Plato User’s Manual

Plato Development Team

October 5, 2017
7 Topology Optimization Parameter Reference

7.1 Sierra/SD .................................................. 26
7.2 Albany .................................................... 27

8 Physics

8.1 Sierra/SD .................................................. 29
  8.1.1 Multiple Load Cases ................................... 29
  8.1.2 Body Loads ........................................... 29
  8.1.3 RBAR Elements ...................................... 30
8.2 Albany .................................................... 30

9 Theory

9.1 Material Model ........................................... 31
  9.1.1 SIMP: Solid Isotropic Material with Penalization 31
  9.1.2 RAMP: Rational Approximation of Material Properties ........................................... 31

10 Data Representations

10.1 Design Domain Solid Model ............................... 32
10.2 Design Domain Mesh ..................................... 32
10.3 Optimized Design Results ................................ 32
10.4 Design Exported to STL Format .......................... 32

References ...................................................... 34
1 Introduction

Plato is a design environment that uses Topology Optimization (TO) to create “Generative Designs”–designs that are optimized to meet user-specified functional requirements. It does this by iteratively running Finite Element Analysis to simulate the required physics and then optimally placing material 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 that allows the user to set up the problem and then actively monitor and guide the evolving design.

Plato leverages existing tools where possible. It is built on the Sandia Analysis Workbench (SAW)–a plug-in environment for setting up and running finite element analysis problems. This environment provides many of the capabilities required for running topology optimization. Plato also leverages the CUBIT geometry and meshing toolkit. The CUBIT functionality is included as a component in the Plato environment. See https://cubit.sandia.gov for more information about CUBIT.
2 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.

2.1 Running on the Sandia CEE LAN

Both the Sandia SRN and SCN CEE LANs have Plato installed on them. Simply type “/projects/plato/plato” to launch Plato on these LANs. 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.

2.1.1 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”. This will probably require manager approval and may take overnight to get into the system.

2.1.2 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. This will probably require manager approval and may take overnight to get into the system. Next, go to wc-tool.sandia.gov/wc-tool 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.

2.2 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 and click on the ”Downloads” link. Once you have downloaded the install package follow the instructions below for your platform.
2.2.1 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. **Note that there is a “PLATO.exe” executable down in the “data” directory but launching this file will not work correctly.**

2.2.2 Mac

After downloading the Mac 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.command” file. **Note that there is a “PLATO.app” executable down in the “data” directory but launching this file will not work correctly.**
2.2.3 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.**

![PLATO directory contents](image)

2.3 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.
3 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.
4 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”.

![Select a Model Category](image-url)
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. Example results for the Crank, Hanger, and RoundTable tutorial examples respectively are shown below for reference.

Crank, Hanger, and RoundTable tutorial example results.
5 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.

5.1 Problem Setup

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

5.1.1 Problem Domain Definition

The design domain or envelope is the allowable space that the optimized design can occupy. This 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. This 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.

5.1.2 Mesh Generation

The input for the topology optimization consists of a finite element mesh of the design domain defined in 5.1.1 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].
5.1.3 Boundary Conditions and Loading

Design functional requirements are communicated to Plato via boundary conditions and loads. For example, in the lantern bracket tutorial problem we need to communicate to Plato that the bracket had to be bolted to a wall and support the load of a hanging lantern. To do this we specify a fixed boundary condition where the design would be attached to the wall and applied a vertical load where the lantern would hang. The boundary condition and load capabilities exposed in Plato come from the underlying finite element analysis package that is being used. Plato is currently using Sierra/SD as its finite element analysis solver. For complete documentation on the available boundary conditions and loads see sections 2.13 and 2.14 in the Sierra/SD User’s Manual [3].

5.1.4 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 it is important to get rough magnitudes of loads correct with regards to units so that the problem is well-posed. This will help the stability of the optimization algorithm.

5.1.5 Output

By default, Plato will generate output containing variables for the design topology, displacements at the nodes, and VonMises stresses on the elements. These will show up in the output exodus result files as variables names ”OptimalTopology”, ”Disp”, and ”VonMises” respectively. The OptimalTopology 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 OptimalTopology field at a value of 0.5 at the number of iterations requested by the user. The OptimalTopology 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 field outputs displacements and rotations in the $x$, $y$, and $z$ directions. Since Plato is built on Sierra/SD, other fields of interest may be output as specified in the Sierra/SD User’s Manual [3] in section 2.8.

5.2 Topology Optimization Parameter Setup

Plato allows the user some control over the topology optimization algorithm by exposing some of the optimization parameters. For example, the user specifies what fraction of the original domain envelope can be used for the final design. The user can also modify
parameters that affect the feature size and smoothness of the final design. The various topology optimization parameters are described in detail in section 7.

5.3 Job Submission

Once all of 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.

5.3.1 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.
The mesh pruning and refining happens at the beginning of the job before the topology
optimization process is launched. Because pruning relies on a previous result a job submission using pruning must be launched as a restart job. To do this you must specify a “restart_index” in the input deck/topology optimization parameters and indicate when submitting the job that you do not want Plato to clear the working directory before starting. This is critical because you will need the results in the working directory from the previous run. The checkbox for telling Plato not to clear the remote directory is in the “Job Submission” panel under the “Machine” section and is labeled “Clear job run directory” – make sure this is not checked.

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_target” input deck parameter instead. “volume_target” 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”.

The controls for specifying the pruning and refining parameters are found in the “Code Parameters” section of the “Job Submission” panel (see image below).

The “Auto prune” checkbox specifies whether pruning will take place. The “Number of 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_index” 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 of 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.
5.4 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 represent 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 quantities of interest such as the current design volume or current optimization objective value vs. iteration to see how quickly the process is converging to a solution.

5.5 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.
6 Graphical User Interface

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

6.1 Main Layout

Plato leverages 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.

6.2 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.

6.3 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”.

6.3.1 Graphics Window View

The graphics window is where you do all of the graphical visualization. This is where you 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 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 user manual [1].
6.3.2 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 of the finite element analysis data and topology optimization data. For Sierra/SD models this node is called “Sierra Structural Dynamics”. 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.
6.3.3 Console View

The console view has multiple functions. 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 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 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.
6.3.4 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. This is a result of the user selecting the “Topology-Optimization” node in a model in the model navigator view. Often times parameters and data in the settings view can be edited by simply clicking on it and entering new values. You will get used to looking 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.
6.3.5 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.

6.3.6 Input Deck Editor View

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.
6.3.7 Machines View

The machines view displays information about the available HPC machines that you can submit jobs to. It displays the current load on the machine and provides functionality for estimating how long it will take your job to get through the queue on a given machine. This information is very valuable in determining where to run jobs.

6.3.8 Job Status View

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 or your currently running jobs.
7 Topology Optimization Parameter Reference

The input deck for a topology optimization run is a combination of parameters needed for the finite element analysis calculations and parameters needed for the topology optimization algorithms. This section is a reference guide for the topology optimization parameters that are found in the input deck. They will be grouped by the physics code being used. For a description of the physics codes see Section 8 on page 29.

7.1 Sierra/SD

The topology optimization parameters for Sierra/SD are in the “Topology-Optimization” block in the input deck. We will only describe the basic topology optimization parameters that a user would typically modify in this manual. If you have questions about other topology optimization parameters in the “Topology-Optimization” block please contact the Plato development team at plato3D-help@sandia.gov.

**volume_fraction**: Specifies the fraction of the starting design domain volume that the user wants in the optimized design. Fixed blocks are included in the calculation of the optimized design volume at each iteration.

**output_frequency**: Specifies how often design results will be output and displayed in Plato during the optimization. A value of 5 means that every 5 iterations of the optimization a result will be outputted and displayed. A value of 0 means that output and display of the results will be turned off.

**max_num_optimization_itr**: Specifies the number of iterations Plato will run before stopping the optimization.

**filter_type**: Specifies the type of filter that Plato will use to smooth the design results. Possible values are “kernel” or “laplacian”. The kernel filter is the default filter and should work for most problems.

**filter_scale**: Specifies the scale factor used in applying the smoothing filter. The scale means different things depending on whether the “filter_type” is kernel or laplacian. If using a kernel filter the filter scale specifies the radius of influence the filter will have. In this case the filter scale is a value that will be multiplied by the average edge length in the mesh to get a filter radius. The default value is 3.5. If using the laplacian filter the filter scale is a value between 0 and 1 that indicates how aggressively the filter should be applied. In general, the larger the filter scale the larger the feature size in the design results.
**filter_absolute:** This parameter only has meaning if the filter_type is kernel. This parameter specifies an absolute value for the filter radius used in kernel smoothing. This parameter would be used in place of filter_scale. If both filter_scale and filter_absolute are specified filter_absolute will be used by Plato.

**filter_symPlane:** This parameter only has meaning if the filter_type is kernel. It is used to specify the reflection plane for symmetric problems to force better smoothing at the interface (no seams or creases). The symmetry plane must align with one of the coordinate axes, and an offset value can also be specified.

**filter_iterations:** Specifies the number of times the filter will be applied at each optimization iteration to smooth the results. 1 or 2 iterations will typically provide adequate results.

**fixed_block_ids:** Specifies the ids of the blocks in the mesh that should not be optimized. Ids should be separated by spaces.

**restart_index:** Specifies the index of the iteration that should be used as the initial guess when the optimization run is restarted. When running with an initial guess Plato will use the existing results in the remote working directory so it is imperative that you specify that Plato not clear the remote directory prior to submitting the job. You can indicate this in the Job Submission View.

**load_case_weights:** Specifies how to weigh the load cases against one another for the optimized design if the multi-load capability is utilized (see Section 8.1.1 for more information). The weights are normalized within the code so specifying weights of 3 and 1 is equivalent to specifying 0.75 and 0.25. By default, if no weights are specified, the cases will be considered equally.

### 7.2 Albany

The topology optimization parameters for Albany are found in the “basic” and “advanced” blocks of the input deck. We will only describe the topology optimization parameters that a user would typically modify in this manual. If you have questions about other topology optimization parameters please contact the Plato development team.

**target_volume_fraction:** Specifies the fraction of the starting design domain volume that the user wants in the optimized design. Fixed blocks are included in the calculation of the optimized design volume at each iteration.
**update frequency**: Specifies how often design results will be output and displayed in Plato during the optimization. A value of 5 means that every 5 iterations of the optimization a result will be output and displayed. A value of 0 means that output and display of the results will be turned off.

**max iterations**: Specifies the number of iterations Plato will run before stopping the optimization.

**filter radius**: Specifies the radius of influence of the filter used for smoothing the optimized design. This value should be at least 1.5 times the average edge length in the mesh. In general, the larger the filter radius the larger the feature size in the design results.

**restart time step**: Specifies the index of the iteration or time step that should be used as the initial guess when the next optimization run is started. When running with an initial guess Plato will use the existing results in the remote working directory so it is imperative that you specify that Plato not clear the remote directory prior to submitting the job. You can indicate this in the Job Submission View.

**fixed block id**: Specifies the id of a block in the mesh that should not be optimized. Add a separate “fixed block id” command for each block in the mesh that should be fixed.

**filtered block id**: Specifies the id of a block in the mesh that should have the filter applied to it for smoothing. Typically, you should enter a “filtered block id” command for each block in the mesh that is not “fixed” (see “fixed block id”).
8 Physics

When a topology optimization run is launched in Plato a finite element analysis program is being run in the background. Plato provides the interface to potentially utilize any finite element analysis code given the right level of integration. The following subsections describe the finite element codes that Plato currently supports.

8.1 Sierra/SD

Sierra/SD is a production-grade structural dynamics finite element analysis code developed at Sandia National Labs. Plato currently utilizes the linear statics capabilities in Sierra/SD for doing compliance minimization topology optimization. For more info see the Sierra/SD User’s Manual.

8.1.1 Multiple Load Cases

Multiple load cases can be used with Sierra/SD using the multi-case option in the Solution block in the input deck. Each solution case is delineated with the "case" keyword in the input deck as follows:

Solution
  case one
    topology optimization
    load = 1
  case two
    topology optimization
    load = 2
End

The topology optimization will be conducted considering both load cases equally if no weighting has been set, or using the weighting factors specified in the load_case_weights (see Section 7.1).

8.1.2 Body Loads

Plato supports incorporating gravity body loads in Sierra/SD into the optimization process for compliance minimization. An example Sierra/SD syntax gravity body load is shown below; no further tokens are needed for Plato to be aware of this load.

LOADS
  body
    gravity
    0.0 0.0 -1.0
Internal to the Plato code, the gravity body load is managed to accurately reflect where the body load should be applied; that is, the body load is applied in the Physics exclusively where material is present. If the gravity body load is the only load present or the dominant load, the developers have noticed that optimal topology can oscillate significantly even after many iterations.

8.1.3 RBAR Elements

Plato supports using the Sierra/SD RBAR element type for applying a load at a single location and distributing that load through rigid beam elements to other nodes. This loading approach is often used when representing a component by its center of mass and then distributing loading on this component to its interfaces with other components. Blocks consisting of RBAR elements must be specified as “fixed” blocks in the optimization since they don’t have material associated with them. A tutorial example problem using RBAR elements (called “Rbar”) is included in Plato.

8.2 Albany

Albany is an open source code that provides various physics simulation capabilities. It has proven to be a powerful research platform for exploring new topology optimization capabilities during the development of the Plato product. Plato provides an interface for doing topology optimization with Albany and the user is free to install Albany and use it in conjunction with Plato but the Albany capabilities are not considered production-grade and are not supported at that level. For more information about Albany see the Albany website.
9 Theory

The basic theory behind some of the topology optimization framework is briefly discussed next. For more in-depth coverage, we encourage the interested reader to refer to the Theory Manual, available upon request from the Plato development team at plato3D-help@sandia.gov.

9.1 Material Model

During the optimization process, the OptimalTopology will assume values in the range $[0, 1]$ where 0 signifies a void and 1 represents solid material. However, in order for the design to converge to mostly solid or void material, it is common practice to penalize the elements in the intermediate range (gray zones). Along those lines, the following material models can be used to perform the optimization.

9.1.1 SIMP: Solid Isotropic Material with Penalization

The Solid Isotropic Material with Penalization (SIMP) model uses a power-law relation to penalize the stiffness of elements compared to the amount of material used when OptimalTopology is neither 0 nor 1. This material is populated in the input deck by default with material_penalty_model = simp. To select the penalization power, the penalty_coefficient can be changed, which is set to a value of 3 by default. Another value, or a range of values, can be input to solve the problem with increasing penalization, for example penalty_coefficient = 1 1.5 2 2.5 3.

9.1.2 RAMP: Rational Approximation of Material Properties

Alternatively, the Rational Approximation of Material Properties (RAMP) model can be used by setting material_penalty_model = ramp. For this model, the penalty_coefficient is set in the same manner as the SIMP model, however, we note that a value of 3 for SIMP should be compared to a value of 4 for RAMP.
10 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 1.

10.1 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.

10.2 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.

10.3 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.

10.4 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”.

32
Figure 1: Models encountered during the topology optimization process in Plato.
References

