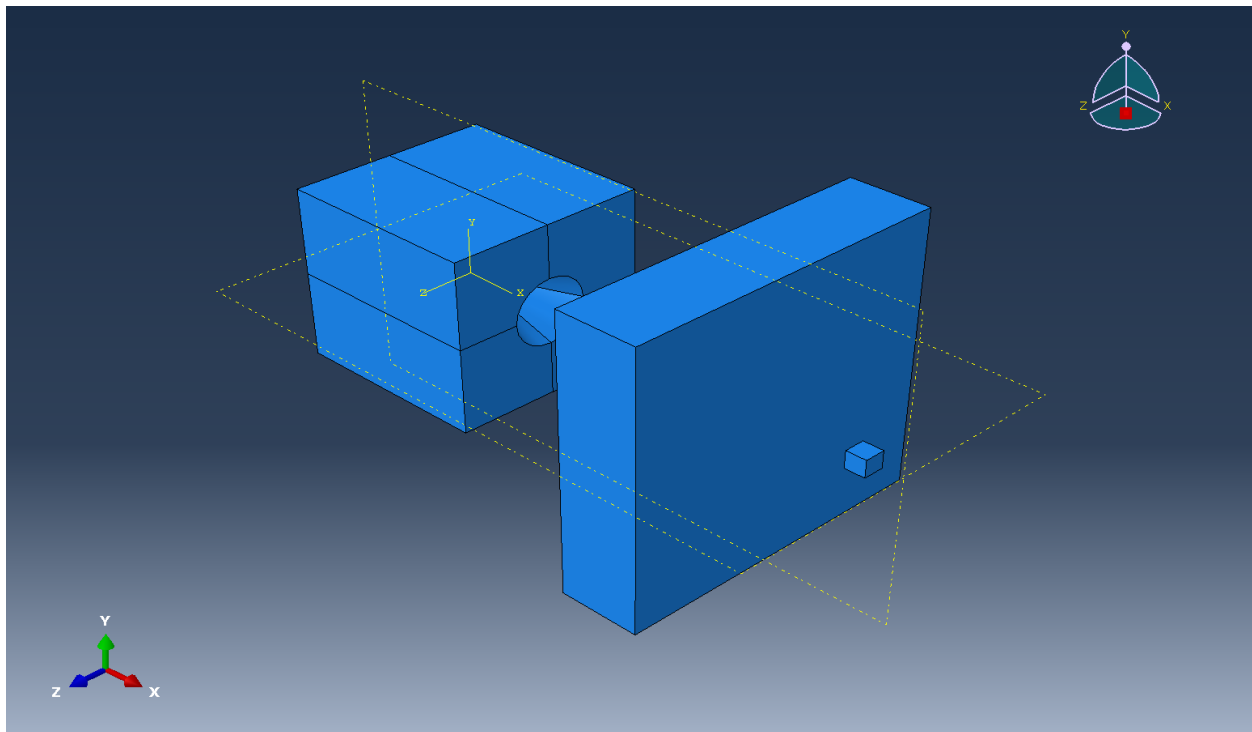


Figure 18: Assembly, Default Coloring

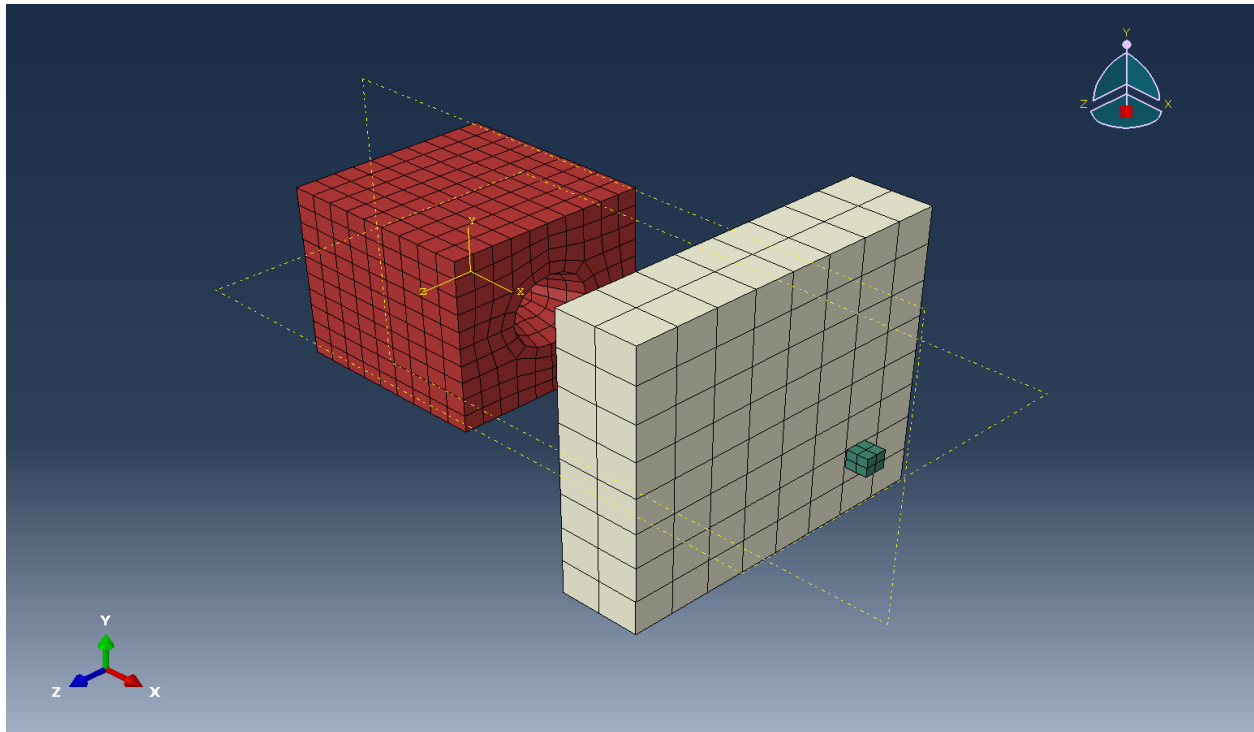


Figure 19: Assembly, Set-based Coloring with Mesh Shown

This confirms that the sets assigned uniquely to each Part have been brought through the meshing and assembly creation processes. The Assembly can also be created with the macro `12_Create_Assembly` noting that adjusting the view following the macro will likely be necessary.

## 2.8 Write Abaqus Mesh File

With the Assembly created and with all Parts correctly positioned, we can now create a “Job” to write the input file. From the Assembly view, select Job → Manager → Create and name the job (which will become the file name created, e.g., `um_ueki_20cm`, where `.inp` will be appended by Abaqus). Click “Continue” and give the job a description (which is reused as the case title in MCNP, e.g., Ueki Benchmark, 20 cm Graphite). Click “OK” and then in the Job Manager click “Write Input.” Having set the Work Directory early on, this file should be created where that work directory was specified. Otherwise, it will be created in Abaqus’s default working location. Once the file is created, open it to see that it was written fully. However, there should be nothing more required from the user within Abaqus at this point.

### 3 Pre-MCNP6 Execution Operations

MCNP6 ships with several utilities intended to make working with unstructured mesh files more convenient. In order to create a skeleton MCNP6 input file, we can use the `um_pre_op` utility (which must be in the user's `PATH` environment variable) with the following command:

```
1 um_pre_op -m um_ueki_20cm.inp -b 3 -ex mcnpinp
```

which will then create the MCNP6 skeleton input file shown below.

```
1 Ueki Benchmark, 20 cm Graphite
2 c
3 c Created from file      : um_ueki_20cm.inp
4 c Created on           : 6-25-2015 @ 7:50:46
5 c
6 c
7 c PSEUDO CELLS
8 1      3      -1.205000E-03  0  u=1
9 2      2      -1.70000      0  u=1
10 3      1      -0.930000     0  u=1
11 4      1      -1.205000E-03  0  u=1
12 c
13 c LEGACY CELLS
14 5      0              -99  fill=1
15 6      0              99
16
17 c
18 c SURFACES
19 99 sph  4.37500E+01  0.00000E+00  0.00000E+00  1.06837E+02
20
21 c
22 c DATA CARDS
23 embed1 meshgeo=abaqus
24      mgeoin=um_ueki_20cm.inp
25      meeout=um_ueki_20cm.eeout
26      length= 1.00000E+00
27      background=      4
28      matcell= 1 1 2 2 3 3
29 c
30 c
31 c
```

In this file, we can see that the Parts we defined previously have corresponding pseudo cells with the material numbers and densities assigned within Abaqus. We now need to transform this skeleton file into a functional MCNP6 input file, primarily by adding appropriate data cards. These details are omitted, but the final



MCNP6 input file is given in Appendix B. However, one note regarding the transformation from skeleton to a functional MCNP6 input: the background (Cell 4) material number was set on the command line but the fill (Cell 5) cells is voided by default (because the background cell material number was not defined in the Abaqus mesh .inp file). Not populating the background cell with the appropriate material is an easily overlooked step. More details on `um_pre_op` can be found in Ref. 2.

## 4 MCNP6 Execution Notes

When executing MCNP6, no special precautions are needed, per se. Naturally, every computer system and organization will have a different way of executing MCNP6 so it is left to the reader to follow-up with his or her system administrator to determine how best to run (e.g., what the procedures and “rules” are for running MCNP6). Moreover, we only need to make sure that the `mgeo.in` file (i.e., the Abaqus mesh file) is with the MCNP6 conventional input file and that write permission is available to create the various working files (e.g., `runtpe`, `outp`, and `eeout` files).

## 5 Results Processing

Having run MCNP6 with the input file shown in Appendix B, in addition to the traditional `outp` and `runtpe` files, we find an `eeout` file that corresponds to unstructured mesh edit output. This file contains a generic description of the unstructured mesh geometry, results, and other information important to MCNP6 when performing a continue run. More information on the `eeout` file is available in Reference 2. We also requested a `gmv` file (not produced by default) which can be used to visualize the unstructured geometry.

### 5.1 Geometry Visualization

The unstructured mesh geometry and materials can be easily visualized using the application `gmv` (or an equivalent application that can read `gmv`-formatted files). Using `gmv` is straightforward and will not be discussed in detail. However, after reading the `um_ueki_20cm.gmv` file in `gmv`, enabling the display of Edges, and clicking apply, we see the geometry and materials shown in Figure 20.

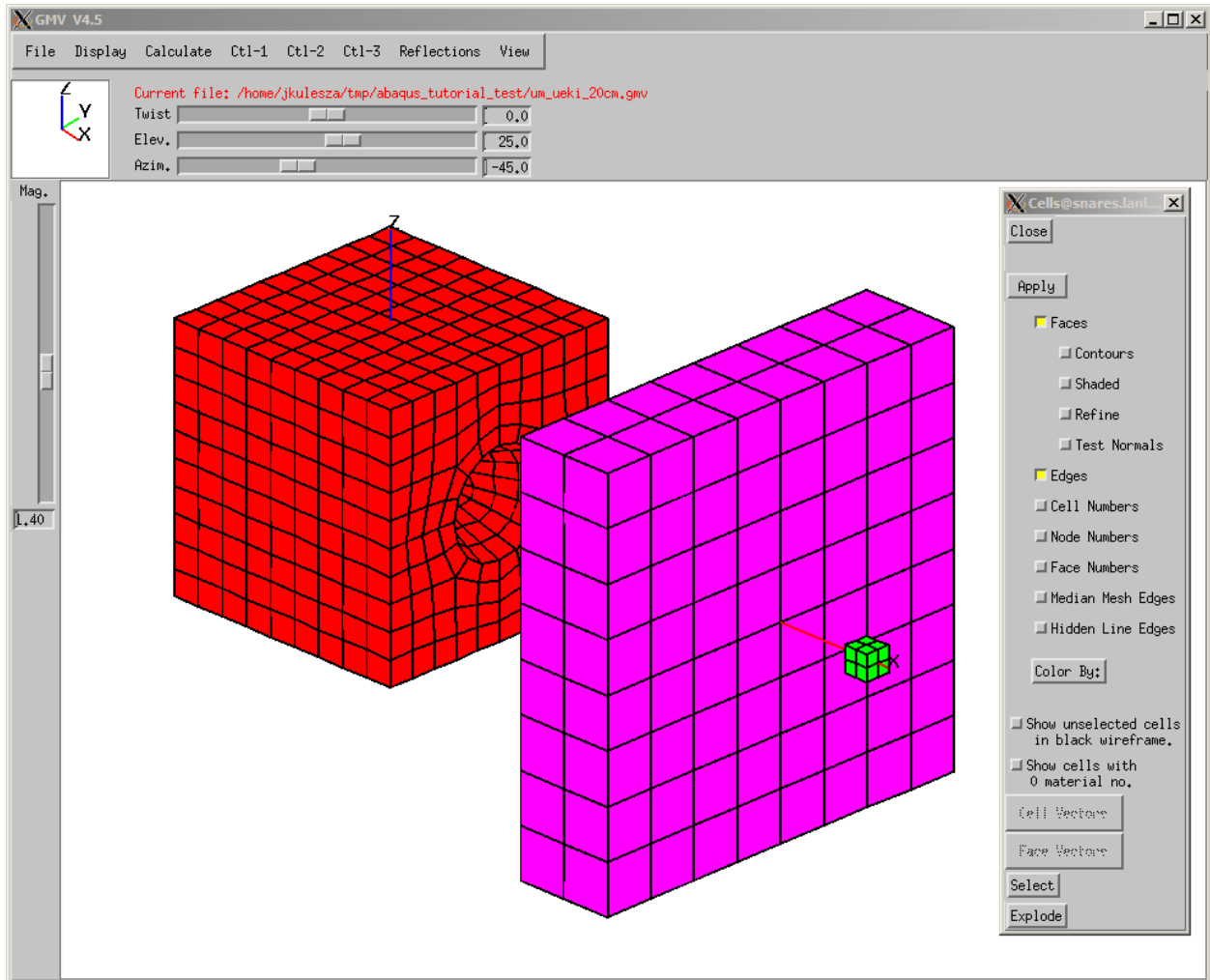


Figure 20: GMV Display of Unstructured Geometry and Materials

Unfortunately, it is not currently possible to visualize results within `gmv`.

## 5.2 Unstructured Mesh Results Visualization

One method to visualize results with the unstructured mesh geometry is to convert the `eeout` results to an Abaqus `.odb` file. This conversion process will not be described herein, and it is assumed that the user has performed this conversion before proceeding.

We first load our `.odb` file into Abaqus with `File` → `Open`, change File Filter to “Output Database (\*.odb)”, navigate to and select the `.odb` file, and click `OK`. The Assembly with a visible mesh is loaded as shown in Figure 21.

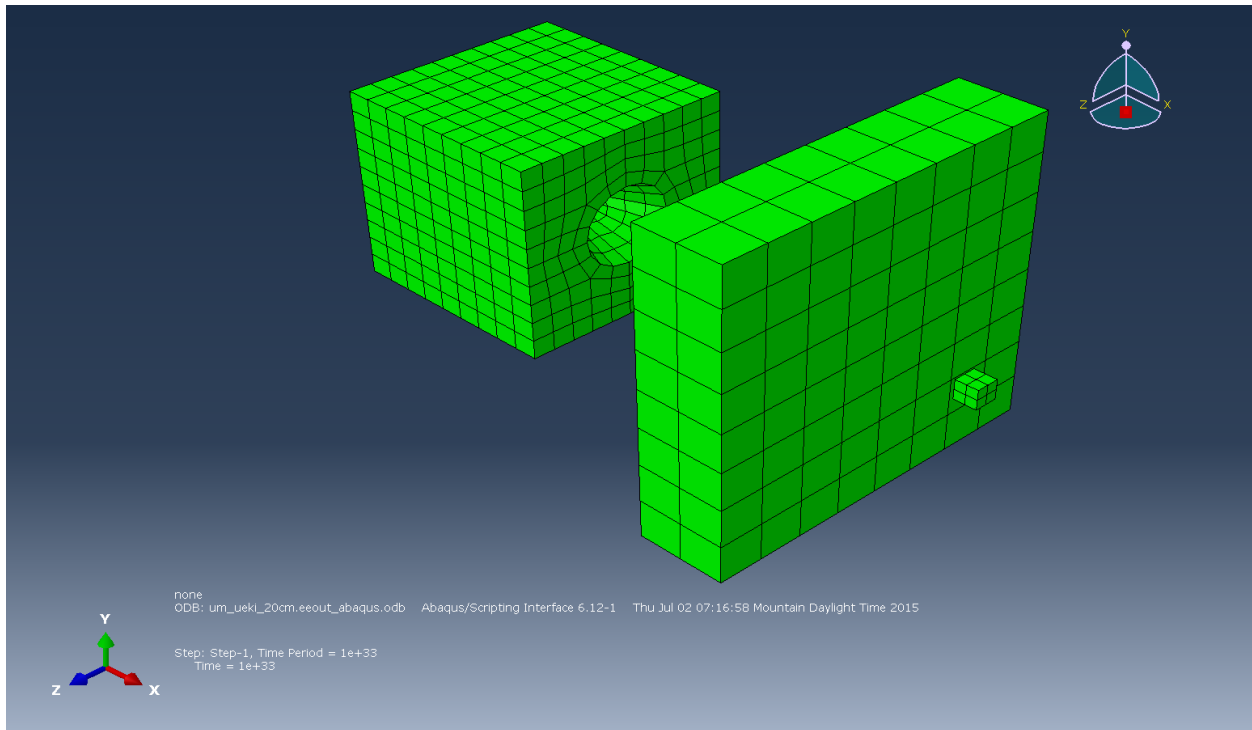



Figure 21: Results Assembly and Mesh

Use the “Plot Contours on Undeformed Shape” function accessed by long-left-clicking on “Plot Contours on Deformed Shape” and moving to the right to select the appropriate operation (). The assembly becomes blue and a colorbar appears to indicate the contour levels (which are not apparent yet). See Figure 22.

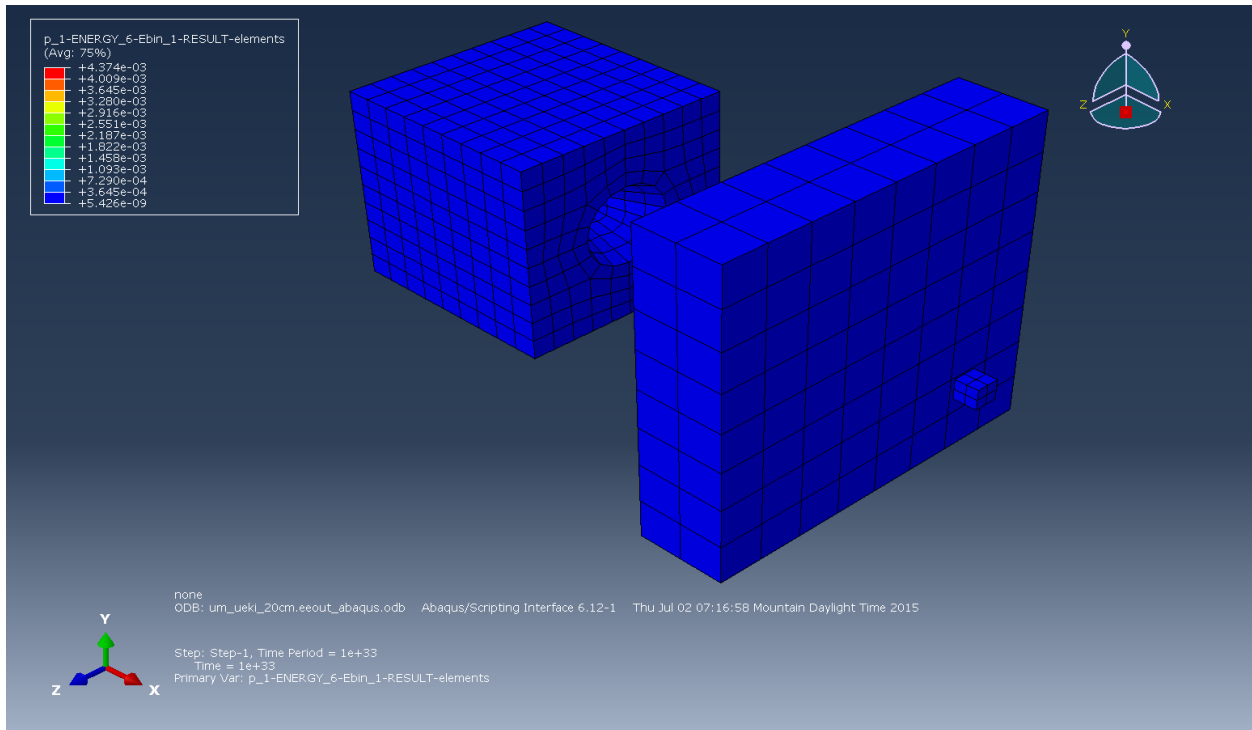



Figure 22: Default Results Display

To make the contours become recognizable, we adjust the Contour Options (  ) wherein we change Interval Type from “Uniform” to “Log” and click OK. The result is shown in Figure 23.

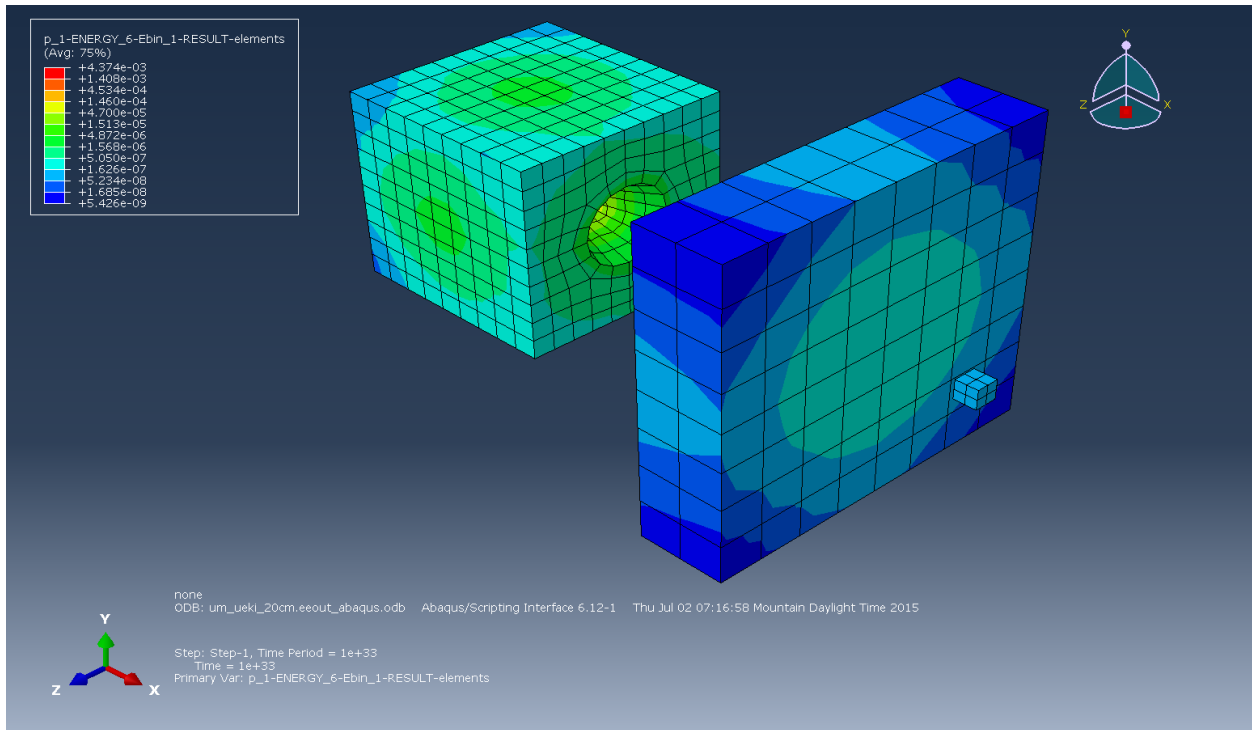


Figure 23: Final Type 6 Edit Appearance

Note that the label above the colorbar indicates that these results correspond to the type 6 edit specified with the **embee** card. To change to the type 4 edit, we select the edit from the dropdown menu in the toolbar (see Figure 24).

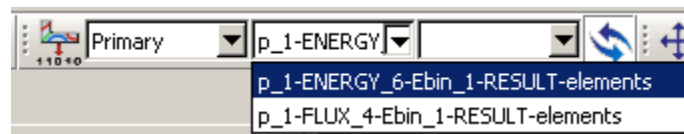


Figure 24: Dropdown to Select Edit for Visualization

Having done this, the viewport updates to display the type 4 edit results as shown in Figure 25.

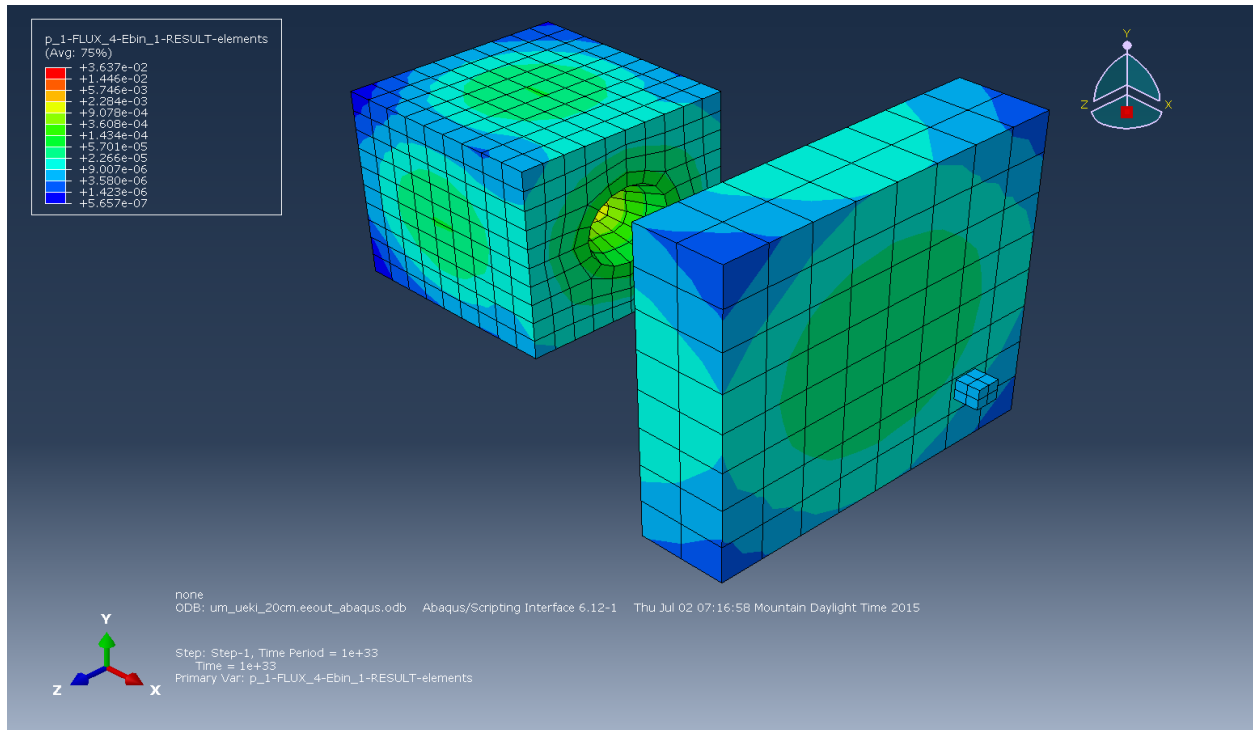


Figure 25: Final Type 4 Edit Appearance

## 6 Conclusions

It is hoped that this document serves as an introductory functional guide to the reader for how to approach unstructured mesh models created within Abaqus destined for use within MCNP6. The major operations have been defined in terms of geometry creation, elset assignment, material definitions, meshing, and ultimately generating the unstructured mesh input file. This file was then processed with `um_pre_op` to create a skeleton MCNP6 input file which the user must then modify to incorporate the proper background/fill materials as well as the appropriate data cards.

## Acknowledgements

The authors wish to thank Karen C. Kelley for a thorough technical review of this document and, particularly, the appendices.

## References

- [1] R. L. Martz and D. L. Crane, “The MCNP6 Book on Unstructured Mesh Geometry: Foundations,” Tech. Rep. LA-UR-12-25478 Rev 1, Los Alamos National Laboratory, Los Alamos, NM, USA, 2014.
- [2] R. L. Martz, “The MCNP6 Book on Unstructured Mesh Geometry: User’s Guide,” Tech. Rep. LA-UR-11-05668 Rev 8, Los Alamos National Laboratory, Los Alamos, NM, USA, 2014.
- [3] K. Ueki, A. Ohashi, and Y. Anayama, “Neutron Shielding Ability of KRAFTON N2 — Mannan — KRAFTON N2 Sandwich-type Materials and Others.,” in *Radiation Protection and Shielding Division Topical Meeting*, (Pasco, WA, USA), American Nuclear Society, April 26 – May 1, 1992.
- [4] K. Ueki, A. Ohashi, N. Nariyama, S. Nagayama, T. Fujita, K. Hattori, and Y. Anayama, “Systematic Evaluation of Neutron Shielding Effects for Materials,” *Nuclear Science and Engineering*, vol. 124, pp. 455–464, 1996.
- [5] S. W. Mosher, A. M. Bevill, S. R. Johnson, A. M. Ibrahim, C. R. Daily, T. M. Evans, J. C. Wagner, J. O. Johnson, and R. E. Grove, “ADVANTG — An Automated Variance Reduction Parameter Generator,” Tech. Rep. ORNL/TM-2013/416, Oak Ridge National Laboratory, Oak Ridge, TN, USA, 2013.
- [6] J. M. Risner, D. Wiarda, M. E. Dunn, T. M. Miller, D. E. Peplow, and B. W. Patton, “Production and Testing of the VITAMIN-B7 Fine-Group and BUGLE-B7 Broad-Group Coupled Neutron/Gamma Cross-Section Libraries Derived from ENDF/B-VII.0 Nuclear Data,” Tech. Rep. NUREG/CR-7045 (ORNL/TM-2011/12), Oak Ridge National Laboratory, Oak Ridge, TN, USA, 2011.

## Appendix A Abaqus Python Macros

The macros (i.e., Python scripts) listed in the subsections of this appendix were recorded within Abaqus. The driver macro `00_Run_All.py` was then created to run them, in the proper order, to fulfill the steps within Section 2. Nevertheless, all of these macros (i.e., `00_ ... 13_`) can be concatenated together into the file `abaqusMacros.py` and used within Abaqus individually to reproduce the steps found within Section 2 or executed at once using `00_Run_All.py`. Note that these macros depend on naming conventions consistent with those described in this document and will not work correctly for models and/or Parts with different names (or if different dimensions are specified for some Parts manually).

Listing 1: "Main Abaqus Driver Macro"

```
1 # -*- coding: mbcs -*-
2 # Do not delete the following import lines
3 from abaqus import *
4 from abaqusConstants import *
5 import __main__
6
7 def a00_Run_All():
8     a01_Rename_Model()
9
10    a02_Create_Paraffin_Block()
11    a03_Create_Cutter()
12    a04_Cut_Paraffin()
13    a05_Create_Graphite()
14    a06_Create_Air()
15
16    a07_Assign_Elsets()
17
18    a08_Create_Materials()
19
20    a09_Partition_Paraffin()
21
22    a10_Mesh_Paraffin()
23    a11_Mesh_Graphite_Air()
24
25    a12_Create_Assembly()
26
27    a13_Create_Job()
```



## A.1 01\_Rename\_Model.py — Rename Model For Convenience

Listing 2: "01\_Rename\_Model.py"

```
1 def a01_Rename_Model():
2     import section
3     import regionToolset
4     import displayGroupMdbToolset as dgm
5     import part
6     import material
7     import assembly
8     import step
9     import interaction
10    import load
11    import mesh
12    import optimization
13    import job
14    import sketch
15    import visualization
16    import xyPlot
17    import displayGroupOdbToolset as dgo
18    import connectorBehavior
19    mdb.models.changeKey(fromName='Model-1', toName='Ueki_20cm')
```

## A.2 02\_Create\_Paraffin\_Block.py — Create Cube to Form Basis of Paraffin Block

Listing 3: "02\_Create\_Paraffin\_Block.py"

```
1 def a02_Create_Paraffin_Block():
2     import section
3     import regionToolset
4     import displayGroupMdbToolset as dgm
5     import part
6     import material
7     import assembly
8     import step
9     import interaction
10    import load
11    import mesh
12    import optimization
13    import job
14    import sketch
15    import visualization
16    import xyPlot
17    import displayGroupOdbToolset as dgo
18    import connectorBehavior
19    s1 = mdb.models['Ueki_20cm'].ConstrainedSketch(name='__profile__',
20        sheetSize=50.0)
21    g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
22    s1.setPrimaryObject(option=STANDALONE)
23    s1.rectangle(point1=(-25.0, 25.0), point2=(25.0, -25.0))
24    p = mdb.models['Ueki_20cm'].Part(name='Paraffin_Block', dimensionality=THREE_D,
25        type=DEFORMABLE_BODY)
26    p = mdb.models['Ueki_20cm'].parts['Paraffin_Block']
27    p.BaseSolidExtrude(sketch=s1, depth=25.0)
28    s1.unsetPrimaryObject()
29    p = mdb.models['Ueki_20cm'].parts['Paraffin_Block']
30    del mdb.models['Ueki_20cm'].sketches['__profile__']
31    p = mdb.models['Ueki_20cm'].parts['Paraffin_Block']
32    f, e = p.faces, p.edges
33    t = p.MakeSketchTransform(sketchPlane=f[5], sketchUpEdge=e[2],
34        sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(0.0, 0.0, 0.0))
35    s = mdb.models['Ueki_20cm'].ConstrainedSketch(name='__profile__',
36        sheetSize=141.42, gridSpacing=3.53, transform=t)
37    g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
38    s.setPrimaryObject(option=SUPERIMPOSE)
39    p = mdb.models['Ueki_20cm'].parts['Paraffin_Block']
40    p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
41    s.rectangle(point1=(-25.0, -25.0), point2=(25.0, 25.0))
42    p = mdb.models['Ueki_20cm'].parts['Paraffin_Block']
43    f1, e1 = p.faces, p.edges
44    p.SolidExtrude(sketchPlane=f1[5], sketchUpEdge=e1[2], sketchPlaneSide=SIDE1,
45        sketchOrientation=RIGHT, sketch=s, depth=25.0,
46        flipExtrudeDirection=OFF)
47    s.unsetPrimaryObject()
48    del mdb.models['Ueki_20cm'].sketches['__profile__']
```

## A.3 03\_Create\_Cutter.py — Create Conic Volume to Subtract from Paraffin Block

Listing 4: "03\_Create\_Cutter.py"

```
1 def a03_Create_Cutter():
2     import section
3     import regionToolset
4     import displayGroupMdbToolset as dgm
5     import part
6     import material
7     import assembly
8     import step
9     import interaction
10    import load
11    import mesh
12    import optimization
13    import job
14    import sketch
15    import visualization
16    import xyPlot
17    import displayGroupOdbToolset as dgo
18    import connectorBehavior
19    s1 = mdb.models['Ueki_20cm'].ConstrainedSketch(name='__profile__',
20        sheetSize=25.0)
21    g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
22    s1.setPrimaryObject(option=STANDALONE)
23    s1.ConstructionLine(point1=(0.0, -12.5), point2=(0.0, 12.5))
24    s1.FixedConstraint(entity=g[2])
25    s1.Spot(point=(0.0, 0.0))
26    s1.Spot(point=(25.0, 0.0))
27    s1.Spot(point=(25.0, 10.3553390593))
28    s1.Line(point1=(0.0, 0.0), point2=(25.0, 0.0))
29    s1.HorizontalConstraint(entity=g[3], addUndoState=False)
30    s1.Line(point1=(25.0, 0.0), point2=(25.0, 10.3553390593))
31    s1.VerticalConstraint(entity=g[4], addUndoState=False)
32    s1.PerpendicularConstraint(entity1=g[3], entity2=g[4], addUndoState=False)
33    s1.Line(point1=(25.0, 10.3553390593), point2=(0.0, 0.0))
34    s1.Line(point1=(0.0, 0.0), point2=(-5.0, 0.0))
35    s1.HorizontalConstraint(entity=g[6], addUndoState=False)
36    s1.setAsConstruction(objectList=(g[6], ))
37    s1.sketchOptions.setValues(constructionGeometry=ON)
38    s1.assignCenterline(line=g[6])
39    p = mdb.models['Ueki_20cm'].Part(name='Paraffin_Cutter',
40        dimensionality=THREE_D, type=DEFORMABLE_BODY)
41    p = mdb.models['Ueki_20cm'].parts['Paraffin_Cutter']
42    p.BaseSolidRevolve(sketch=s1, angle=360.0, flipRevolveDirection=OFF)
43    s1.unsetPrimaryObject()
44    p = mdb.models['Ueki_20cm'].parts['Paraffin_Cutter']
```

## A.4 04\_Cut\_Paraffin.py — Perform ‘Cut’ Operation on Paraffin Block with Cubic Volume

Listing 5: "04\_Cut\_Paraffin.py"

```
1 def a04_Cut_Paraffin():
2     import section
3     import regionToolset
4     import displayGroupMdbToolset as dgm
5     import part
6     import material
7     import assembly
8     import step
9     import interaction
10    import load
11    import mesh
12    import optimization
13    import job
14    import sketch
15    import visualization
16    import xyPlot
17    import displayGroupOdbToolset as dgo
18    import connectorBehavior
19    a = mdb.models['Ueki_20cm'].rootAssembly
20    a.DatumCsysByDefault(CARTESIAN)
21    p = mdb.models['Ueki_20cm'].parts['Paraffin_Block']
22    a.Instance(name='Paraffin_Block-1', part=p, dependent=ON)
23    p = mdb.models['Ueki_20cm'].parts['Paraffin_Cutter']
24    a.Instance(name='Paraffin_Cutter-1', part=p, dependent=ON)
25    a = mdb.models['Ueki_20cm'].rootAssembly
26    a.InstanceFromBooleanCut(name='Paraffin',
27        instanceToBeCut=mdb.models['Ueki_20cm'].rootAssembly.instances['Paraffin_Block-1'],
28        cuttingInstances=(a.instances['Paraffin_Cutter-1'], ),
29        originalInstances=DELETE)
30    p1 = mdb.models['Ueki_20cm'].parts['Paraffin']
```

## A.5 05\_Create\_Graphite.py — Create 20 cm Graphite Shield

Listing 6: "05\_Create\_Graphite.py"

```
1 def a05_Create_Graphite():
2     import section
3     import regionToolset
4     import displayGroupMdbToolset as dgm
5     import part
6     import material
7     import assembly
8     import step
9     import interaction
10    import load
11    import mesh
12    import optimization
13    import job
14    import sketch
15    import visualization
16    import xyPlot
17    import displayGroupOdbToolset as dgo
18    import connectorBehavior
19    s1 = mdb.models['Ueki_20cm'].ConstrainedSketch(name='__profile__',
20        sheetSize=80.0)
21    g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
22    s1.setPrimaryObject(option=STANDALONE)
23    s1.rectangle(point1=(70.0, -40.0), point2=(90.0, 40.0))
24    p = mdb.models['Ueki_20cm'].Part(name='Graphite', dimensionality=THREE_D,
25        type=DEFORMABLE_BODY)
26    p = mdb.models['Ueki_20cm'].parts['Graphite']
27    p.BaseSolidExtrude(sketch=s1, depth=40.0)
28    s1.unsetPrimaryObject()
29    p = mdb.models['Ueki_20cm'].parts['Graphite']
30    del mdb.models['Ueki_20cm'].sketches['__profile__']
31    p = mdb.models['Ueki_20cm'].parts['Graphite']
32    f, e = p.faces, p.edges
33    t = p.MakeSketchTransform(sketchPlane=f[5], sketchUpEdge=e[6],
34        sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(80.0, 0.0,
35        0.0))
36    s = mdb.models['Ueki_20cm'].ConstrainedSketch(name='__profile__',
37        sheetSize=256.12, gridSpacing=6.4, transform=t)
38    g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
39    s.setPrimaryObject(option=SUPERIMPOSE)
40    p = mdb.models['Ueki_20cm'].parts['Graphite']
41    p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
42    s.rectangle(point1=(-40.0, -10.0), point2=(40.0, 10.0))
43    p = mdb.models['Ueki_20cm'].parts['Graphite']
44    f1, e1 = p.faces, p.edges
45    p.SolidExtrude(sketchPlane=f1[5], sketchUpEdge=e1[6], sketchPlaneSide=SIDE1,
46        sketchOrientation=RIGHT, sketch=s, depth=40.0,
47        flipExtrudeDirection=OFF)
48    s.unsetPrimaryObject()
49    del mdb.models['Ueki_20cm'].sketches['__profile__']
50    p1 = mdb.models['Ueki_20cm'].parts['Graphite']
```

## A.6 06\_Create\_Air.py — Create $5 \times 5 \times 5$ cm Air Detector Region

Listing 7: "06\_Create\_Air.py"

```
1 def a06_Create_Air():
2     import section
3     import regionToolset
4     import displayGroupMdbToolset as dgm
5     import part
6     import material
7     import assembly
8     import step
9     import interaction
10    import load
11    import mesh
12    import optimization
13    import job
14    import sketch
15    import visualization
16    import xyPlot
17    import displayGroupOdbToolset as dgo
18    import connectorBehavior
19    s1 = mdb.models['Ueki_20cm'].ConstrainedSketch(name='__profile__',
20        sheetSize=5.0)
21    g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
22    s1.setPrimaryObject(option=STANDALONE)
23    s1.rectangle(point1=(112.5, -2.5), point2=(107.5, 2.5))
24    p = mdb.models['Ueki_20cm'].Part(name='Air', dimensionality=THREE_D,
25        type=DEFORMABLE_BODY)
26    p = mdb.models['Ueki_20cm'].parts['Air']
27    p.BaseSolidExtrude(sketch=s1, depth=2.5)
28    s1.unsetPrimaryObject()
29    p = mdb.models['Ueki_20cm'].parts['Air']
30    del mdb.models['Ueki_20cm'].sketches['__profile__']
31    p = mdb.models['Ueki_20cm'].parts['Air']
32    f, e = p.faces, p.edges
33    t = p.MakeSketchTransform(sketchPlane=f[5], sketchUpEdge=e[6],
34        sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(110.0, 0.0,
35        0.0))
36    s = mdb.models['Ueki_20cm'].ConstrainedSketch(name='__profile__',
37        sheetSize=235.21, gridSpacing=5.88, transform=t)
38    g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
39    s.setPrimaryObject(option=SUPERIMPOSE)
40    p = mdb.models['Ueki_20cm'].parts['Air']
41    p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
42    s.rectangle(point1=(-2.5, -2.5), point2=(2.5, 2.5))
43    p = mdb.models['Ueki_20cm'].parts['Air']
44    f1, e1 = p.faces, p.edges
45    p.SolidExtrude(sketchPlane=f1[5], sketchUpEdge=e1[6], sketchPlaneSide=SIDE1,
46        sketchOrientation=RIGHT, sketch=s, depth=2.5, flipExtrudeDirection=OFF)
47    s.unsetPrimaryObject()
48    del mdb.models['Ueki_20cm'].sketches['__profile__']
49    p1 = mdb.models['Ueki_20cm'].parts['Air']
```

## A.7 07\_Assign\_Elsets.py — Assign Element Set Identifiers to Parts

Listing 8: "07\_Assign\_Elsets.py"

```
1 def a07_Assign_Elsets():
2     import section
3     import regionToolset
4     import displayGroupMdbToolset as dgm
5     import part
6     import material
7     import assembly
8     import step
9     import interaction
10    import load
11    import mesh
12    import optimization
13    import job
14    import sketch
15    import visualization
16    import xyPlot
17    import displayGroupOdbToolset as dgo
18    import connectorBehavior
19
20    p = mdb.models['Ueki_20cm'].parts['Paraffin']
21    p.Set(cells=p.cells, name='Set_material_tally_1')
22
23    p = mdb.models['Ueki_20cm'].parts['Graphite']
24    p.Set(cells=p.cells, name='Set_material_tally_2')
25
26    p = mdb.models['Ueki_20cm'].parts['Air']
27    p.Set(cells=p.cells, name='Set_material_tally_3')
```

## A.8 08\_Create\_Materials.py — Create Material Names & Assign Densities

Listing 9: "08\_Create\_Materials.py"

```
1 def a08_Create_Materials():
2     import section
3     import regionToolset
4     import displayGroupMdbToolset as dgm
5     import part
6     import material
7     import assembly
8     import step
9     import interaction
10    import load
11    import mesh
12    import optimization
13    import job
14    import sketch
15    import visualization
16    import xyPlot
17    import displayGroupOdbToolset as dgo
18    import connectorBehavior
19    mdb.models['Ueki_20cm'].Material(name='Material_Paraffin_1')
20    mdb.models['Ueki_20cm'].materials['Material_Paraffin_1'].Density(table=((-0.93,
21    ), ))
22    mdb.models['Ueki_20cm'].Material(name='Material_Graphite_2')
23    mdb.models['Ueki_20cm'].materials['Material_Graphite_2'].Density(table=((-1.7,
24    ), ))
25    mdb.models['Ueki_20cm'].Material(name='Material_Air_3')
26    mdb.models['Ueki_20cm'].materials['Material_Air_3'].Density(table=((-0.001205,
27    ), ))
```



## A.9 09\_Partition\_Paraffin.py — Partition Paraffin to Permit Hexahedral Meshing

Listing 10: "09\_Partition\_Paraffin.py"

```
1 def a09_Partition_Paraffin():
2     import section
3     import regionToolset
4     import displayGroupMdbToolset as dgm
5     import part
6     import material
7     import assembly
8     import step
9     import interaction
10    import load
11    import mesh
12    import optimization
13    import job
14    import sketch
15    import visualization
16    import xyPlot
17    import displayGroupOdbToolset as dgo
18    import connectorBehavior
19    p = mdb.models['Ueki_20cm'].parts['Paraffin']
20    p.DatumPlaneByPrincipalPlane(principalPlane=XYPLANE, offset=0.0)
21    p = mdb.models['Ueki_20cm'].parts['Paraffin']
22    p.DatumPlaneByPrincipalPlane(principalPlane=XZPLANE, offset=0.0)
23
24    p = mdb.models['Ueki_20cm'].parts['Paraffin']
25    d = p.datums
26    p.PartitionCellByDatumPlane(datumPlane=d[3], cells=p.cells)
27    p = mdb.models['Ueki_20cm'].parts['Paraffin']
28    d1 = p.datums
29    p.PartitionCellByDatumPlane(datumPlane=d1[4], cells=p.cells)
```

## A.10 10\_Mesh\_Paraffin.py — Seed and Mesh Paraffin with Hexahedral Elements

Listing 11: "10\_Mesh\_Paraffin.py"

```
1 def a10_Mesh_Paraffin():
2     import section
3     import regionToolset
4     import displayGroupMdbToolset as dgm
5     import part
6     import material
7     import assembly
8     import step
9     import interaction
10    import load
11    import mesh
12    import optimization
13    import job
14    import sketch
15    import visualization
16    import xyPlot
17    import displayGroupOdbToolset as dgo
18    import connectorBehavior
19    p = mdb.models['Ueki_20cm'].parts['Paraffin']
20    p.seedPart(size=5.0, deviationFactor=0.1, minSizeFactor=0.1)
21    p = mdb.models['Ueki_20cm'].parts['Paraffin']
22    p.seedPart(size=8.0, deviationFactor=0.1, minSizeFactor=0.1)
23    p = mdb.models['Ueki_20cm'].parts['Paraffin']
24    p.seedPart(size=5.0, deviationFactor=0.1, minSizeFactor=0.1)
25    p = mdb.models['Ueki_20cm'].parts['Paraffin']
26    p.generateMesh()
```

## A.11 11\_Mesh\_Graphite\_Air.py — Seed and Mesh Graphite & Air with Hexahedral Elements

Listing 12: "11\_Mesh\_Graphite\_Air.py"

```
1 def a11_Mesh_Graphite_Air():
2     import section
3     import regionToolset
4     import displayGroupMdbToolset as dgm
5     import part
6     import material
7     import assembly
8     import step
9     import interaction
10    import load
11    import mesh
12    import optimization
13    import job
14    import sketch
15    import visualization
16    import xyPlot
17    import displayGroupOdbToolset as dgo
18    import connectorBehavior
19
20    p1 = mdb.models['Ueki_20cm'].parts['Graphite']
21    p = mdb.models['Ueki_20cm'].parts['Graphite']
22    p.seedPart(size=10.0, deviationFactor=0.1, minSizeFactor=0.1)
23    p = mdb.models['Ueki_20cm'].parts['Graphite']
24    p.generateMesh()
25
26    p1 = mdb.models['Ueki_20cm'].parts['Air']
27    p = mdb.models['Ueki_20cm'].parts['Air']
28    p.seedPart(size=2.5, deviationFactor=0.1, minSizeFactor=0.1)
29    p = mdb.models['Ueki_20cm'].parts['Air']
30    p.generateMesh()
```

## A.12 12\_Create\_Assembly.py — Create Assembly from Individual Parts

Listing 13: "12\_Create\_Assembly.py"

```
1 def a12_Create_Assembly():
2     import section
3     import regionToolset
4     import displayGroupMdbToolset as dgm
5     import part
6     import material
7     import assembly
8     import step
9     import interaction
10    import load
11    import mesh
12    import optimization
13    import job
14    import sketch
15    import visualization
16    import xyPlot
17    import displayGroupOdbToolset as dgo
18    import connectorBehavior
19    a1 = mdb.models['Ueki_20cm'].rootAssembly
20    p = mdb.models['Ueki_20cm'].parts['Air']
21    a1.Instance(name='Air-1', part=p, dependent=0N)
22    p = mdb.models['Ueki_20cm'].parts['Graphite']
23    a1.Instance(name='Graphite-1', part=p, dependent=0N)
24    p = mdb.models['Ueki_20cm'].parts['Paraffin']
25    a1.Instance(name='Paraffin-1', part=p, dependent=0N)
```

## A.13 13\_Create\_Job.py — Create Abaqus Mesh Input File from Assembly

Listing 14: "13\_Create\_Job.py"

```
1 def a13_Create_Job():
2     import section
3     import regionToolset
4     import displayGroupMdbToolset as dgm
5     import part
6     import material
7     import assembly
8     import step
9     import interaction
10    import load
11    import mesh
12    import optimization
13    import job
14    import sketch
15    import visualization
16    import xyPlot
17    import displayGroupOdbToolset as dgo
18    import connectorBehavior
19    mdb.Job(name='um_ueki_20cm', model='Ueki_20cm',
20            description='Ueki Benchmark, 20 cm Graphite', type=ANALYSIS,
21            atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=90,
22            memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
23            explicitPrecision=SINGLE, nodalOutputPrecision=SINGLE, echoPrint=OFF,
24            modelPrint=OFF, contactPrint=OFF, historyPrint=OFF, userSubroutine='',
25            scratch='', parallelizationMethodExplicit=DOMAIN, numDomains=1,
26            activateLoadBalancing=False, multiprocessingMode=DEFAULT, numCpus=1)
27    mdb.jobs['um_ueki_20cm'].writeInput(consistencyChecking=OFF)
```



