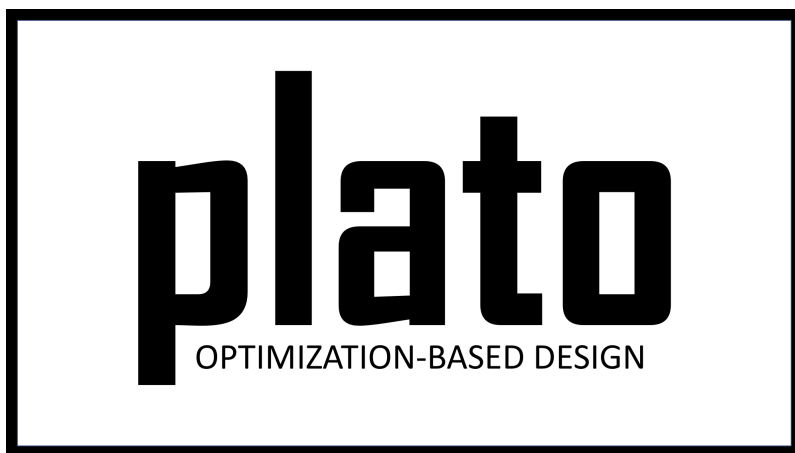


**Plato 2.12 User's Manual**  
SAND2023-10962O  
Unclassified Unlimited Release

Plato Development Team  
Sandia National Labs \*

October 16, 2023



---

\*Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

# Contents

<b>1</b>	<b>Release Notes</b>	<b>4</b>
1.1	Plato 2.12 . . . . .	4
1.2	Plato 2.11 . . . . .	4
1.3	Plato 2.10 . . . . .	4
1.4	Plato 2.9 . . . . .	4
<b>2</b>	<b>Introduction</b>	<b>6</b>
<b>3</b>	<b>Installation and Running Plato</b>	<b>7</b>
3.1	Running on the Sandia CEE LAN . . . . .	7
3.1.1	Signing Up For Sierra Access . . . . .	7
3.1.2	Obtaining an HPC Account . . . . .	7
3.2	Running Your Own Local Installation of Plato at Sandia . . . . .	7
3.2.1	Windows . . . . .	8
3.2.2	Mac . . . . .	8
3.2.3	Linux . . . . .	8
3.3	Installing and Running Outside of Sandia . . . . .	9
<b>4</b>	<b>Tutorial Problems</b>	<b>10</b>
<b>5</b>	<b>Canned Example Problems</b>	<b>11</b>
<b>6</b>	<b>The Overall Process</b>	<b>14</b>
6.1	Problem Setup . . . . .	14
6.1.1	Design Envelope Definition . . . . .	14
6.1.2	Mesh Generation . . . . .	14
6.1.3	Boundary Conditions and Loading . . . . .	15
6.1.4	Units and Magnitudes . . . . .	15
6.1.5	Output . . . . .	15
6.2	Topology Optimization Setup . . . . .	15
6.3	Job Submission . . . . .	16
6.3.1	Auto Mesh Prune and Refine . . . . .	16
6.4	Job Monitoring . . . . .	18
6.5	Export for Manufacture . . . . .	19
<b>7</b>	<b>Graphical User Interface</b>	<b>20</b>
7.1	Main Layout . . . . .	20
7.2	Menus . . . . .	20
7.3	Views . . . . .	20
7.3.1	Graphics Window View . . . . .	21

7.3.2	Model Navigator View . . . . .	22
7.3.3	Console View . . . . .	23
7.3.4	Settings View . . . . .	24
7.3.5	CUBIT™ Command Panel View . . . . .	25
7.3.6	Input Deck Editor View . . . . .	26
7.3.7	Job Status View . . . . .	27
<b>8</b>	<b>Plato Input Deck</b>	<b>28</b>
<b>9</b>	<b>Data Representations</b>	<b>29</b>
9.1	Design Domain Solid Model . . . . .	29
9.2	Design Domain Mesh . . . . .	29
9.3	Optimized Design Results . . . . .	29
9.4	Design Exported to STL Format . . . . .	29
<b>10</b>	<b>Plato Diagnostic File</b>	<b>31</b>
10.1	Optimality Criteria Algorithm . . . . .	31
	<b>References</b>	<b>32</b>

# 1 Release Notes

## 1.1 Plato 2.12

Plato 2.12 has a new capability for shape optimization based on mass properties in Sierra/SD. The mass properties criterion expresses a weighted sum of squared misfits over volume, mass, center of gravity, and inertia. This release also includes an initial capability for stochastic optimization problems in Plato Analyze, as yet without support in the high-level input deck.

## 1.2 Plato 2.11

Plato 2.11 supports use of the AFLR mesher for multi-block shape optimization workflows, greatly improving mesh quality. The multi-block shape optimization workflow has further been enhanced with bugfixes and by removal of the requirement to draw the interface between blocks manually.

## 1.3 Plato 2.10

Plato 2.10 includes three new capabilities to improve the ability to model and solve optimization problems. The first is the ability to model multiple material blocks or volumes on gradient-based shape optimization. This new workflow enables the solution of more realistic customer problems where the parts of the geometry that are being optimized are composed of more than one material or volume. The second new feature is added support for higher-order Finite Element types and the Trilinos linear solver Tacho provide robust and performance-portable physics solves within Plato's optimization ecosystem. Finally, support for multiple constraints has been added to our ROL interface as we move towards making ROL the default optimizer package in Plato for all problem types.

## 1.4 Plato 2.9

Plato 2.9 includes a new capability for generating designs that match a given temperature profile. This capability is the first to interface Plato with Sierra/TF. This capability is available for shape optimization problems but not yet topology optimization problems. The inputs are a Sierra/TF input deck, an existing exodus mesh that has the desired temperature profile stored as nodal variables at locations of interest, and an Engineering Sketch Pad model with the CAD parameters to optimize. Below are some images of a problem where the goal was simply to recapture the CAD parameters so that the new design matches the “gold” temperature profile. The first image shows the gold geometry and temperature profile. The middle image shows the starting geometry for the optimization run and its associated temperature profile. The last image shows the geometry Plato came up with to match the temperature profile.

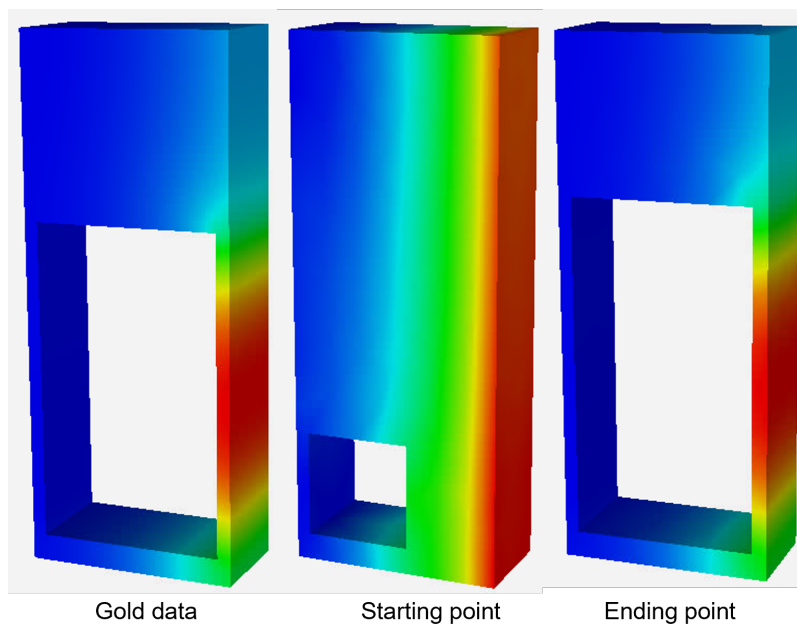


Figure 1: Temperature matching shape optimization problem.

## 2 Introduction

Plato is a design environment that uses Topology Optimization (TO) or Shape/CAD Parameter Optimization (SO) to create designs that are optimized to meet user-specified functional requirements. It works by iteratively running Finite Element Analysis to simulate the required physics and then optimally placing material or modifying CAD parameters to meet the desired objective. The objective could be things like maximizing stiffness, maximizing heat flow, or minimizing stress. The user specifies the “function” and Plato determines the “form.” Plato includes a powerful graphical user interface (GUI) that allows the user to set up the problem and then actively monitor and guide the evolving design.

Plato leverages various existing capabilities such as 1) the Sandia Analysis Workbench (SAW)—a plug-in environment for setting up and running finite element analysis problems, 2) the Sierra/SD physics code for doing steady state linear mechanics, and 3) the [CUBIT<sup>TM</sup>](#) geometry and meshing toolkit. Plato also includes a powerful research physics code, Plato Analyze, for doing GPU-accelerated physics.

## 3 Installation and Running Plato

These instructions are specific to Sandia users. If you are not a Sandia user and need help configuring your installation of Plato, please contact the Plato team at [plato3D-help@sandia.gov](mailto:plato3D-help@sandia.gov).

### 3.1 Running on the Sandia CEE LAN

Plato is installed on the Sandia SRN CEE LAN. Simply type “/projects/plato/plato” to launch Plato on this LAN. The default installation of Plato uses Sierra/SD as its finite element engine, so you will need to have permission to run the Sierra physics codes. If you do not already have access to Sierra physics codes, see the section below on signing up for Sierra access. To launch topology optimization runs on the HPC platforms, you will need an HPC account. If you do not already have an HPC account, see the section below on obtaining an HPC account.

#### 3.1.1 Signing Up For Sierra Access

To sign up for access to run Sierra physics codes, go into Webcars and sign up for “SIERRA Analysts Code Access.” The process of obtaining manager approval and receiving access may take some time.

#### 3.1.2 Obtaining an HPC Account

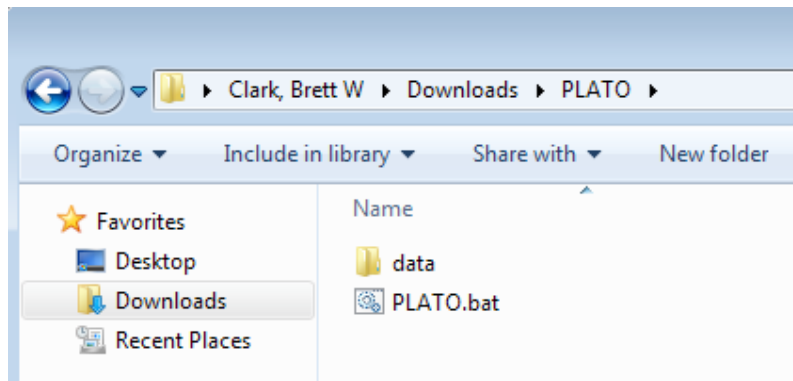
To sign up for an HPC account, first go into Webcars and sign up for “SRN Capacity Clusters” or “SCN Capacity Clusters,” depending on your needs. The process of obtaining manager approval and receiving access may take some time. Next, go to [wc-tool.sandia.gov/wctool](http://wc-tool.sandia.gov/wctool) and submit a request for a WCID. You will use your WCID when launching topology optimization runs on the HPC platforms. If necessary, work with your manager to obtain the correct input for your WCID application.

### 3.2 Running Your Own Local Installation of Plato at Sandia

If you are not running on a LAN where Plato is already installed, you will need to download your own version of Plato and install it on your local machine. To download Plato, go to [www.sandia.gov/plato3d](http://www.sandia.gov/plato3d) and click on the “Downloads” link. Once you have downloaded the install package, follow the instructions below for your platform.

### 3.2.1 Windows

If you downloaded the .zip file, unzip it (it is recommended that you choose an area to unzip the file with no spaces in the path and in a location with a fairly short path name). If you downloaded the self-extracting .exe file, launch it to unzip the package. After you unzip the package, the contents of the top level PLATO directory will look something like that shown below. To launch Plato, double-click on the “PLATO.bat” file (it might just show up as “PLATO”). **Note that there is a “PLATO.exe” executable down in the “data” directory, but launching this file will not work correctly.**

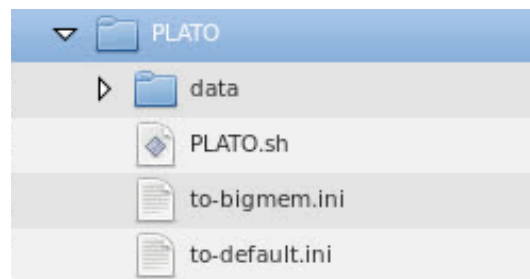


### 3.2.2 Mac

After downloading the Mac install package, unzip the file if necessary. After you unzip the file, you should be able to launch it directly to start PLATO.

### 3.2.3 Linux

After downloading the Linux install package, unzip the file if necessary. After you unzip the file the contents of the top level PLATO directory will look something like that shown below. To launch Plato, double-click on the “PLATO.sh” file. **Note that there is a “PLATO” executable down in the “data” directory, but launching this file will not work correctly.** If prompted as to whether you want to run PLATO or display its contents, choose “Run.”





### 3.3 Installing and Running Outside of Sandia

If you plan to load both Plato and Sierra on a local machine, please see the instructions in [Local Installations](#). If you plan to load Sierra on a remote cluster, please contact the Plato team at [plato3D-help@sandia.gov](mailto:plato3D-help@sandia.gov).

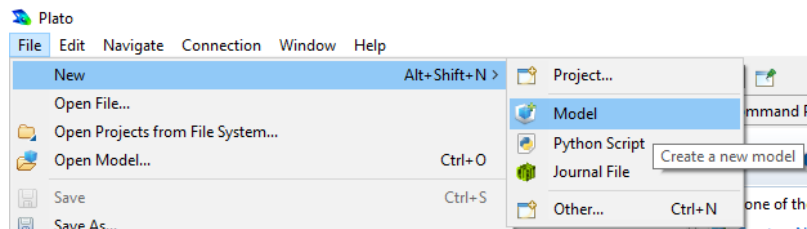
## 4 Tutorial Problems

Various tutorials can be found at the [tutorials](#) section of the Plato website. Start with the one called "Getting Started". You can also try running some canned example problems as described in the next section.

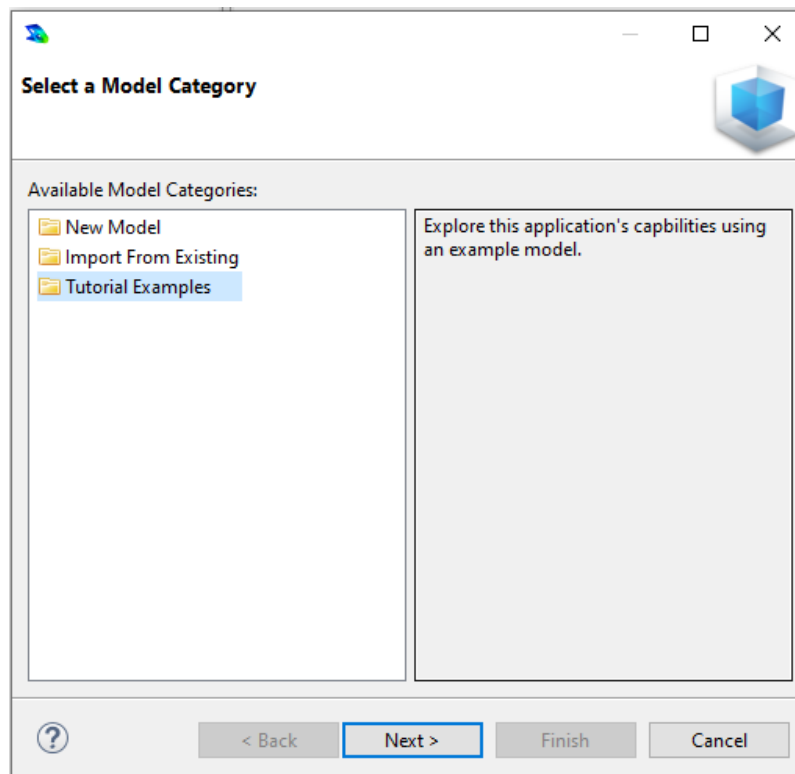
## 5 Canned Example Problems

Plato has some example problems completely setup that you can load and run. This section will describe the process for loading these examples.

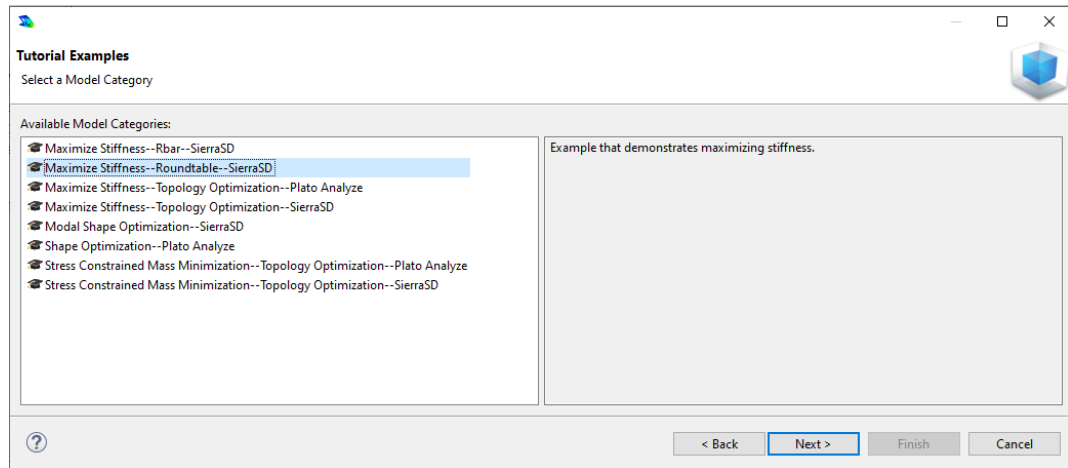
1. Create a new model by choosing “File->New->Model” from the menus.



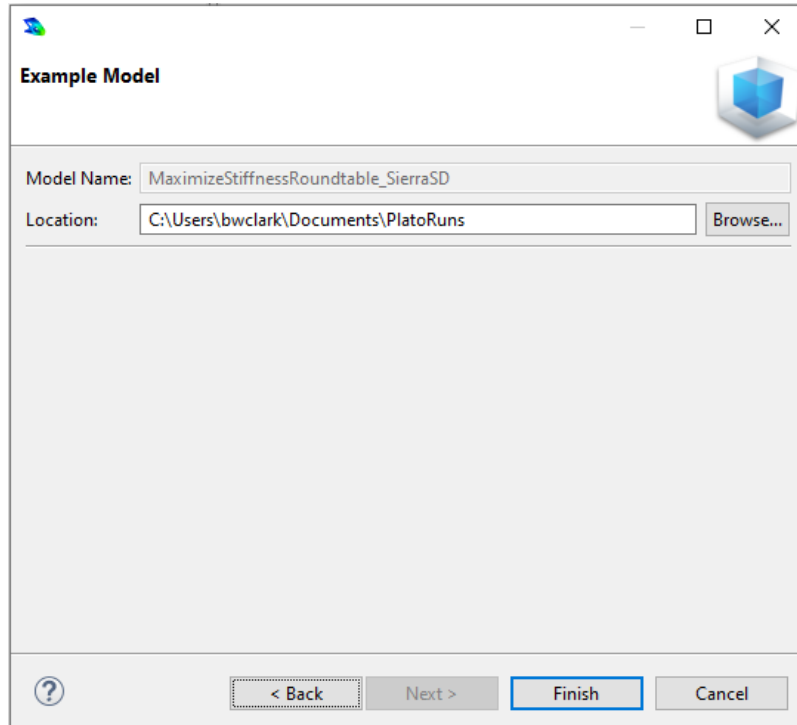
2. Choose “Tutorial Examples” for the category, and click “Next”.



3. Choose one of the tutorial example models, and click “Next”.



Accept the default model name and location, and click “Finish”.



At this point, Plato will create a new model containing the example problem, and you can modify the default parameters or choose your own and run the optimization as usual. An example result for the RoundTable tutorial example is shown below for reference.



RoundTable tutorial example result.

## 6 The Overall Process

Designing with topology optimization is fundamentally different from traditional CAD design. In traditional design, an engineer or designer will typically mock up a design in a CAD system using his or her knowledge and experience related to the design space. The result is a CAD model specifying the “form” of the design with the hope that it will meet the “function” required of the design. The function is then often assessed using experimentation or computational analysis. When designing with topology optimization, this traditional design paradigm is inverted. Instead of defining the form of the design, the engineer or designer specifies the required function and lets the topology optimization come up with the form that will meet that function based on computational analysis. This section describes the main steps in designing with topology optimization using Plato.

### 6.1 Problem Setup

Problem setup consists of defining the design envelope or allowable space in which the optimized design can reside, the functional requirements that need to be met by the design, and the desired objective to optimize.

#### 6.1.1 Design Envelope Definition

The design domain or envelope is the allowable space that the optimized design can occupy. The design domain could be as simple as a cubic bounding box or much more complicated if the component being designed is part of a complex assembly of components. Based on your requirements, you will need to define what this allowable space looks like and generate a solid model to represent it. Specifications of the design domain can be done within Plato using the solid modeling capabilities there or in another CAD package if it is easier. If you will be generating this solid model external to Plato, you should save it in the STEP format so that it can be imported into Plato. If you will be generating the solid model for the design domain using the solid modeling capabilities in Plato, please see the [CUBIT™ user manual \[1\]](#) for complete documentation on these capabilities.

#### 6.1.2 Mesh Generation

The input for the topology optimization consists of a finite element mesh of the design domain defined in [6.1.1](#) and an input deck with parameters defining the functional requirements of the design and other parameters to guide the optimization. Therefore, the next step is to generate a finite element mesh of the design domain solid model. A mesh can either be created outside of Plato and then imported into Plato, or it can be generated in Plato using the CUBIT™ mesh generation capabilities. For detailed documentation on using the CUBIT™ mesh generation capabilities incorporated in Plato, please see the [CUBIT™ user manual \[1\]](#) and [CUBIT™ tutorials \[2\]](#).

### 6.1.3 Boundary Conditions and Loading

Design functional requirements are communicated to Plato via boundary conditions and loads. For example, in the lantern bracket problem in the “Getting Started” tutorial (see [tutorials](#)), we need to communicate to Plato that the bracket had to be bolted to a wall and support the load of a hanging lantern. Appropriate boundary conditions need to be specified: a fixed boundary condition where the design would be attached to the wall, and a vertical load where the lantern would hang.

### 6.1.4 Units and Magnitudes

It is up to the user to maintain consistent units when setting up a problem. For example, material properties and loads should have consistent units. For compliance minimization (stiffness maximization) the position and direction of a load are critical for determining the “stiffest” configuration of material – not the magnitude of the load. However, if there are multiple loads, the relative magnitudes of the loads are important. Even though absolute magnitudes don’t play a role in determining optimally “stiff” structures, getting rough magnitudes of loads correct with regards to units will help with the stability of the optimization algorithm.

### 6.1.5 Output

Plato can generate output containing variables for the design topology, displacements at the nodes, and VonMises stresses on the elements. These quantities will show up in the output exodus result files as variable names “Topology,” “dispx,” “dispy,” “dispz,” and “vonmises.” The Topology field is a nodal field which contains the values of the design variables in the range of  $[0, 1]$  with 0 signifying void material and 1 for solid material. Upon completion, isosurfaces are extracted using the Topology field at a value of 0.5 (currently not modifiable by user) at the number of iterations requested by the user. The Topology field must be present for extraction of the isosurfaces. Vonmises is an element field containing the values of the Von Mises stress for each element. The disp fields output displacements in the  $x$ ,  $y$  and  $z$  directions.

## 6.2 Topology Optimization Setup

The user sets up objectives and constraints that define the functional requirements of the design. For example, the user may choose to define an objective to have the design be as stiff as possible with a volume fraction constraint. The user also has some control over the optimization algorithm through parameters in the input deck. For example, the user can specify the maximum number of iterations and the smoothing filter radius. The various topology optimization parameters are described in detail in [section 8](#).

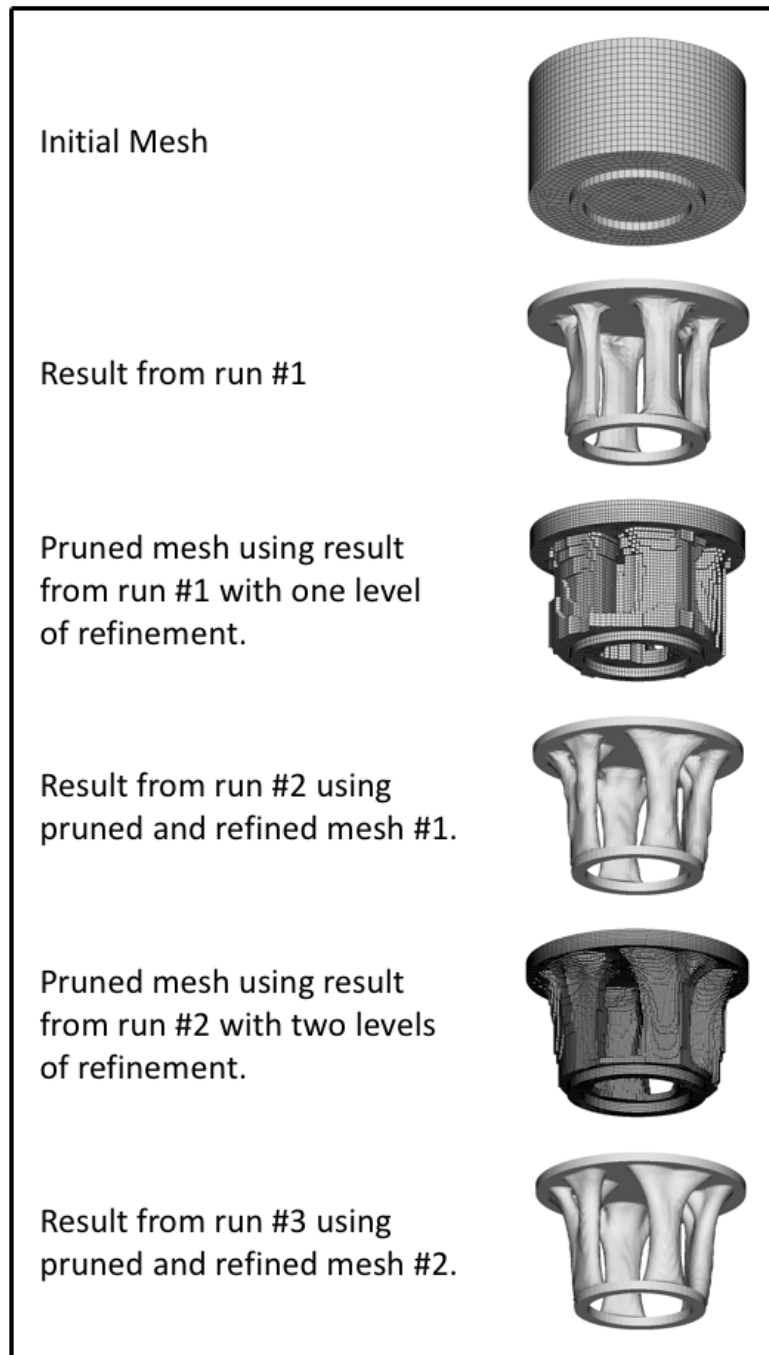
### 6.3 Job Submission

Once all the input for the topology optimization problem is defined, the job is submitted. The job can be submitted either to run on the local machine or on a remote cluster or HPC platform. Plato provides a very nice interface for specifying where and how the job will be run. The user can specify the machine, number of processors, requested time, etc. and then launch the job with a single click. Valuable information such as current machine load is shown graphically in the “Machines” view and helps in determining the best place to run the job. Multiple jobs can be launched and running simultaneously in the Plato environment.

#### 6.3.1 Auto Mesh Prune and Refine

Plato provides a capability called mesh pruning and refining that is used to help reduce computational cost while providing higher resolution of design features. This feature allows the user to specify how many levels of mesh refinement he wants and how much of the “unused” mesh from a previous run he wants to prune away. Thus, the pruning aspect of this feature only makes sense when doing a restart from a previous iteration where a design exists. Example results and intermediate pruned and refined meshes are shown in the diagram below. Note that this prune feature is almost always employed to produce smooth designs ready to directly print on an additive manufacturing (AM) machine.





The mesh pruning and refining happens at the beginning of a restart of a prior optimiza-

tion, before the next topology optimization process is launched. Because pruning relies on a previous result, a job submission using pruning must be launched as a restart job. The user must specify a “restart iteration” in the input deck/topology optimization parameters, indicating which iteration of the prior optimization to use as the restart.

Because the volume of the starting mesh will change when using a pruned mesh, the target “volume fraction” parameter in the input deck would also need to be updated to reflect the fact that the input mesh was pruned. Instead of calculating an updated “volume fraction” value you can simply use the “volume absolute” input deck parameter instead, which is the absolute target volume that you want the resulting optimized design to have. Using this parameter avoids the need to constantly be updating the target volume fraction. The absolute target volume can be calculated by multiplying the volume of your original design domain by “volume fraction.” If you don’t know the original volume, use CUBIT<sup>TM</sup> to calculate this, using the “list volume” command.

The prune and refine parameters are in the “optimization parameters” section of the input deck. The user can request pruning using “prune mesh true.” The “number buffer layers” parameter is only applicable when pruning and specifies how many layers of elements outside of the elements intersected by the design will be retained in the pruned mesh. The “restart iteration” parameter you specify in the input deck will tell Plato which design iteration from the previous run should be used to guide the pruning. All elements that either lie in the design or intersect the boundary of the design will be kept by default, and then the user-specified number of buffer layers outside of these elements will also be retained. “Number refines” specifies how many levels of global refinement should be applied to the mesh (pruned or not) before launching the topology optimization run. Refinement is independent of pruning, and thus you can specify to simply globally refine the input mesh without doing any pruning. Furthermore, refinement is independent of restarting and can be specified regardless of whether or not the run is a restart run.

## 6.4 Job Monitoring

Plato provides some nice tools for monitoring the topology optimization process as it progresses. For jobs being run on queued systems, the “Job Status” view lets the user know when the job gets through the queue and actually starts running. This view can also be used to kill running jobs. Once a job starts running, the user will see snapshots of the evolving design in the graphics window. The user can change how often these results are shown, depending on how large the problem is and how long each iteration takes. Each of these snapshots represents an actual design iteration that can be viewed and manipulated within Plato and then exported for 3D printing if desired. At any time, the user can also plot the optimization objective value vs. iteration to see how well the process is converging to a solution.

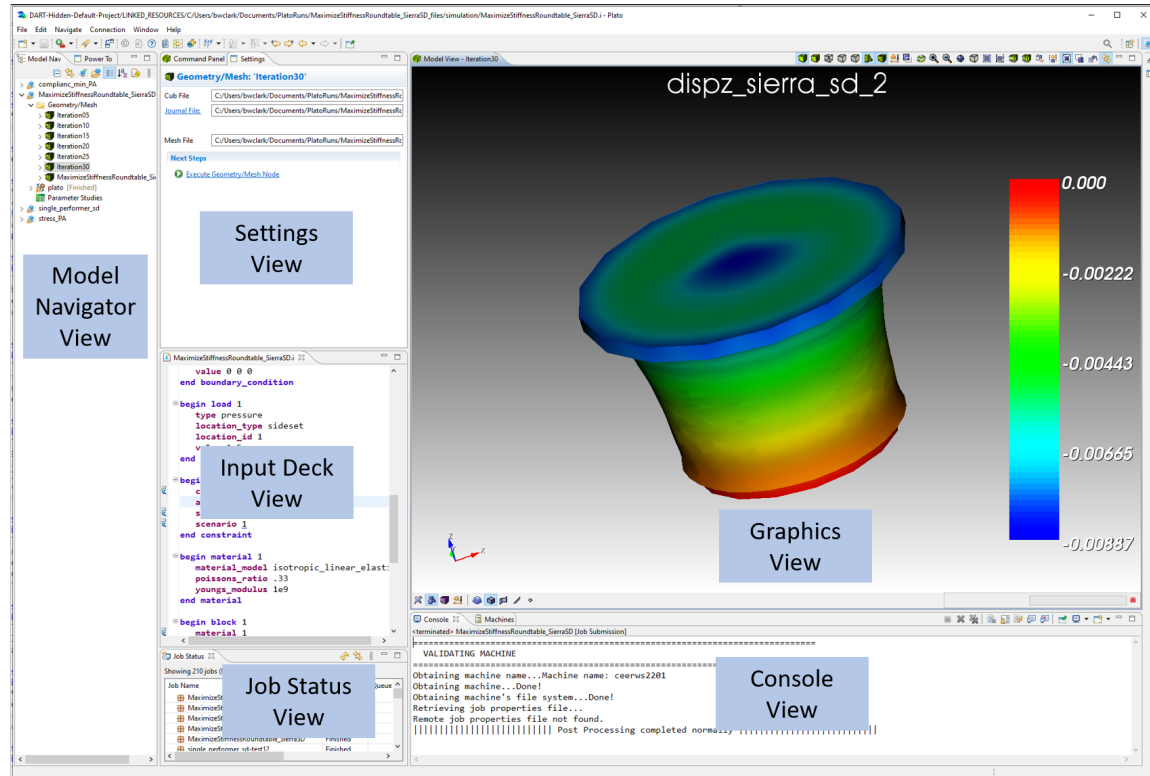
## **6.5 Export for Manufacture**

At any point during or after the optimization run, the user can export any of the design iterations that were generated to an STL file for 3D printing.

## 7 Graphical User Interface

This section will describe the different aspects of Plato's graphical user interface.

### 7.1 Main Layout



Plato has a full-featured graphical user interface (GUI) to provide a powerful, user-friendly environment for doing topology optimization-based design. The image above depicts a typical Plato layout, with many of the GUI features present. In the sections that follow, the various features of the Plato GUI will be discussed in more detail.

### 7.2 Menus

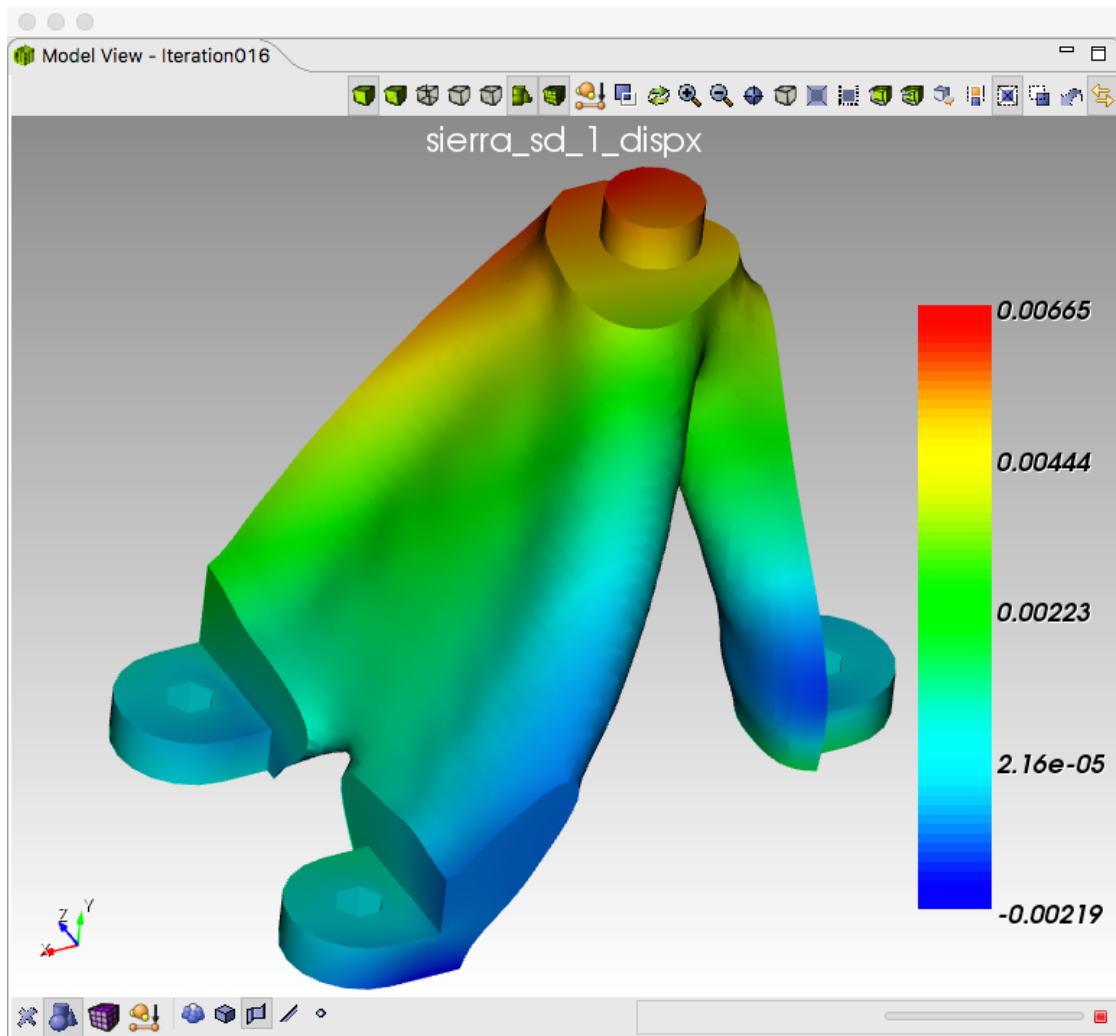
As with most modern GUIs, Plato has a set of top level menus for accessing many of the administrative tasks related to managing projects, models, preferences, etc.

### 7.3 Views

Most of the features in the Plato GUI are in the form of a separate window or view. Each of the views can be dragged outside of the main Plato application or “undocked” as a

separate window. To re-dock a window back into the Plato application, simply click on the tab just under the window title bar and drag that tab back into Plato. To open views that have been closed or did not show up in the default layout when Plato started, you can go to “Window->Show View” in the main menus and choose the view you want to show. In the sections that follow, most of the commonly used views will be shown undocked, and each will be described in more detail. Other views are accessible by going to “Window->Show View.”

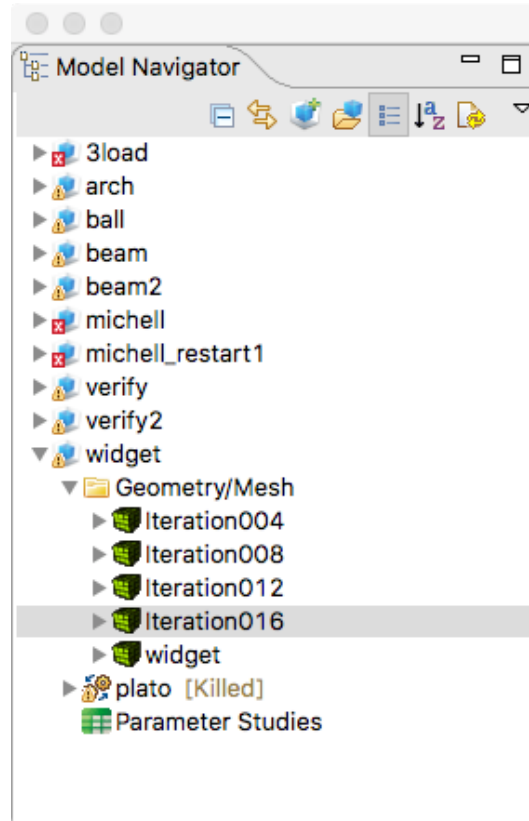
### 7.3.1 Graphics Window View



In the graphics window, you can see the domain envelope, any meshes you generate, design results coming back from the optimization, etc. Like most CAD systems, you can pan,

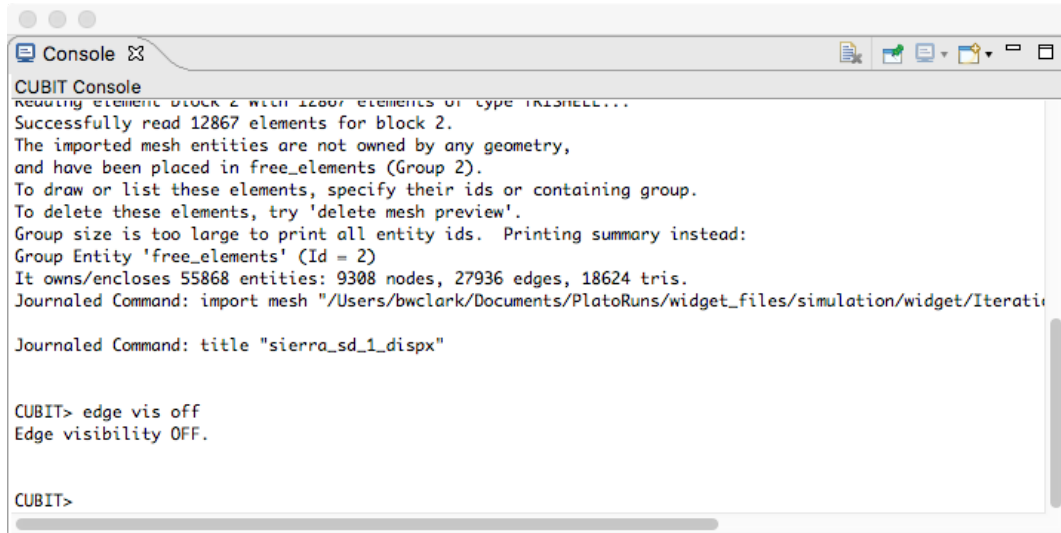
zoom, and rotate the view in the graphics window by dragging the mouse while holding down different mouse buttons. The graphics window is the same one used in the CUBIT<sup>TM</sup> mesh generation software package and has many powerful capabilities for interacting with the various models you will generate. For more details about these capabilities see the [CUBIT<sup>TM</sup> user manual \[1\]](#).

### 7.3.2 Model Navigator View



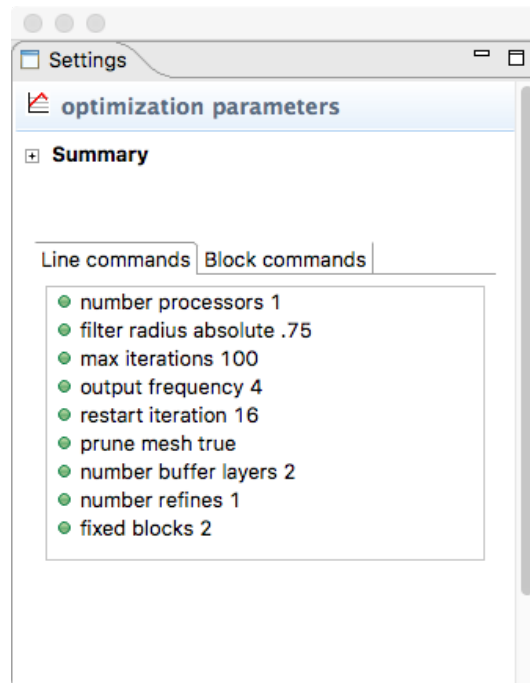
The model navigator view is where all of your models are displayed. Each model is represented as a tree, and you can explore the different parts of the model by expanding its various branches. Each model has a “Geometry/Mesh” node that contains the domain envelope solid model, domain envelope mesh, and optimized design results. Each model also has a node containing all the finite element analysis data and topology optimization data. This node is called “plato.” After a topology optimization job has been submitted, a “Simulation Job” node in the tree will appear, which contains the contents of the local and remote directories where the optimization was run.

### 7.3.3 Console View



The console view allows for text input and output in various modes. One function of the console view is to show text output from running processes. For example, when a job is submitted, the console window displays text output about what is happening at each stage of the job submission. Monitoring this output is a good way to make sure things are progressing as expected or to identify errors or problems during the process. Another purpose of the console view is to provide a place to enter commands when using the CUBIT<sup>TM</sup> geometry and meshing component in Plato. These different functions or modes of the console view are managed by actually having multiple consoles. There is one console for displaying output and a different console for entering CUBIT<sup>TM</sup> commands. You can toggle through the different consoles at any time by clicking on the icon that looks like a computer monitor in the toolbar in the top right portion of the console view.

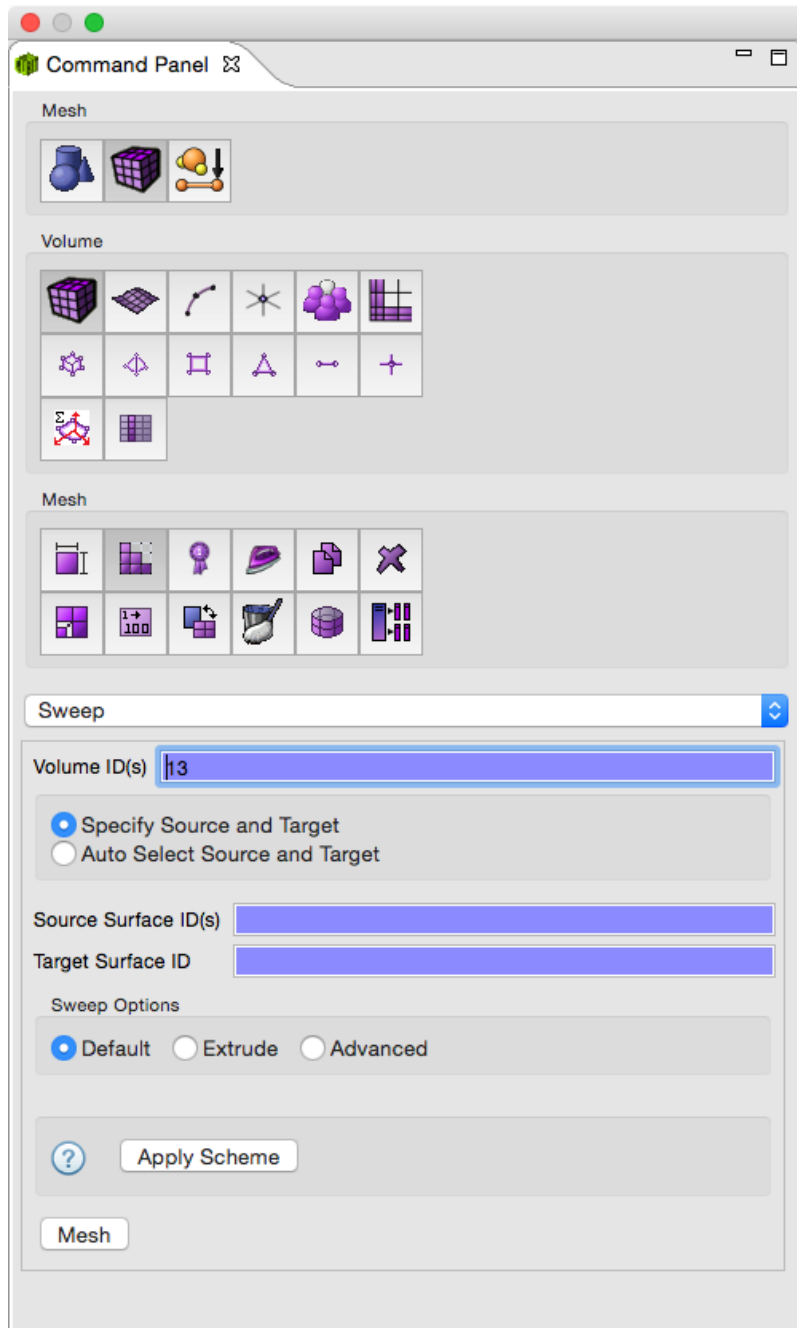
### 7.3.4 Settings View



The settings view is a general-purpose view where many different kinds of data are displayed and edited, depending on what is selected elsewhere in the Plato GUI. One of the most common uses of the settings view is to view and edit the data associated with whatever is selected in the model navigator view. For example, in the image above, topology optimization parameters are being displayed, which is enabled by the user selecting the “Optimization Parameters” node in a model in the model navigator view. Often, parameters and data in the settings view can be edited by simply clicking on it and entering new values. Look to the setting view for various kinds of information. If the settings view ever seems to “disappear,” make sure that it isn’t just behind another tab in the window where it is docked.



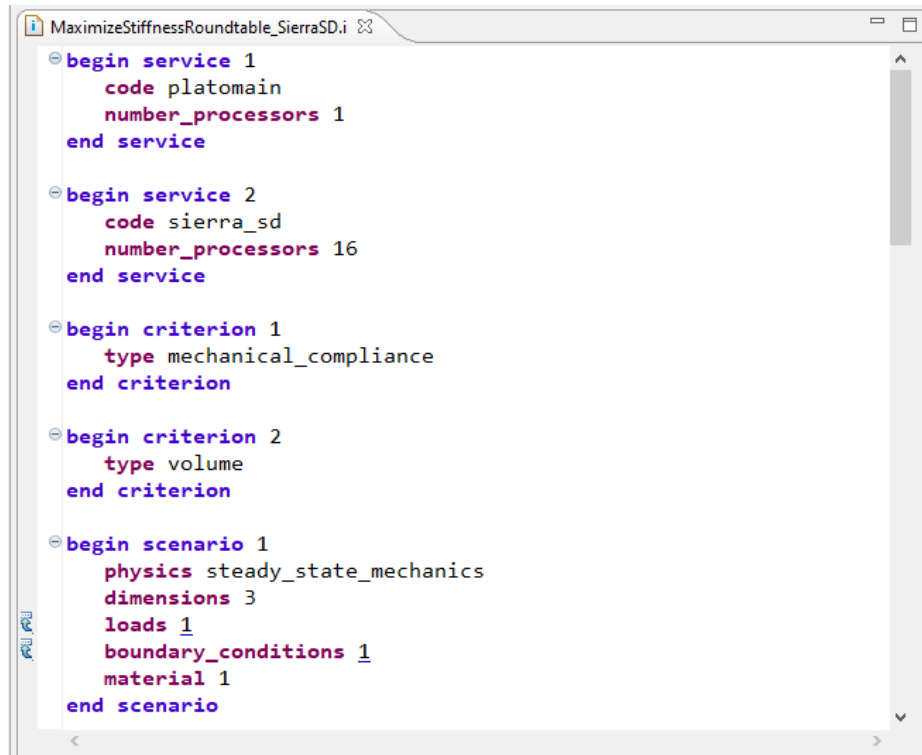
### 7.3.5 CUBIT™ Command Panel View



The CUBIT™ command panel view is where you can find all of the GUI panels related to performing geometry and meshing commands using the CUBIT™ component in Plato.

For help on learning more about the CUBIT™ command panel, see the [CUBIT™ tutorial \[2\]](#) on the CUBIT™ GUI.

### 7.3.6 Input Deck Editor View



```
begin service 1
  code platomain
  number_processors 1
end service

begin service 2
  code sierra_sd
  number_processors 16
end service

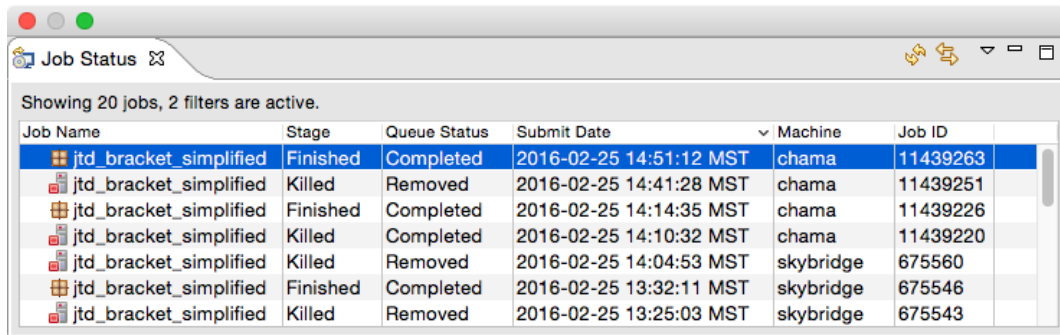
begin criterion 1
  type mechanical_compliance
end criterion

begin criterion 2
  type volume
end criterion

begin scenario 1
  physics steady_state_mechanics
  dimensions 3
  loads 1
  boundary_conditions 1
  material 1
end scenario
```

Plato contains a very nice input deck editor. The input deck is the file that controls the simulation. This input deck editor view is where the user can view and edit the input deck parameters defined in the model navigator tree. This editor is syntax aware so that it can flag errors in the input deck and provide context sensitive suggestions. The input deck editor is completely coupled with the model navigator and settings views so that any edits made to parameters in the input deck editor are immediately reflected in the model navigator and settings and graphics views and vice versa.

### 7.3.7 Job Status View



The screenshot shows a window titled "Job Status" with a search icon. Below the title bar, it says "Showing 20 jobs, 2 filters are active." The main content is a table with the following columns: Job Name, Stage, Queue Status, Submit Date, Machine, and Job ID. The table contains 7 rows of data, with the first row highlighted in blue. Each row has a small icon to the left of the Job Name.

Job Name	Stage	Queue Status	Submit Date	Machine	Job ID
jtd_bracket_simplified	Finished	Completed	2016-02-25 14:51:12 MST	chama	11439263
jtd_bracket_simplified	Killed	Removed	2016-02-25 14:41:28 MST	chama	11439251
jtd_bracket_simplified	Finished	Completed	2016-02-25 14:14:35 MST	chama	11439226
jtd_bracket_simplified	Killed	Completed	2016-02-25 14:10:32 MST	chama	11439220
jtd_bracket_simplified	Killed	Removed	2016-02-25 14:04:53 MST	skybridge	675560
jtd_bracket_simplified	Finished	Completed	2016-02-25 13:32:11 MST	skybridge	675546
jtd_bracket_simplified	Killed	Removed	2016-02-25 13:25:03 MST	skybridge	675543

The job status view displays information about all of the jobs that have previously run or are currently running. This view provides one way for you to monitor the status of your currently running jobs. You can click on jobs to control them.

## 8 Plato Input Deck

The Plato input deck defines an optimization problem that can be solved with a collection of codes. The input deck provides a unifying abstraction from the input formats of the underlying codes. A complete description of the input deck can be found in the [input deck reference manual](#).

## 9 Data Representations

During the topology optimization process, various models are generated, and various data representations are used. This section will describe the various models and data representations that you will see as you progress through the design process in Plato. A graphical depiction of the process and models is shown in [Figure 2](#).

### 9.1 Design Domain Solid Model

Ultimately, the design domain needs to be represented as a finite element mesh for the topology optimization run. If such a mesh already exists, it can be imported into Plato directly. Otherwise, a solid model of the design domain can be imported into Plato and then meshed using the geometry and meshing capabilities in Plato. Solid models can also be created directly in Plato if desired and then meshed. The best format for solid models being imported into Plato is the .sat ACIS format or STEP.

### 9.2 Design Domain Mesh

A finite element mesh of the design domain is required as input to the topology optimization run. The design domain mesh can be generated in Plato from the design domain solid model or in another package and then imported into Plato. The best format if importing the mesh is the Exodus format. It will typically have a .exo, .e, .gen, or .g file extension.

### 9.3 Optimized Design Results

As the optimization process advances snapshots of the evolving design will be loaded and displayed in Plato. These design iterations are in the Exodus mesh file format and are simply a set of water-tight mesh triangles representing the boundary of the optimized design. Note that this is only a triangle mesh and does not contain volume mesh elements on the interior.

### 9.4 Design Exported to STL Format

Any one of the design results loaded into Plato during the optimization process can be exported to an STL file for 3D printing by simply right-clicking on the design in the Model Tree and choosing “Generate STL”.

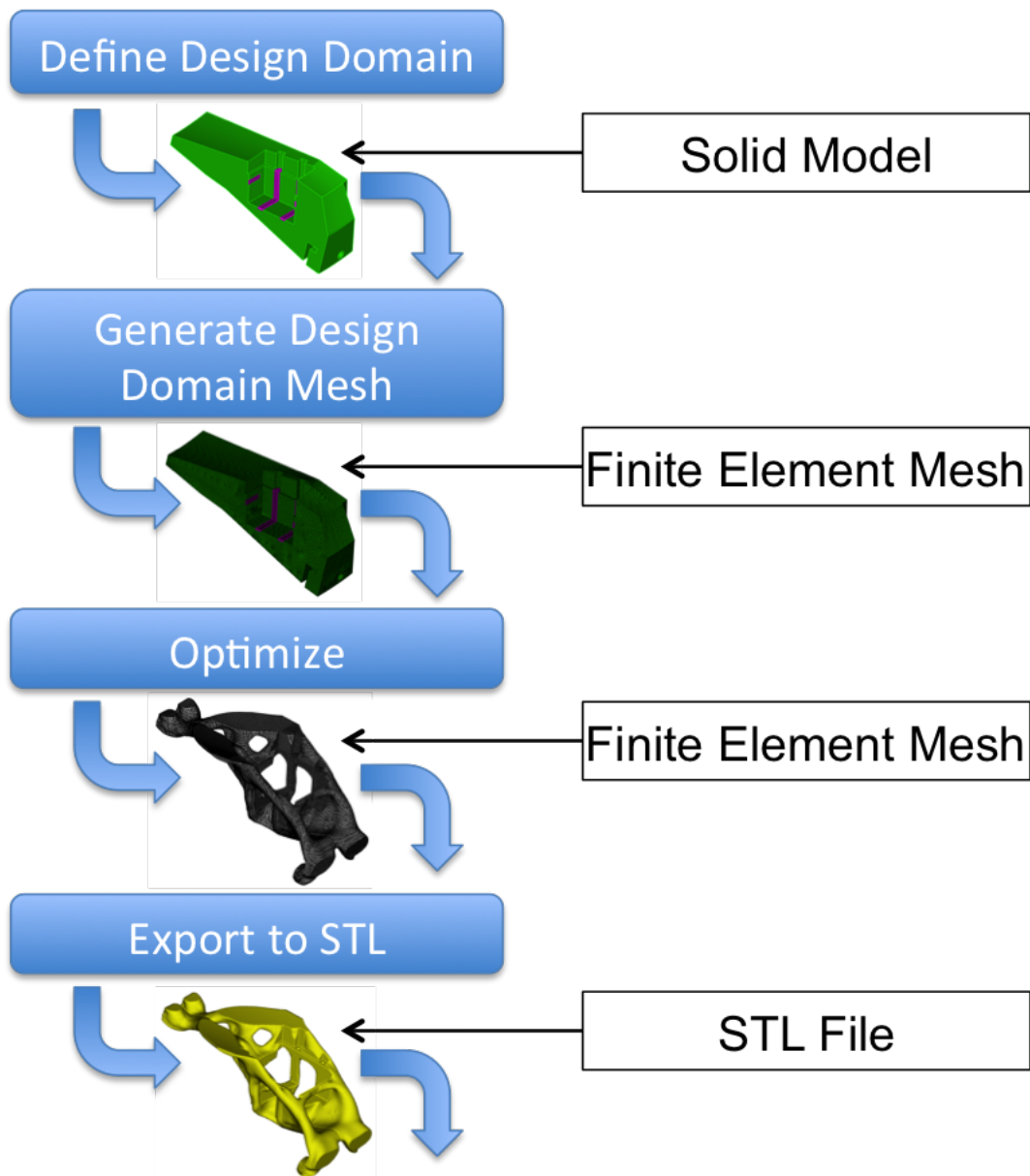


Figure 2: Models encountered during the topology optimization process in Plato.

## 10 Plato Diagnostic File

The Plato diagnostic file is a table of statistics describing the progress of the optimization algorithm at every iteration. The statistics depend on the optimizer selected by the user. The statistics are always written to a text file, which users can track throughout the optimization iterations.

### 10.1 Optimality Criteria Algorithm

This section presents the format used to present the statistics associated with the Optimality Criteria algorithm. These statistics are collected at every iteration and saved in an output file named `plato_optimality_criteria_diagnostics.txt`. The following table lists the headings found in `plato_optimality_criteria_diagnostics.txt`.

Table 1: Headings found in `plato_optimality_criteria_diagnostics.txt`.

Heading	Description
Iter	Outer optimization iteration.
F-count	Number of function evaluations.
F(X)	Objective function value.
Norm(F')	Norm of the objective function gradient.
$H_i(X)$	Inequality constraint value, where $i = 1, \dots$ , the total number of constraints.
abs(dX)	Control stagnation metric. The control stagnation metric measures the infinity norm of the misfit between two subsequent vectors of design variables (i.e., controls).
abs(dF)	Objective function stagnation metric. The objective function stagnation metric measures the misfit between two subsequent objective function evaluations.

## References

- [1] Online CUBIT<sup>TM</sup> User's Documentation, <https://cubit.sandia.gov/public/documentation.html>.
- [2] CUBIT<sup>TM</sup> tutorials, <https://cubit.sandia.gov/public/tutorials.html>.
- [3] "Sierra/SD User's Manual - 5.0", SAND2021-2952, Albuquerque, NM, Sierra/SD Team, Nathan Crane, March 2021, Unclassified Unlimited Release.