The LS-TaSC™ Tool

Topology and Shape Computations

User's Manual

Version 4.1

Corporate Address
Livermore Software Technology Corporation
P. O. Box 712
Livermore, CA 94551-0712

Support Addresses
Livermore Software Technology Corporation
7374 Las Positas Road
Livermore, CA 94551-0712
T: 925 449 2500
F: 925 449 2507
E: sales@lstc.com
W: www.lstc.com

Livermore Software Technology Corporation
1740 West Big Beaver Road, Suite 100
Troy, MI 48084-3507
T: 248 649 4728
F: 248 649 6328

PREFACE TO VERSION 4.1

Version 4.1, started in early 2019, is a minor release focusing on incorporating design for natural frequency using modern mathematical programming techniques. It contains the following features:

- Animations of the design iterations.
- Constraint bounds can be placed on a frequency. This is possible only for a single eigenvalue load case – full MDO will follow in a later release.
- Mode tracking for frequency constraints.
- Linear pentahedral and tetrahedral elements, as well as, the *CONSTRAINED_NODAL_RIGID_BODY keyword are supported for frequency design.
- User defined DSA for the global variables. This reduces the amount of FEA evaluation for constrained optimization. The user must be able to provide an estimate of the DSA values.

Thanks are due to many people. My colleague Guilian Yi was responsible for the various frequency optimization topics. Luo Liangfeng did the GUI development – our product has achieve high standards, especially from a GUI usability perspective. Imtiaz Gandikota served as our main contact and support engineer for the customers. Katharina Witowski from Dynamore, the editor of the example problem manual, suggested many improvements and coordinated support for the European market, while Åke Svedin, also from Dynamore, took care of matters computer science related. At the Livermore office thanks are also due to Philip Ho for managerial inputs regarding the LS-PrePost integration and to Yanhua Zhao for overseeing a smooth interaction with the contractors in China. Valuable feedback from customers is also acknowledged.

Willem Roux

Livermore CA,

November 2019

PREFACE TO VERSION 4

Version 4, started in the summer of 2016, is a major release focusing on incorporating design for natural frequency using modern mathematical programming techniques. It contains the following major features:

- Design for *natural frequency*,
- A *multidisciplinary design optimization formulation* considering the simultaneous design for large deformation, static, and NVH load cases, and
- The *projected subgradient design optimization method* enabling the above for huge FEA models.

It also contains the following minor features:

- Good convergence for *small mass fractions* (e.g. 0.02) using the projected subgradient method.
- *Different material models* can be used in the different disciplines.
- The *element deletion* behavior can be set to be different for implicit and explicit cases.
- Support for *encrypted data* is allowed. It won't be parsed, but will be passed on to LS-DYNA.
- Small *unconnected regions* can be automatically be deleted.
- For the *multi-tensor scheme* some defaults were changed to allow for fewer function evaluations. The starting iteration may be set and it can be controlled whether the load case weights will be used as global variables.

Thanks are due to many people. Firstly Luo Liangfeng, who did the integration with LS-PrePost and the latest GUI development – our product has achieve high standards, especially from a GUI usability perspective. My colleague Guilian Yi focused on the speed up of the optimization algorithms, various frequency optimization topics, and updating the manuals. Imtiaz Gandikota served as our main contact and support engineer for the customers. Katharina Witowski from Dynamore suggested many improvements and coordinated support for the European market, while Åke Svedin, also from Dynamore, took care of all matters computer science related. At the Livermore office thanks are also due to Philip Ho for managerial inputs regarding the LS-PrePost integration and to Yanhua Zhao for overseeing a smooth interaction with the contractors in China. Valuable feedback from customers is also acknowledged.

Willem Roux

Livermore CA,

June 2018

PREFACE TO VERSION 3.2

Version 3.2, started in the summer of 2015, is a minor release focusing on improving the GUI workflow and the multi-point constrained design optimization algorithm introduced in version 3.1. The GUI workflow was improved as follows:

- The interface was simplified for the average user by merging minor and obscure tools.
- Tooltips were added to the GUI to better explaining the defaults settings.

It also contains the following new features:

- Unconnected regions in a part can be identified and deleted.
- The job submission system has been updated to match LS-Opt version 5.
- The multi-point constrained design optimization method has been significantly upgraded internally for both better current use and future expansion.

Thanks are due to many people. Firstly Luo Liangfeng, who did the integration with LS-PrePost and the latest GUI development – our product has achieve high standards, especially from a GUI usability perspective. My colleague Attila Nagy took some time off from the NVH project to upgrade the job submission systems and to maintain the theory manual. Imtiaz Gandikota served as our main contact and support engineer for the customers. Katharina Witowski from Dynamore suggested many improvements and coordinated support for the European market, while Åke Svedin, also from Dynamore, took care of all matters computer science related. At the Livermore office thanks are also due to Philip Ho for managerial inputs regarding the LS-PrePost integration and to Yanhua Zhao for overseeing a smooth interaction with the contractors in China. Valuable feedback from customers is also acknowledged.

Willem Roux

Livermore CA,

June 2016

PREFACE TO VERSION 3.1

Version 3.1 was started late in 2013 focusing on updating the constrained design optimization algorithm. It contains the following major new features:

- Multi-point derivative scheme considering the derivative of the response with respect to the part masses and load case weights
- Optimization using mathematical programming and the multi-point derivatives
- Generalized constraints:
    - All LS-Dyna® output, similar to LS-Opt®
    - Mathematical expressions
- User-defined results
- Iso-surface plots of a design

Some minor features are:

- Element results filter radius can now be set relative to the element dimensions. It used to be global. The default was changed to be relative to the element.
- The *INCLUDE keyword is supported.
- The topology algorithm is restricted from deleting too many elements per iteration.
- The logic of the solid/void schemes has been clarified.

Many thanks are due to Luo Liangfeng, who did the integration with LS-PrePost and the latest GUI development – hopefully the project allowed him and everybody else professional growth. Imtiaz Gandikota, our LS-TaSC support contact, performed many QA tasks, specifically GUI testing, along with some usability contributions, and served as our main contact with customers. The iso-surface plots were contributed by David Wynn together with airbag results access. Attila Nagy also joined the group, improved the theory manual, and contributed some usability suggestions. At the Livermore office thanks are also due to Philip Ho for managerial inputs regarding the LS-PrePost integration and to Yanhua Zhao for overseeing a smooth interaction with the contractors in China. Katharina Witowski and Peter Schumacher from Dynamore helped improve the manual, specifically the example problems, suggested many improvements, as well as working with the European market, while Åke Svedin maintained our development tools. Valuable feedback from customers and co-workers is also acknowledged, specifically Honda R&D Americas and JSOL, one of our distributors in Japan.

Willem Roux

Livermore CA,

April 2015

PREFACE TO VERSION 3

Version 3 was started in spring of 2012 focusing on free surface design as well as imbedding the LS-TaSC product into the LS-PrePost framework. Version 3 is an important step forward containing the following major new features:

- Free surface design of solids including
    - Geometry definitions
        - Extrusions
        - Symmetry
        - Edge smoothing
    - Automatic mesh smoothing
- Integration into the LS-Prepost framework. This is a long term project which at this stage includes:
    - Expanding the previous GUI capabilities for free surface design
    - The model tree on the left on the screen allowing quick navigation of the LS-TaSC model
    - Picking of parts and surfaces
    - Integrated editing of the LS-DYNA FE model to create surfaces, coordinates systems, and other entities required for the LS-TaSC design.

Some minor features are:

- Support of *MAT_ORTHOTROPIC_ELASTIC for the topology design of solids.
- Support of the d3part database for reading field results.
- The LCSS curve option of *MAT_PIECEWISE_LINEAR_PLASTICITY is now supported.
- Checking and adding the LS-Dyna binary output requests required for constraints
- The iteration count now starts at 0, with iteration 0 being the initial design provided by the user.
- The Material Utilization plot is now scaled with the value of the target field value. A value larger than 1 indicates that an element is highly used, while a value smaller than 1 indicates that an element is lightly used.
- Existing lst_output.txt files will be copied to a new name, instead of being appended to, if the environment variable LSTASC_SEPARATE_OUTPUT is set.

Many thanks are due to Luo Liangfeng, who did the integration with LS-PrePost and the latest GUI development – Luo had to master many topics in order to achieve this. At the Livermore office thanks are due to Philip Ho for managerial inputs regarding the LS-PrePost integration and to Yanhua Zhao for overseeing a smooth interaction with the contractors in China. Valuable feedback from customers and co-workers is also acknowledged.

Willem Roux

Livermore CA,

July 2013

PREFACE TO VERSION 2.1

Version 2.1, started in spring of 2011, is a refinement of version 2. It contains the following major new features:

- *Dynamic load case weighting.* This algorithm obtains a design equally relevant for all design load cases.
- *Forging geometry definition.* This geometry definition is similar to a two-sided casting except that a forging thickness is introduced.

New minor features are:

- Castings can have interior holes.
- Pentahedral elements are supported.
- The memory footprint is reduced more than a factor of 2 and an option is provided which can be set to reduce memory use by a further factor of 2.
- *MAT_ELASTIC is supported for the design part.
- Lightly used elements can be kept instead of deleted.
- The SIMP algorithm can be switched on and off.
- Coordinate systems are no longer limited to DIR=X.
- Restarting was improved to be faster by using more archived results.
- A fringe plot of the material utilization as considered in the design process can be viewed.
- The fraction of the original number of elements used in the design can be viewed as a history.
- The global constraint handling has been changed to consider only active constraints. If no global constraints are active anymore, then the algorithm will slowly return to the user specified mass fraction.

Many thanks are due to David Björkevik for the GUI design and implementation. Valuable feedback from customers and co-workers is also acknowledged.

Willem Roux

Livermore CA,

November 2011

PREFACE TO VERSION 2

Version 2 was started in spring of 2010 in response to industrial feedback regarding version 1. Version 2 is an important step forward containing the following major new features:

- Shell structure support
- Global constraints
- Multiple parts
- Symmetry definitions
- Casting direction definitions

Some minor features are:

- Tetrahedral solid element and triangular shell element support
- The speed of some algorithms was improved
- Improved integration with LS-DYNA

Many thanks are due to David Björkevik for the GUI design and implementation, Tushar Goel for the initial global constraints implementation, and Trent Eggleston for assistance with distributed computing. Valuable feedback from customers and co-workers is also acknowledged.

Willem Roux

Livermore CA,

January 2011

PREFACE TO VERSION 1

The development of the topology code started in the fall of 2007 in response to a request from a vehicle company research group. The alpha version was released in the spring of 2009 to allow the vehicle company research groups to give feedback from an industrial perspective, while the beta version was released in November 2009.

Most of the methodology developments in version 1.0 are due to Tushar Goel who worked on the engine implementation and algorithm design. Additionally, he also wrote the manual together with Willem Roux.

The project architecture was the responsibilities of Willem Roux and David Björkevik. David had the lead role with regard to the graphical user interface aspects, while Willem had the senior role looking after the overall project and the project management.

Thanks are also due to Nielen Stander from LSTC who helped to coordinate the efforts in the LS-OPT group and sourced the initial version of the technology, John Renaud and Neal Patel for discussion regarding topology optimization, Kishore Pydimarry and Ofir Shor for evaluating the alpha version, and Fabio Mantovani and Stefano Mazzalai for their help with LS-DYNA simulations.

Willem Roux

Livermore CA,

January 2010

TABLE OF CONTENTS

1. Introduction

1.1. Classification of Structural Optimization Techniques

Engineering optimization finds new designs that satisfy the system specifications at a minimal cost. Different types of structural optimization are described in the following sections.

1.1.1. Topology Optimization

This is a first-principle based approach to develop optimal designs. In this method, the user needs to provide the design domain, load and boundary conditions only. The optimal shape including the shape, size, and location of gaps in the domain is derived by the optimizer. While the most flexible method, topology optimization is indeed the most complex optimization method due to a multitude of reasons, like, large number of design variables, ill-posed nature of the problem, etc. Nevertheless, the benefits of using topology optimization include the possibility of finding new concept designs that have become feasible due to recent advances in technology, e.g., new materials. The LS-TaSC program can be used to this design work.

1.1.2. Topometry Optimization

Topometry optimization, a methodology closely related to topology optimization, changes the element properties on an element by element basis. With the LS-TaSC program, the shell thicknesses can be designed.

1.1.3. Size Optimization

In this mode, the designer has already finalized the configuration of the system but improvements are sought by changing the thickness of members of the structure on a part basis instead of an element by element basis as done for topometry optimization. There is usually no need to re-mesh the geometry. This class of optimization problems is the most amenable to meta-model based optimization. The LS-OPT® program should be used for this instead of this program.

1.1.4. Shape Optimization

Shape optimization further expands the scope of design domain by allowing changes in the geometry of the structure, for example the radius of a hole. While there is more freedom to explore the design space, the complexity of optimization increases due to the possible need to

mesh different candidate optimum designs. We distinguish between two methods of doing shape design: using free surface shape design and using parameters.

Firstly you can do free surfaces shape design as with this program. This approach is very easy to use, but has the drawback of not being very general.

Secondly you can do shape design using parameters such the radius of a hole or shape vector magnitude. This is a very general approach, able to consider all crash specific constraints. Use the LS-OPT® program together with a preprocessor such as LS-PREPOST® instead of this program.

1.2. Topology Optimization Method in LS-TaSC

The product has two algorithms for the topology design: (i) an optimality criteria method named the Optimality Criteria for Dynamic Problems and (ii) a mathematical programming approach named the Projected Subgradient Descent. Our choice of methodology is mostly driven by (i) the lack of gradient information for crashworthiness problems, (ii) the very large problems being considered in practice, and (iii) the integration of topology optimization into the overall design process. The optimality criterion, that of an uniform internal energy density, has a long, successful history in engineering design (see, for example, Venkayya's work in 1968). The advantage of the Projected Gradient Descent Method is that it can design for both crash and NVH simultaneously by include analytical gradient information from linear, implicit computations. Another large theoretical expansion is the computation of numerical gradient information for crashworthiness problems using the multi-tensor method. Our methodology is therefore always evolving in major directions. A more complete description can be found in the LS-TaSC Theory Manual.

1.3. Finding Information

This manual is divided into parts. The user's manual describes how to do topology optimization using LS-TaSC. A few examples are provided to cover different options in the topology optimization program. Some common errors and tips on troubleshooting are provided in a separate chapter. The scripting manual lists the command language used to interact with the topology optimization code together with some examples. In the theory manual, the method for topology optimization is described. Setting up queuing systems is described yet another manual. All manuals are bundled with the executables and can be found in the same location after installation.

2. Topology Optimization

Topology optimization computes the lay-out of a structure: where material should be located to provide a loadbearing structure. The criterion is that the material should be fully used; this is implemented by designing for a uniform internal energy density in the structure while keeping the mass constrained. The outcome is the stiffest structure for the given weight (minimum compliance design).

2.1. The Design Parts

The design domain is specified by selecting parts – the optimum parts computed will be inside the boundaries delimited by these parts. The part must be defined using *PART, not *PART_*OPTION*. The parts may contain holes: a structured mesh is accordingly not required.

2.1.1. Design of Solids

The designed topology of a solid part is described by the subset of the initial elements used. Unused material will be removed during the design process thereby revealing the structural shape that can bear the loads efficiently. The amount of material removed is specified by the user through the mass fraction parameter.1

Each solid element is controlled by changing the amount of material in the element. This is achieved by assigning a design variable to the density of each element. The design variable *x,* also known as relative density, varies from 0 to 1 where 0 indicates void and 1 represents the full material. The upper bound on the design variable is 1, while elements with design variable value less than a user-defined minimum value (0.05 for dynamic problems, and 0.001 for linear) are deleted to improve numerical stability.

In this approach, the design variable is linked to a material with the desired density. The material properties are obtained using an appropriate interpolation model as described in the theoretical manual.

The final design variable value for each element will be driven to full use of the element (the maximum value of 1) or deletion of the element (values below the user-defined minimum) using the SIMP algorithm described in the theoretical manual. The use of the SIMP algorithm can however be de-activated using the advanced options described later in this chapter, in which case the design variables will have intermediate values selected to achieve a uniform internal energy density in the part.

2.1.2. Design of Shells

For shells the thicknesses are changed to achieve a uniform internal energy density in the part. The upper bound on the design variables is the original shell thicknesses, while elements with shell thicknesses less than a user-defined minimum value (0.05 for dynamic problems, and 0.001 for linear) are deleted to improve numerical stability.

The final shell thicknesses will have values varying between the original shell thickness (the maximum value) and the user-defined minimum value, if not deleted for stability reasons. The shell thicknesses will not be driven to the maximum or minimum values using the SIMP algorithm described in the theoretical manual. The SIMP algorithm can however be activated using the advanced options described later in this chapter, in which case the behavior will be similar the default behavior for solids.

2.1.3. Element types

Solid elements must be eight-noded solid elements, four-noded tetrahedral elements, or six-noded pentahedral elements. Equilateral element shapes are the best for the current algorithm.

Shell elements may be four-noded shell elements or three-noded shell elements. The triangular elements must be specified as four-noded shell elements by specifying the last node twice. Equilateral element shapes are the best for the current algorithm.

Tetrahedral and triangular elements cannot be used in an extrusion geometry definition.

2.1.4. Material data

For the design of a shell structure any material can be used, while the design of solid structures is limited to the use of certain materials for the design part.

For the topology design of solids the design parts must be modeled using *MAT_ELASTIC, or *MAT_ORTOTROPIC_ELASTIC, or *MAT_PIECEWISE_LINEAR_PLASTICITY.

For some *MAT_PIECEWISE_LINEAR_PLASTICITY material data the topology algorithm (SIMP algorithm) will create materials for which the slope of the stress-strain curve is higher in plastic regime than in the elastic one; in this case the errors and warnings in the LS-DYNA output should be consulted for feedback on how to modify the material stress-strain curve in the LS-DYNA input deck.

2.1.5. Solid/Void behavior

The best design of a structure modelled using solid elements would normally consists of a large amount of porous material. Such a structure cannot be manufactured. For the final design it is desired that a elements be completely filled with material (solid) or completely unused (void). For shells this is not an issue.

There are several schemes enforcing this desired solid/void behavior. SIMP, the academic standard, severely reduces the stiffness of lightly used elements. The gradual SIMP approach, more commonly known as SIMP with continuation, gradually applies the SIMP effect over several iterations. The “True Mechanics” approach, unique to the crash industry, is more accurate at predicting energy absorption and displacements during earlier iterations.

The use of SIMP have resulted in many problems when applied to industrial problems. Specifically, the displacements and energies of the models will also be very high during initial iterations because the scaling of the stiffness. For a mass fraction of 0.3, this may result in a factor 10 difference; for a mass fraction of 0.01, the potential difference grows to an astonishing 10000. Because of this, the SIMP scheme is not recommended for industrial problems involving highly nonlinear mechanics and constraints. The “True Mechanics” is recommended as well as the default. The exception being for problems involving the fundamental frequency for which a SIMP approach must currently be used.

For shells the default is to use no such scheme, because this is not an issue.

2.2. Geometry and Manufacturing Definitions

For each part several geometry and manufacturing definitions such as being an extrusion may be specified.

The geometry definitions, as shown in Figure 2-1, are:

- *Symmetry.* For these the geometry is duplicated across a symmetry plane. The part as supplied by the user must be symmetric: an element must have a matching element on the other side of the symmetry plane.
- *Extrusion.* An element set is extruded in a certain direction. Allowable set definitions are *SET_SOLID, *SET_SOLID_LIST, *SET_SHELL, and *SET_SHELL_LIST. The part as supplied by the user must be an extrusion with every element in the elements set must have the same number of extruded elements. Only hexahedrons and quadrilateral elements can be extruded.
- *Casting.* Material is removed only from a given side of the structure. The structure therefore will have no internal holes. The casting constraints can be one sided or two-sided. This capability is available only for solids.
- *Forging.* This is similar to a two-sided casting, except that a minimum thickness of material will be preserved. The geometry definition will therefore not create holes through the structure.
- *No interior holes.* This prevents interior holes (holes not connected to the outside). It can also preserve features smaller than a specified size, which is a capability intended to be use with lattice structures.

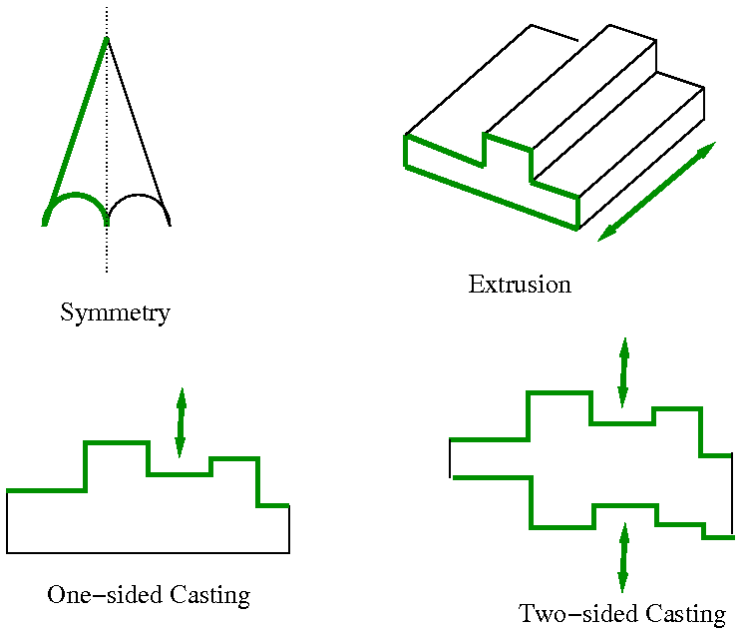

Figure 2-1: Geometry definitions

Multiple geometry constraints can be specified for each part. Some combinations of geometry constraints may however not be possible. A maximum of three geometry definitions per part is possible. The symmetry planes must be orthogonal to each other, the extrusion direction must be on the symmetry planes, the casting direction must be on the symmetry planes, and the extrusion directions must be orthogonal to casting directions. Only one casting definition may be defined per part.

The symmetry and extrusion definitions are implemented by assigning multiple elements to a variable, while the casting definitions are implemented as inequality constraints requiring certain variables to be larger than others according to the cast direction.

For a casting definition, the free faces are selected as shown in Figure 2-2. It can be seen that that free faces can occur in many places, for example, inside a hole, which cannot be created using a casting manufacturing process. In version 2.1 onward the algorithm will ignore the internal cavities in the selection of the free surface. This is to allow an analyst to have cavities introduced say by drilling into a cast part. All of the material shown can be considered to be defined using a single *PART definition, from which it can be noted that the object to the right is considered for design even though it is in the ‘shadow’ of the object to the left. An analyst can enforce a complex behavior by breaking the part up in smaller parts and applying the casting definition only where desired.

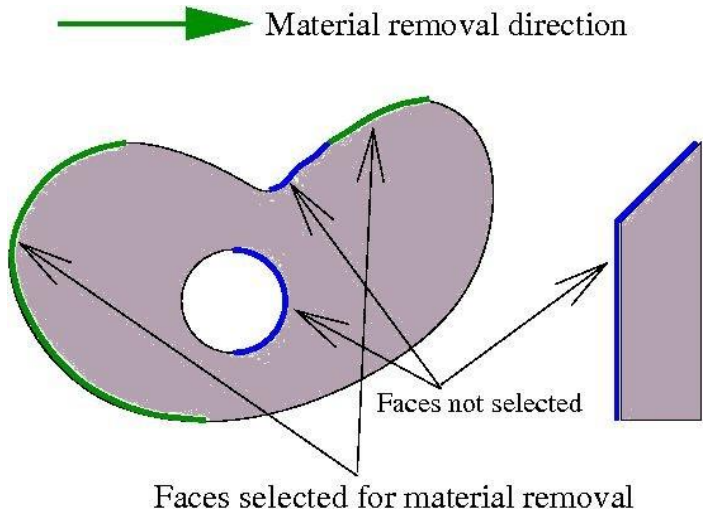

Figure 2-2: The faces selected for design in a casting definition are all the faces facing the material removal direction. The algorithm will not consider the faces shown in blue.

2.2.1. Investigating casting problems

The results can be dumped and displayed in LS-PrePost as user-defined results.

2.3. Design algorithms

The recommended method is the projected subgradient method. This is a modern mathematical programming method allowing for multi-disciplinary optimization of huge problems considering small mass fractions. Specifically the combination of explicit dynamics, linear statics, and NVH cases are allowed.

The optimality criteria method, our older method, is still available for backward compatibility. This is a very established and widely tested method documented in most textbooks. The underlying optimality criteria was rigorously shown to be a sufficient and necessary condition for the optimal design of linear structures.

2.4. Convergence

For solids, considering that the SIMP model drives the element to an either fully used or deleted state, it is useful to monitor the fraction of elements used for convergence. This will converge to the mass fraction of the part if the elements are of uniform size.

The projected subgradient method monitors the solidification of the part for convergence. The solidification measure which fraction of the elements is either fully used or deleted. The value will be one for a fully converged design, that is a design with all elements either fully used or deleted.

The optimality criteria algorithm monitors the mass redistributed per iteration for convergence. Ideally this number will be zero for a converged design, but in practice it only goes down to a small number.

Typically the problem is converged in less than 30 iterations, but this is not guaranteed. The projected subgradient typically converges the fastest.

2.5. Design Variables

2.5.1. Mapping Elements to the Design Variables

A design variable is assigned to every finite element in the design parts. For geometry constraints, the variables are defined only on a subset of elements.

2.5.2. Filtering of Results

Using only the result at a specific element to do the design update of that specific element will lead to a checkerboard design pattern. The result used for the element design update is therefore computed using a scaled combination of it the element's value and that of its neighbors. A radius based strategy is used to identify neighbors. In this strategy, a virtual sphere (of default or user-defined radius) is placed at the center of an element. All the elements within this sphere are considered the neighbors of the corresponding element. The filter radius can be specified to constant over the whole part or relative to each element's size.

Mesh independent designs can be achieved by using the same global filter radius for the different meshes.

For dynamic problems, it was observed that accounting for the history of evolution induces stability by reducing the element deletion rate. Hence, the field variable (internal energy density) of i^{th} cell at iteration t is updated by defining a weighted sum on the field variable of three previous iterations.

2.5.3. Initialization, Deletion, and Regeneration of the Design Variables

The design variables are initialized to satisfy the mass fraction. All variables in a part are assigned the same initial value. All associated field variables are also initialized to zero.

The variable value of the element depends on its loading together with that of its neighbors due to filtering. If the variable value is too low, then the element is removed from the model once the variable value is smaller than the minimum allowable value. The defaults are 0.05 for explicit and 0.001 for implicit. The element can be kept in the model in later or all iterations by decreasing this minimum allowable value of the variable fraction, but this may result in instability of the FE model. If an structural evaluation is comprised of a mixture of explicit and implicit analyses, then the methodology can be set such that the elements are deleted only for the explicit analyses, but not for other – in this case the minimum variable values of the implicit

load cases will be set the default for implicit, while for the explicit analyses, the elements will be deleted at the minimum allowable variable value as provided by the user.

The element will be regenerated if its neighbors are highly stressed in a later generation. Unless the neighborhood radius is set close to 0, in which case it won't be regenerated, because it does not receive any information from its neighbors.

2.6. LS-DYNA® Modeling Specifics

The portions of the FE model related to the design parts are extensively edited by the optimization algorithm. In these segments of the FE model only specific versions of *PART, *SET, and *CONTACT keywords may be used as described in the relevant sections. Portions of the model not edited by the optimization algorithm are not subjected to this rule.

2.6.1. Contact Definitions

This discussion applies only to solid structures. For the design of shell structures no action is required, because the part and contact definitions will not be edited by LS-TaSC.

Contact involving a solid design part requires special handling, because a design part ID is changed by the topology algorithm. There are two options to model contact involving the solid design parts: defining contact using part sets, or using specific *CONTACT_AUTOMATIC_[OPTION] definitions.

Firstly the contact can be defined using part sets containing the design part. LS-TaSC will rewrite part sets to reflect changes to the design part. This will allow any *CONTACT definition to be used.

The alternative option is modeling the contacts involving the design parts using either the *CONTACT_AUTOMATIC_SURFACE_TO_SURFACE[_ID] or the *CONTACT_AUTOMATIC_SINGLE_SURFACE[_ID] options. These automatic contact options are general enough to accommodate the changes in the geometry of the design parts during the optimization to maintain valid contacts.

Other contact types are not edited by LS-TaSC. They can be used (*i*) if the contact does not involve the design part or (*ii*) if the contact is defined for a part set containing the design part, because LS-TaSC will rewrite part sets to reflect changes to the design part.

It is also recommended to specify the contact options (e.g., friction coefficients) appropriately accounting for the changes in the geometry may result in significantly different material properties for some elements near the contacts. Too restrictive values may cause instabilities in the LS-DYNA® simulations for intermediate geometries.

LS-TaSC will set the SOFT=2 on the optional card A to improve contact behavior if the optional card A is not specified for the contact types named in the first paragraph. This can be overridden by specifying the optional card A.

2.6.2. Part Definition

The part must be defined using *PART, not *PART_*OPTION*.

2.6.3. Part Set Definition

The part sets involving the design parts should be defined using *SET_PART or *SET_PART_LIST. Neither the *generate* nor the *column* options are edited by LS-TaSC – do not use these options to include the design part.

2.6.4. Element Set Definition

The sets involving the design parts should be defined using *SET_SOLID, *SET_SHELL, or *SET_SHELL_LIST. Neither the *generate, general, list_generate,* nor the *column* options are edited by LS-TaSC – do not use these options to include the design part.

2.6.5. Coordinate Systems

Only coordinate systems defined using *DEFINE_COORDINATE_NODES can be used in geometry definitions. Others will be passed on to LS-DYNA and otherwise ignored by LS-TaSC.

2.6.6. Include file

The *INCLUDE keyword is supported from version 3.1 onwards. The content of the included file will be transcribed to the lst_master.k file. Note that when using queuing systems the content of the included file will be copied to the remote node. The *INCLUDE_*OPTION* (e.g. *INCLUDE_TRANSFORM) keywords will not be interpreted, and these keywords therefore cannot be used to include a design part, nor will the content be transferred to a remote disk.

2.6.7. Encrypted input

Encrypted input will not be read, but will instead simply be passed on to the LS-Dyna solver. Information critical to the functioning of LS-TaSC must therefore not be encrypted. The encrypted input should be in ASCII and included using *INCLUDE.

2.6.8. Disallowed Keywords

In general, all keywords are allowed, but LS-TaSC will only edit the listed keywords to reflect changes to the design part.

The *PARAMETER keyword is not supported. The keyword will be echoed to the LS-TaSC run files, but an error will occur should LS-TaSC need to read or edit an input card containing the parameter.

See the section regarding designing for the fundamental frequency for restrictions relative to this topic.

2.6.9. Automatic Keyword Edits by LS-TaSC

Automatic keyword edits preserve the stability of the LS-DYNA simulation by deleting elements that are inverting or have a very small timestep. The values of variables are reset as in the following table. The user can override the values supplied by LS-TaSC using the information in the table.

Table 2-1: Automatic Keyword Edits by LS-TaSC

Keyword	Variable	LS-TaSC Auto Set	User override
CONTROL_ TIMESTEP	ERODE	1 (erode elements)	Set to -1 to force of value of 0 (no erosion)
CONTROL_ TERMINATION	DTMIN	0.01 if less than or equal to 0.	Set to a positive value to force the use of this positive value

Remarks:

1. DTMIN was set to 0.001 in versions before version 3.0., but this value was increased on the basis that larger values should be more useful for topology design. Aggressively large values in topology design may results in critical load paths being deleted during the evolution of the structure.
2. TSMIN = DTMIN * DTSTART, with TSMIN the minimum timestep and DTSTART the initial timestep. Elements with a smaller timestep will be eroded. Alternatively the analysis terminates if element erosion is inactive and the timestep falls below TSMIN.
3. In version 2.1 and earlier the value of TSSFAC in CONTROL_TIMESTEP was set to 0.9. The default for TSSFAC is 0.9 (or .67 for high explosives), so setting it to 0.9 was discontinued.
4. The use of PSFAIL on *CONTROL_SOLID overrides the ERODE setting.

2.6.10. LS-DYNA® Simulation

The modified input deck is analyzed using LS-DYNA®. One can take advantage of multiple processors using the MPP version of LS-DYNA® by specifying the simulation options as part of the command. Queuing system can also be used as described in Section 4.5.2.

If you desire to use less disc space, then the options are to reduce the LS-Dyna output or to create a file named “clean” (“clean.bat” in Windows) in the directory containing the database. This “clean” file must be set to be executable and can contain lines such as “rm –rf d3hsp scr00*”. LS-TaSC will execute this “clean” script in every directory where LS-DYNA ran successfully.

2.6.11. Troubleshooting

See the separate chapter on troubleshooting.

2.7. Design for the Fundamental Frequency

This feature is developed for use in multidisciplinary applications where the fundamental frequency is one of several load cases alongside other load cases such as impact and statics.

The fundamental frequency is that of the whole structure which can be composed of many parts. Only one part is allowed to be a design part if designing for the fundamental frequency. There must a load case specifying an eigenvalue analysis.

Designing for only the fundamental frequency load cases is characterized by two problems:

- Converging eigenvalues. The second eigenmode can converge with the first, which causes the solution to oscillate between the two modes.
- Regions with low stiffness. This low stiffness region can develop the lowest eigenmode in the structure, at which point the algorithm starts redesign this region. Visually it can be observed that the region starts behaving like a vibrating jelly which then gets redesign to have a microstructure with a higher speed of sound. All of which are undesirable.

Workarounds are described the *options* section below.

2.7.1. Objective

The objective can only be the fundamental frequency. If you are interested in another mode, then the solution is to apply a constraint bound to that mode.

2.7.2. Constraining a specific frequency

You can put an upper bound, a lower bound, or both on a specific frequency. Mode tracking will be used – in the base analysis the mode shape is noted, and in subsequent iterations the constraint is applied to that specific mode. This is currently not possible for studies containing multiple load cases or multiple parts – the full multidisciplinary ability will follow in a later release.

2.7.3. Required LS-DYNA® Version, Keywords, and Element Types

The LS-DYNA sensitivity analysis capability is only for models created using linear hexahedral, pentahedral, and tetrahedral elements.

The LS-DYNA executable must be version 11 or newer to design for the fundamental frequency.

This version, unlike the previous version, supports linear pentahedron and tetrahedron elements as well as *CONSTRAINED_NODAL_RIGID_BODY keyword.

The LS-DYNA input deck must of course be set up as an eigenvalue analysis. The number of eigenvalues requested should be at least 4.

2.7.4. Options

The projected subgradient descent method must be used.

Multidisciplinary problems, for example a static load case together with a fundamental frequency load case, is numerically more stable than designing for only a frequency load case.

It may help to set the desired mass flow (which controls the optimization step size) lower. For smoother convergence try say a value of 0.02 or even “0.5*Default” (the default is actually a function). This can prevent oscillations in the solution. This is a solution in particular for converging eigenvalues – with a smaller step sizes the oscillations between the eigenmodes are smaller. The desired mass flow option can be found in the method panel in the user interface.

Specify symmetry conditions where possible. This will result in quicker and better convergence for symmetric structures.

Small geometric features can cause oscillations in the design, specifically if the feature size is close to the element size. Setting the part neighbor radius larger will prevent the formation of small features.

Setting element deletion can be useful, as well as the deletion of unconnected regions. There is an option to use element deletion for the explicit load cases, but not for the implicit load cases.

2.8. Constrained Optimization using the Multipoint Algorithm

Constrained optimization uses the part mass fractions and load case weights as global variables. It is therefore the values of these global variables that are used in the constrained optimization and not that of the element density variables, which are computed from the values of the global variables. This design process uses a multi-point approach: either to compute derivatives with respect to the global variables or by evaluating a number of designs selecting randomly around the last best one.

It should be understood that the structure is always designed to be stiffest structure possible, but with the global variables selected such as to satisfy the constraints. Displacement constraints, being directly related to the stiffness of the structure, are good choices of constraints. But the topology design process will not resolve issues such as stress concentrations.

2.8.1. Methodologies

Several methods are available:

- *Standard* This is standard constrained optimization. Specify an objective and constraints with bounds. The values of the global variables will be computed for this optimization problem.
- *Standard with variable splitting* The mass fraction variables are used differently from the load case variables in this algorithm. The mass fractions are set to satisfy the constraints, while the load case weights are set to satisfy the dynamic weighing.
- *Random search* Designs with different values of the global variables are selected using randomly and evaluated. The best design is selected and used as the starting point for the next design cycle. The selection of random designs is done in a subregion around the last best design.

2.8.2. Global variables move limits

The algorithm evaluates a number of points in a region around the design of the previous iteration. The bounds of this region, the move limits, are relative to the current value of the global variable, can be set by the user, and are available for plotting as part of the design histories.

Currently the move limits are set using an equation; e.g. "0.05+0.05*exp(-(lst_Iteration-1)/10.)", which means that the move limit is decreased as the iteration count (lst_Iteration) increases. The move limits are started large and then decreased in order for the global variables to converge before the local variables. If the local variables are closed to converged, then large changes in the global variables are no longer possible – in fact, the update of the global variables will be terminated if this situation is detected.

The move limit can be set to a constant value, e.g. 0.1, but the move limits will still be relative to the global variable value. So if the move limit is set to 0.1, and the global variable has a value of 0.2, then global variable values will be restricted to the interval [0.18,0.22] for that iteration.

2.8.3. Approximations

The approximation, if requested, will always be a linear Taylor expansion constructed using the derivatives of the constraints. Different ways of computing the derivatives used to construct the approximation are available:

- *Forward differences* The standard forward differences numerical differences algorithm.
- *Central differences* The standard central differences numerical differences algorithm.
- *RSM* A linear response surface (least squares fit) is created. The global variables are selected using a space filling designs as described in the LS-OPT manual.
- *Random* Designs are selected randomly around the previous best design using the Latin Hypercube Design as described in the LS-OPT manual. In this case no approximations are used.

2.8.4. Reducing the number of FEA evaluations

The number of FEA evaluations (calls to LS-DYNA) is controlled by the DSA computation frequency, the sampling scheme, and the starting iteration. For example, if the DSA frequency is set to 5, then the multipoint scheme will only compute the derivatives every fifth iteration. Therefore a large value of the DSA computation frequency together with a forward differences sampling will result in the smallest number of FEA evaluations. This will reduce the accuracy for some structures. For a well behaved structure or less accurate results, try a starting iteration of 5, a DSA frequency of 10, and a forward difference sampling scheme. For more accurate results a central difference scheme and a DSA frequency of 1 is best.

An alternative is for the user to specify estimates of the values of the derivatives. In this cases no FEA calls are made to compute the derivatives. If this option is selected, then the program will create a sample file with user defined values, and stop after the first iteration asking the user to inspect this file. When the program is restarted, it will use the derivatives provided in that file. If the file already exists in the directory, then the program will not stop after the first iteration.

2.8.5. Best practices

Always scale the constraints to have a bound of -1 or 1. For example, instead of *s < 10e6* and *u < 0.0001*; first created scaled variables such as $s_{scaled} = s / 10.e6$ and $u_{scaled} = u / 0.0001$, and then give the constraints as $s_{scaled} < 1$ and $u_{scaled} < 1$.

The use of dynamic load case weighing tend to result in a better solution. For example, instead of specifying $u_1 < 0.001$ and $u_2 < 0.002$; specify $u_2 = 2*u_1$ using dynamic weighing together with $u_{1\_scaled} = u_1 / 0.001$ and u*1_scaled < 1*.

Equality constraints can be specified by specifying the lower and upper bounds to be equal. Equality constraints does not work well in the current version, especially if other constraints are present.

The best point sampling scheme in general is the central difference scheme. Use the forward difference scheme to reduce cost or if the derivatives are well behaved.

Superior numerical stability of the optimization scheme can be obtained by using our LS-OPT product to control the LS-TaSC global variables. The better numerical stability is at the cost of more LS-DYNA evaluations.

2.9. Dynamic Load Case Weighing

The choice of the load case weights is critical for designing for multiple load cases. A single load case may dominate the topology of the final design thereby making the structure perform badly for the other load cases. This can be resolved by assigning different weights to the load cases, but it is difficult to know the values in advance. Dynamic weighing of the load cases is used to select the load case weights based on the responses of the structure as the design evolves, thereby resulting in a design that performs well for all load cases.

A dynamic weighing relationship is defined considered the responses of all the load cases. For, example, you may require that all load cases have equal displacements. The algorithm will then select the load case weights to achieve this relationship. Say we have constraint C_1 from the first load case and constraint C_2 from the second load case, then we write our desired behavior as $k_1 C_1 + offset_1 = k_2 C_2 + offset$ with *C* the constraint value, *k* a scale factor, and an offset added.

There are several methods of computing the load case weights:

- General constrained optimization as described in the next section will use the load case weights as variables. This can be done in two ways:
    - Simply specify the problem as a constrained problem. If the problem contains multiple load cases, then the load case weight ratios will be used as design variables, and the load case ratios will be such as to satisfy the constraints.
    - Specify a dynamic weighing relationship and set the optimization method to this option. In this case the mass fraction variables and the load case weight variables will be treated differently: the load case weights will be used to satisfy the dynamic weighing, while the mass fraction variables will be used to satisfy the constraints.
- There is also an older version of solving doing dynamic weighting employing some heuristics instead of numerical derivatives. It has the advantage of needing less computational effort. The disadvantages are that it only works for specific responses, though it specifically works for the case of displacements which is the most common one. This option can be specified in conjunction with the older global constraints option.

The above is better illustrated using an example. With two load cases, you can define two constraints:

$$g_{lc1} < a$$
$$g_{lc2} < b$$

Say the two constraints depend on the global variables, M_p, w_1 and w_2, which are the mass fraction of part *p*, the load case weights for load case 1, and the load case weights for load case 2 respectively. Then we can solve using the general constrained optimization formulation as:

$$g_{lc1}(M_p, w_2) < a$$
$$g_{lc2}(M_p, w_2) < b$$

with w_1 taken as constant, because it is ratio of weights that are important not their absolute values.

Alternatively, we rewrite to have the dynamic weighing formulation, in which we solve for the mass fraction and load case weight ration separately using either constrained optimization or the older heuristic methods:

$$g_{lc1}(M_p) < a$$
$$\frac{1}{a} g_{lc1}(w_2) = \frac{1}{b} g_{lc2}(w_2)$$

In the dynamic weighing formulation the constraint bounds need not be enforced, so the mass fraction can be constant. Which mean that you may enforce only the following:

$$\frac{1}{a} g_{lc1}(w_2) = \frac{1}{b} g_{lc2}(w_2)$$

It is also fine to keep both of the original constraints in the problem statement:

$$g_{lc1}(M_p) < a$$

$$g_{lc2}(M_p) < b$$

$$\frac{1}{a} g_{lc1}(w_2) = \frac{1}{b} g_{lc2}(w_2)$$

The final weights computed are not suitable for restarting. They can be examined though for an indication of good values of the weights. But usually the final weights computed using dynamic weighting will be different from the constant weight values required to compute the same structure.

2.10. Simple Global Constraints using a Single Mass Fraction

This is an older version of solving for constraints. It has the advantage of needing less computational effort. The disadvantages are that it only works for one part and only specific types of responses, though it specifically works for the common cases of stiffness and compliance. In general the multi-point method is preferred above this method – the forward differences sampling scheme together with a large value for the DSA computation frequency can be used if a similar number of LS-DYNA evaluations is required.

Global responses depend on the design of the whole structure. Two types of global responses are:

- *Stiffness*. This is specified as displacement constraint.
- *Compliance*. This is specified as a reaction force constraint.

Satisfying the global constraints is done as a search for the mass of the structure. If the displacements are too large, then mass are added to the structure to increase the stiffness. If the reaction forces are too large, then mass is removed from the structure to reduce the force.

It should be understood that the structure is always designed to be stiffest structure possible, and in this process the mass fraction is adjusted to satisfy the constraints. Displacement constraints, being directly related to the stiffness of the structure, are good choices of constraints. But the topology design process will not resolve issues such as stress concentrations.

Multiple global constraints may be specified. If the constraints are in conflict, then a trade-off is done, and a design is selected resulting in the minimum violation of any given constraint.

The global constraint handling considers only active constraints. If none of global constraints is active anymore, then the algorithm will slowly return to the user specified mass fraction.

The algorithm increases the user specified mass fraction by 25% for use as an initial value, because this allows for better convergence. If no constraints are active then the value of the mass fraction will slowly return to the user specified value.

Local effects such as stress concentrations are not handled by this algorithm.

This can be specified in conjunction with dynamic weighting to obtain a more powerful approach. This combination of methods allows one active constraint per load case.

2.11. Designing for Small Mass Fractions

The projected subgradient method has internal logic for the design of small mass fractions. A small mass fraction would be values from 0.1 to 0.005. Initially convergence may appear to be slow; this is because a variable change from 0.01 to 0.02, while doubling the value, is not as visible as a value change from 0.1 to 0.2, which also doubles the value.

One pitfall if not using element deletion is to have the minimum variable value too close to the initial mass fraction – say starting the design process with a mass fraction of 0.01 and an minimum variable value of 0.005. This can have two side effects. The first side effect is that the elements at the minimum variable value of 0.005 counts towards the mass fraction of 0.01, which means they make up half of the mass of the structure. The second side effect can be lack of solidification of the part, because the undeleted elements contains so much mass that they provide structural support. The workarounds here are using element deletion or a smaller minimum element variable value. Additionally, solidification can be forced using the Gradual True Mechanics solid/void scheme at the expensive of having more iterations (about 50 or more iterations).

3. Free Surface Design

Free surface design revises a solid surface shape to have a uniform surface stress for the given loads.

3.1. The Design Surfaces

The surface of a solid part can be redesigned to reduce stress concentrations.

There is no restriction on the element type. The surface is defined using a *SET_SEGMENT definition in the LS-DYNA input deck.

Shells structures cannot be designed in this version of LS-TaSC.

3.2. Geometry and manufacturing definitions

For each surface geometry and manufacturing definitions such as being an extrusion may be specified.

The geometry definitions, as shown in Figure 2-1, are:

- *Symmetry.* For these the geometry is duplicated across a symmetry plane. The part as supplied by the user must be symmetric: an element must have a matching element on the other side of the symmetry plane.
- *Extrusion.* The surface is extruded in a certain direction. The initial surface as supplied by the user must already be an extrusion.
- *Smooth transition.* A smooth transition between the free surface and the surrounding material is achieved by gradually smoothing out the transition between the modified and unmodified surface at a surface edge specified using a node set.

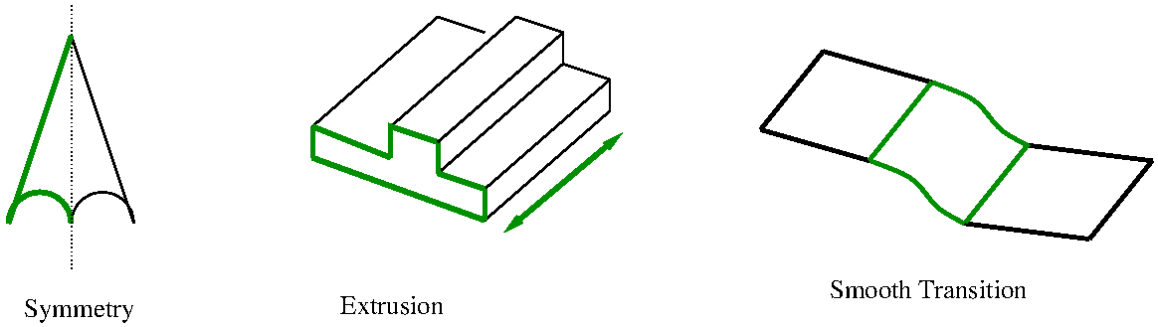

Figure 3-1: Geometry definitions

Multiple geometry constraints can be specified for each part. Some combinations of geometry constraints may however not be possible. The symmetry planes must be orthogonal to each other and the extrusion direction must be on the symmetry planes.

The symmetry and extrusion definitions are implemented using equality constraints, while the smooth transition is imposed scaling the design variables at the nodes considering their distance from the transition.

3.3. Convergence

For shape computations the objective is to have a constant stress over the design surface. The convergence is defined relative to how much of an improvement in the objective was achieved with respect to the initial design. Consider Figure 3-2 showing both the stress range and the integral defining the smoothness of the stress.

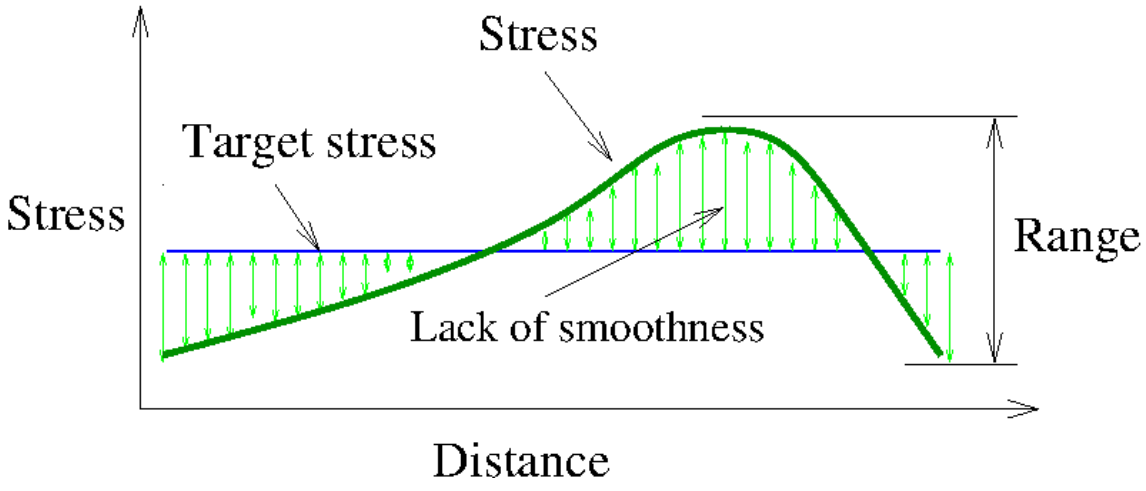

Figure 3-2 Convergence for shape design

Four strategies of setting the target stress are allowed:

- Match average. This is the recommended default which uses the average stress over the surface as the new target stress. This results in the removal of stress concentrations.

- Minimize volume. The maximum value on the surface will be selected. In this case the weight will be reduced.
- Minimize stress. The minimum value on the surface of the surface will be used as the new target. In this case the average stress will be reduced.
- A user-defined value.

The convergence criterion / tolerance reported by the code is a measure of how much the surface stress was smoothed from the initial variation of the stress relative to having an uniform stress: a value of 1 indicates that an uniform stress was achieved, while a value of 0 indicates no improvement.

3.4. Design Variables

A design variable is assigned to every node in the design surface.

3.5. Filtering of Results

A radius based strategy is used to identify neighbors. In this strategy, a virtual sphere (of default or user-defined radius) is placed at the center of an element. All elements that are within this sphere are considered the neighbors of the corresponding element. The result at an element is computed scaled from its own value and of its neighbors.

The default radius is the average element length / $\sqrt{2.1}$ which means a sphere of this radius will include the centroids of all connected elements in a regular quadrilateral mesh.

For dynamic problems, it was observed that accounting for the history of evolution induces stability. Hence, the field variable (internal energy density) of i^{th} cell at iteration t is updated by defining a weighted sum on the field variable of three previous iterations.

3.6. LS-DYNA® Modeling Specifics

3.6.1. Surface Definition

The design surfaces for shape optimization must be defined using *SET_SEGMENT.

3.6.2. Smooth Transition

The transition is defined using node set definitions (*SET_NODE and *SET_NODE_LIST) defining a line on the edge of the surface.

3.6.3. Disallowed Keywords

The *PARAMETER keyword is not supported in the current version. All other keywords are allowed, but LS-TaSC will only edit the nodal locations to reflect changes to the design. The *INCLUDE keyword is supported from version 3.1 onwards.

3.7. Automatic mesh smoothing

The interior nodes of the FE model related to the design surfaces are smoothed by the design algorithm. The mesh is smoothed for a certain depth below the surface. The default value of the remesh depth (defined in the number of elements) should be fine for most problems, but problems with few elements in the depth direction will require this value to be reduced.

4. Program Use

Both topology and shape design consist of describing the topology design problem together with the solution methodology, the scheduling of the automated design, and the evaluation of the results.

4.1. Running the Program

The LS-TaSC GUI is launched from the command prompt by running the executable (*lstasc*). If a project already exists, then the project database name (**.lstasc*) can be supplied in two ways:

1. With the execution command

   $ lstasc myProject.lstasc

2. The *file open* dialogue, available from the File pulldown menu

LS-TaSC can be run without the GUI from the command line using the command `lstasc_script myDataBaseFile.lstasc` or as `lstasc_script myScriptFile` with the script commands as described in the scripting manual.

4.2. Opening and Saving Projects

The standard File pulldown is provides the ability to open and save projects. The name of the database can also be specified on the command line when staring the GUI as *lstasc lst_project.lstasc*.

4.3. Problem Definition

The topology design problem is defined by (i) the allowable geometric domain, (ii) how the part will be used, and (iii) properties of the part such as manufacturing constraints. Additionally, you have to specify methodology requirements such as termination criteria and management of the LS-DYNA® evaluations. In the GUI, provide this information using the following headings:

- *Cases.* These store the load case data such as, the LS-DYNA® input deck and executable to use. The *Cases* data therefore contain the information on how to simulate the use of the part.

- *Parts.* The properties of the parts such as the part ID, mass reduction, and geometric definitions are given here. This is only required for topology optimization.
- *Surfaces.* The properties of the surfaces such as the segment set ID and geometric definitions are given here. This is only required for free surface design.
- *Constraints.* This optional information prescribes the stiffness or compliance of the whole structure.
- *Completion.* These are methodology data such as the convergence criterions.

4.4. LS-DYNA® Simulation

The modified input deck is analyzed using LS-DYNA®. One can take advantage of multiple processors using the MPP version of LS-DYNA® by specifying the simulation options as part of the command. Queuing system can also be used as described in Section 4.5.2.

If you desire to use less disc space, then the options are to reduce the LS-Dyna output or to create a file named “clean” (“clean.bat” in Windows) in the directory containing the database. This “clean” file must be set to be executable and can contain lines such as “rm –rf d3hsp scr00*”. LS-TaSC will execute this “clean” script in every directory where LS-DYNA ran successfully. You can also use the advanced options capability (see section 4.6.1) to read results from the *d3part* database instead.

4.5. Setting up the Problem

The GUI consists of a number of panels. Complete the panels from top to bottom as described in the following subsections.

4.5.1. The Toplevel GUI

The toplevel GUI contains the LS-TaSC tool as shown in Figure 4-1. The toolbar associated with the LS-TaSC tool is also shown. The feature tree contains all the items in the currently open LS-TaSC database such as parts.

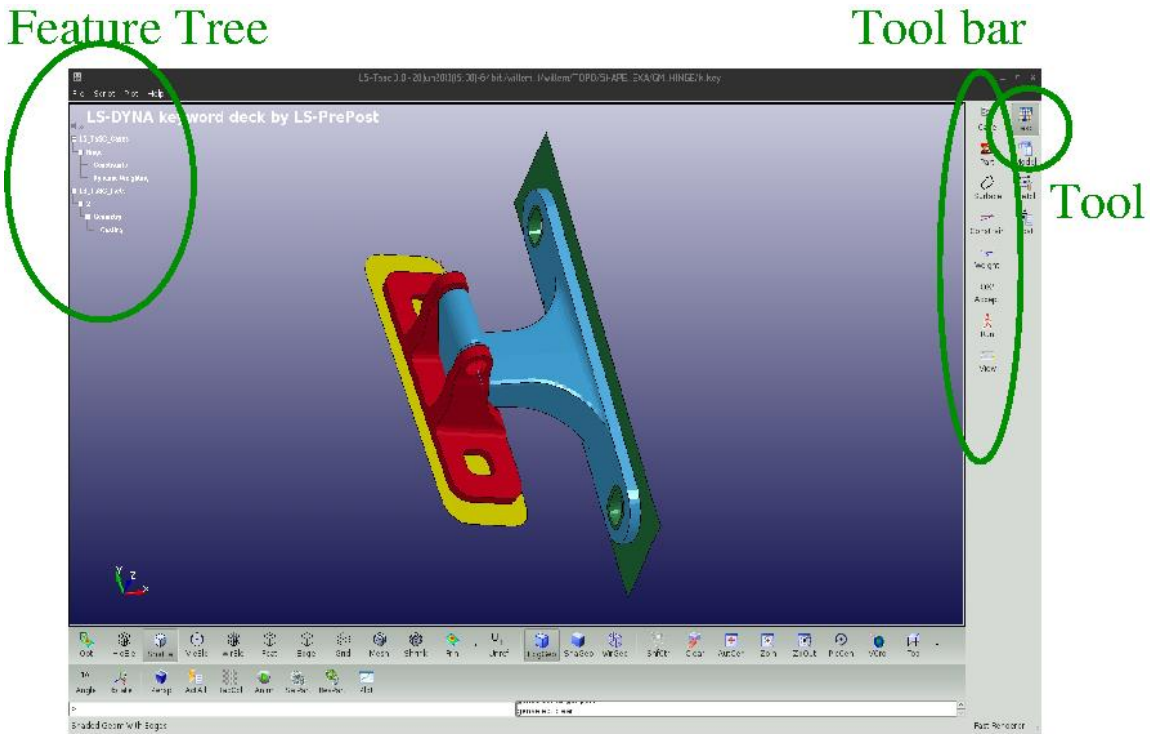

Figure 4-1: The toplevel GUI

4.5.2. The Cases Panel

The cases panel contains all of the load cases to be analyzed using LS-DYNA®. See the following table and Figure 4-2 for more details.

Table 4-1: Cases Data

Cases data	
`Name`	Each case is identified with a unique name e.g., TRUCK. The same name would be used to create a directory to store all simulation data.
`Execution Command`	The complete solver command or script (e.g., complete path of LS-DYNA executable) is specified.
`Input File`	The LS-DYNA input deck path is provided.
`Weight`	The weight associated with a case is defined here. This enables the user to specify non-uniform importance while running multiple cases.
`Number of`	This parameter indicates the number of LS-DYNA evaluations to be run simultaneously. This parameter only applies if

`jobs`	multiple cases must be evaluated or if the multipoint method is being used. Effectively only the value for the first case needs to be set, because the value used is the maximum found over all the cases. Note that the number is not per case, but is the overall number of all LS-DYNA evaluations to be run for all the cases together.
Queue system	This parameter is used to indicate the queuing system. The options are: lsf, loadleveler, pbs, nqs, user, aqs, slurm, blackbox, msccp, pbspro, Honda. By default, no queuing system would be used. See the appendix for a description of setting up the queuing systems. The system is the same as used in LS-OPT®, so a queuing system definition is the same.

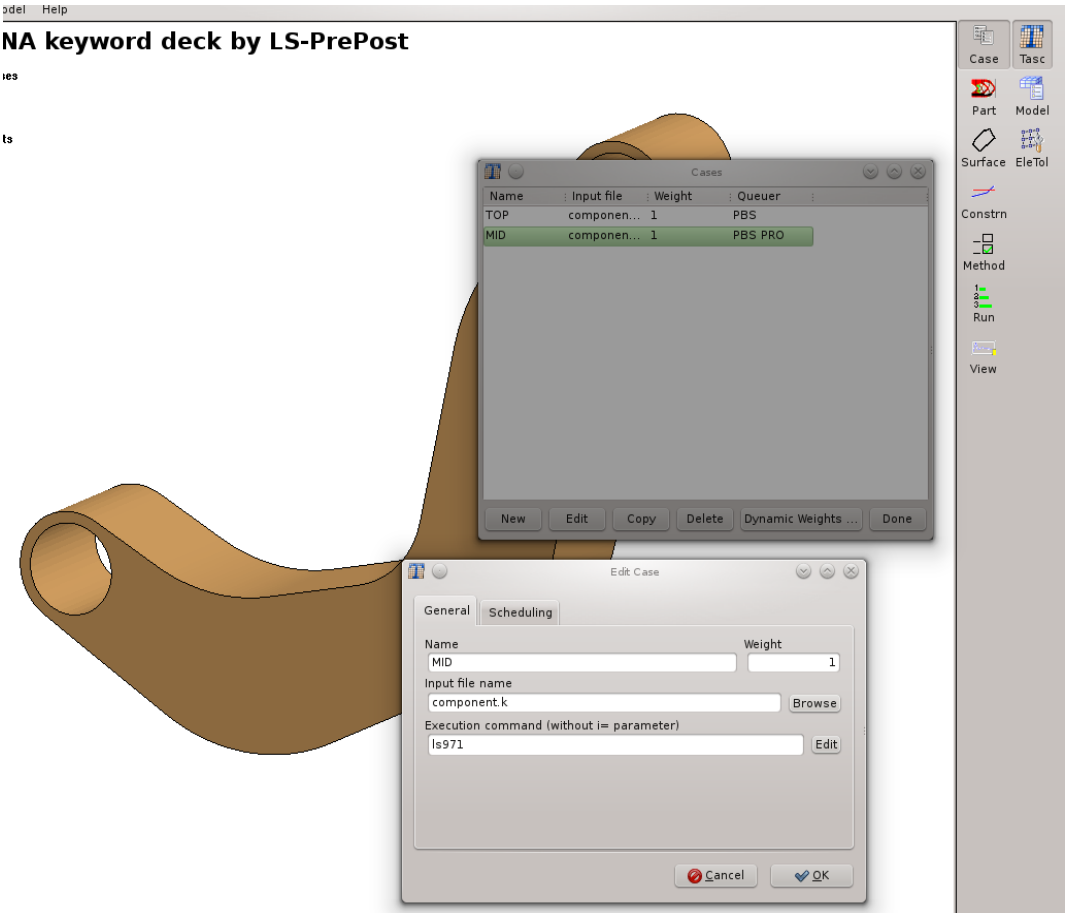

Figure 4-2: The cases panel.

4.5.3. The Dynamic Weighing Panel

The dynamic weighing panel is used to specify the equation defining the dynamic weighing. See the following table and Figure 4-3 for more details. The load case weights will be adjusted to satisfy this equation.

Table 4-2: Dynamic Weighing data

Dynamic weighing data	
`Activate dynamic weights`	If checked, then dynamic weighing will be used.
`Numerical values`	The constraint values can be scaled using these values. Note that the GUI is laid out to specify an equation. Specify these values to complete the dynamic weighing equation.
`Constraint`	A constraint for each load case must be selected in order to specify the dynamic weighing equation.

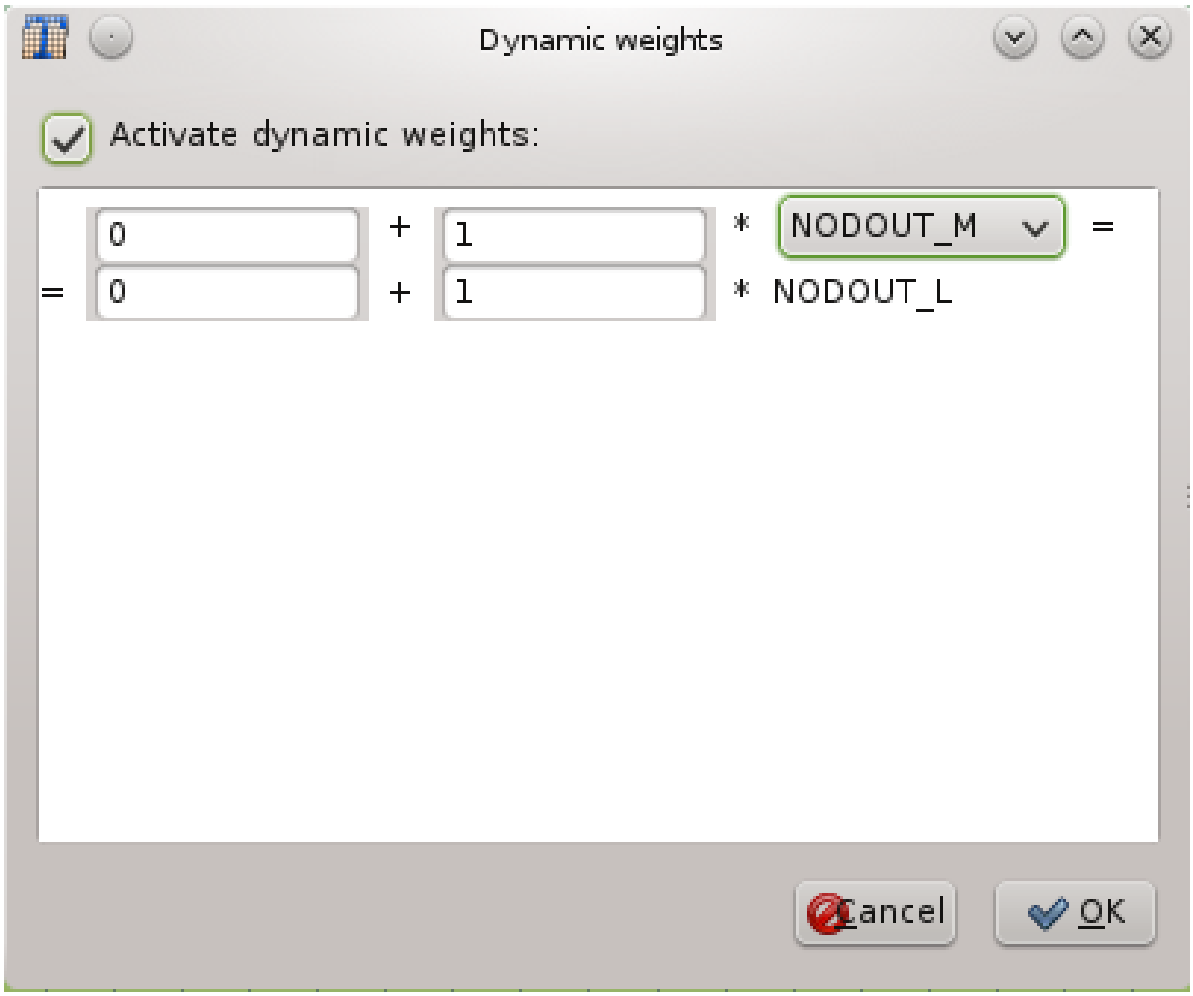

Figure 4-3: The dynamic weighing panel.

4.5.4. The Parts Panel

The part definition panel contains information about the parts to be designed, such as the geometry and mass fraction. See the following table, Figure 4-4, Figure 4-5, and the chapter on topology optimization for more details.

Table 4-3: Part data

Part data	
`Design part ID`	The user needs to specify the design domain for topology optimization. To facilitate the identification of design domain, all elements in the design domain are put in a single part in the LS-DYNA input deck. The information about the design domain is then communicated through the corresponding *part-id*.

	Note: For multiple load cases, the user must ensure that the design domain mesh and the *part-id* remain the same in all input decks.
`Mass fraction`	This parameter describes the fraction of the mass of the part to be retained. The rest will be removed. A part with an initial weight of 5, designed using a *Mass Fraction* of 0.3 will have a final weight of 1.5. A negative value is used to specify the actual desired mass of a part; so, using the previous example, a value of -1.5 will result in a part with a mass of 1.5 and mass fraction of 0.3 relative to an initial mass of 5.0.
`Neighbor radius`	All elements within a sphere of radius of this value are considered the neighbors of an element. The design variable at an element is updated using the result at the element averaged together with that of its neighbors. Smaller values of this parameter yield finer-grained structures. If the value is negative then the value is assumed to be element specific and the radius used for an element is the absolute value of the specified value times twice the average distance from the center of the element to the nodes. If the value is positive then the specified value is applied to all elements. The default value is -1.0, which means the results from all elements sharing a node with an element are likely to be used.
`Minimum variable fraction`	If the design variable value associated with and elements is too small then that element is deleted to preserve the stability of the model. An appropriate value ($0.05 < x < 0.95$) is supplied here. The default is 0.05 for non-linear problems and 0.001 for linear problems.

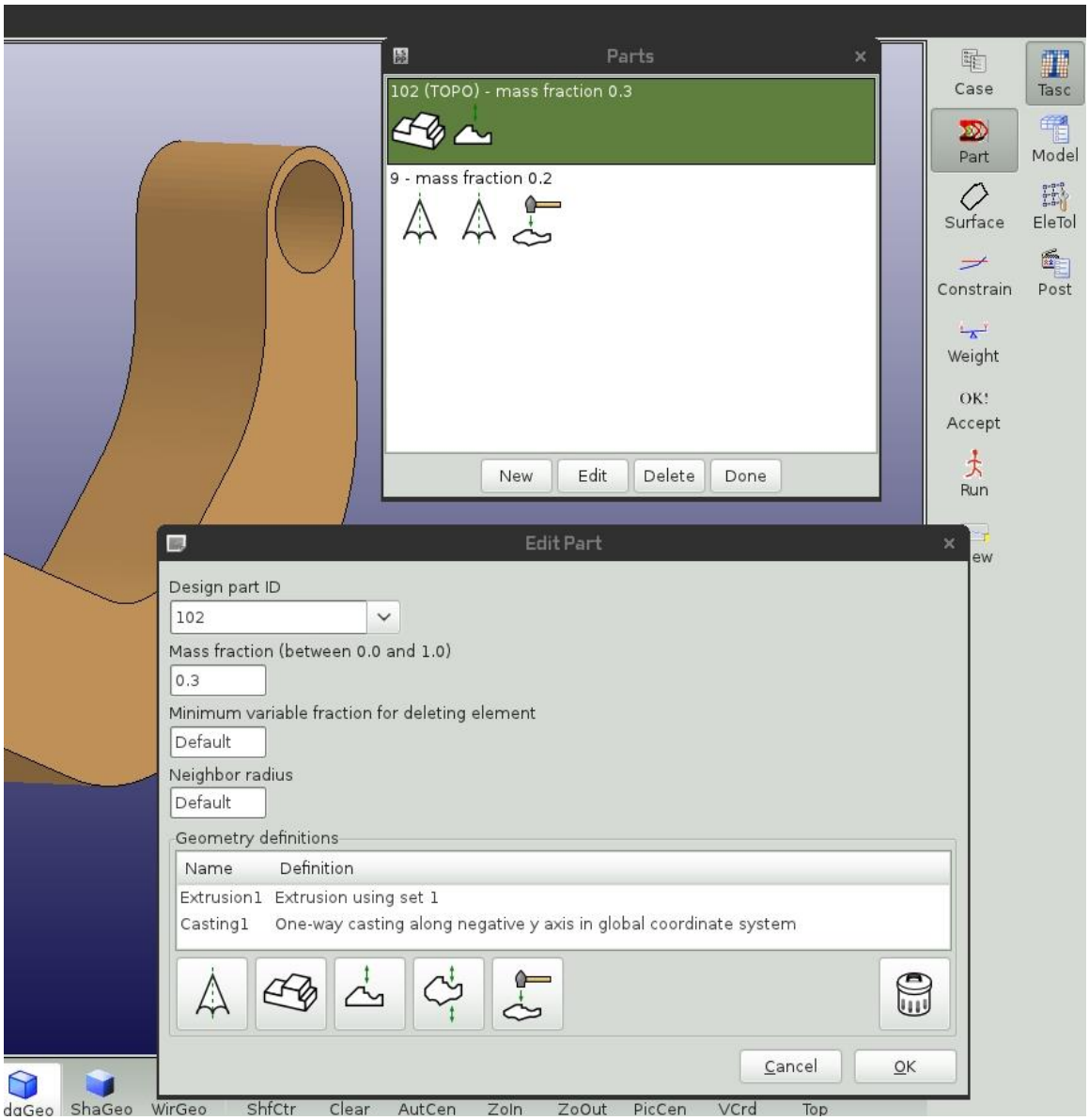

Figure 4-4: The parts panel.

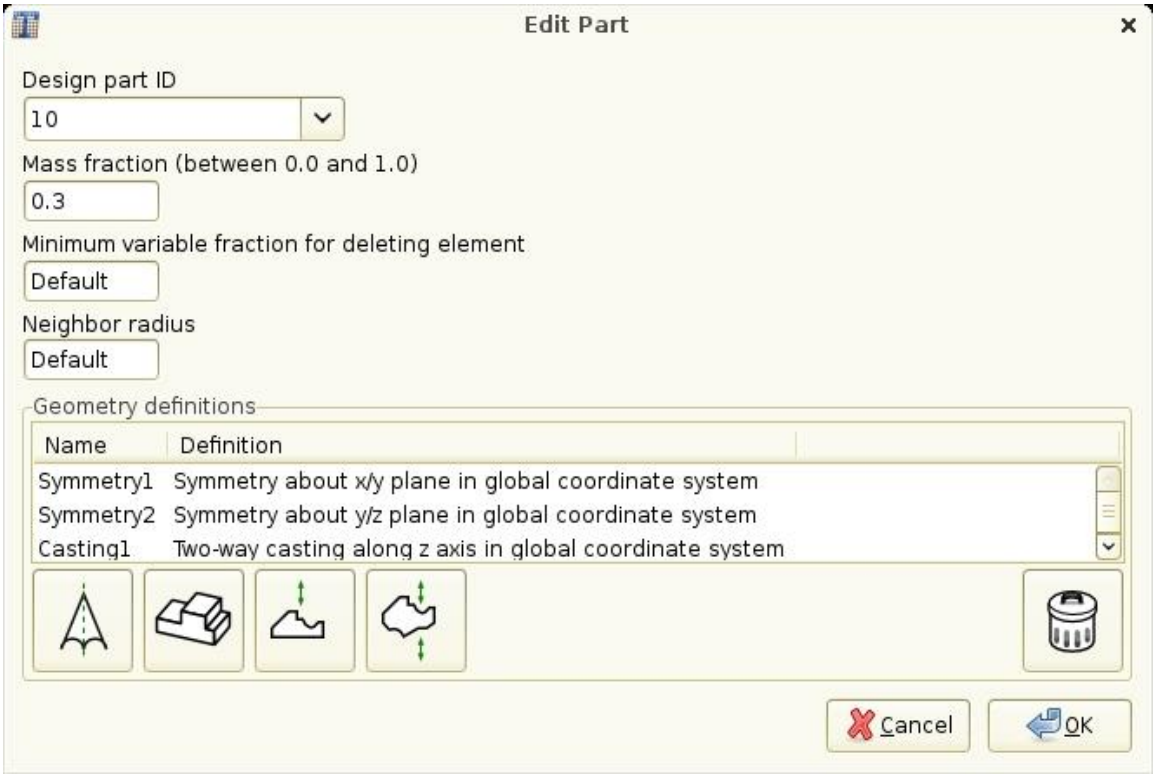

Figure 4-5: The panel to create part and geometry.

4.5.5. The Surface Panel

The shape definition panel contains information about the surfaces to be designed, such as the geometry. It is similar in function and layout to the parts panel described in the previous section. See the following table and Figure 4-6 for more details.

Table 4-4: Surface data

Surface data	
`Segment ID`	The ID of the solid surface that must redesigned.
`Objective`	The objective for the redesign of the surface. One of "Match average" will smooth out the surface stress by considering the average stress over the surface. "Minimum stress" will use the minimum stress on the surface as a target. "Minimum volume" will use the maximum stress on the surface as a target. For Match target the target value must be specified.
`Target value`	If the objective is set to a target value, then the target

	value must be specified using this parameter. Otherwise this value will be ignored.
`Neighbor radius`	All nodes within a sphere of radius of this value are considered the neighbors of a node. The design variable at a node is updated using the result at the node averaged together with that of its neighbors. The default radius is the average element length / $\sqrt{2.1}$ which means a sphere of this radius will include the centroids of all connected elements in a regular quadrilateral mesh.
`Move limit`	This is the maximum distance a node will be moved per iteration. The default is one fifth of the average element length.
`Remesh depth`	This is the number of elements that should be considered in re-meshing after a shape change was done. The default is four elements. The length of the average element is used over the whole part.

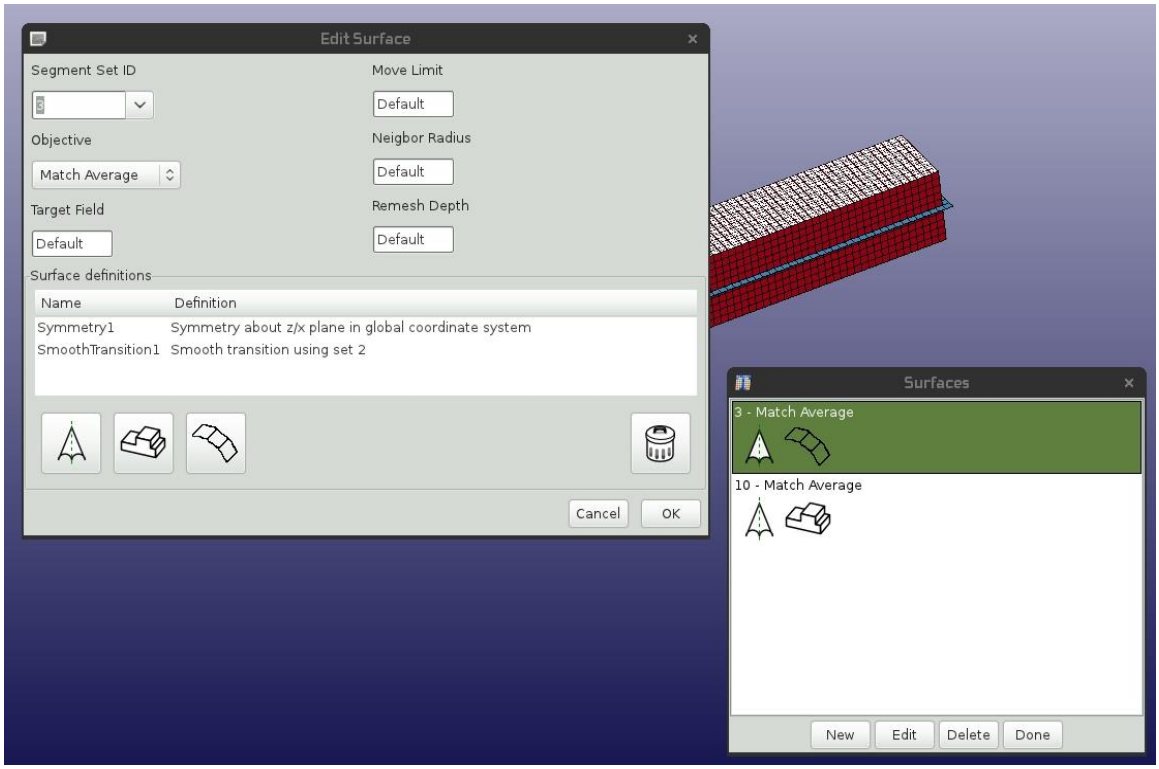

Figure 4-6: The surface panel.

4.5.6. Part and Surface Geometry

The geometric properties can be defined for every part and surface. See the following table and Figure 4-7 for more details.

Table 4-5: Geometry data

Geometry data	
Name	The geometric property can assigned a name or a default name can be used.
Extrusion set ID	To define an extruded part, the user firstly creates a set of all elements that would be extruded. Allowable set definitions are *SET_SOLID, *SET_SOLID_LIST, *SET_SHELL, and *SET_SHELL_LIST.
Symmetry Plane	Specify a symmetry plane to define symmetry.
Cast direction	A cast direction is required for a casting or forging constraint. The direction can be negative. This is the direction in which the material will be removed. It is the opposite of the direction in which a casting die will be removed.
Coordinate system ID	The geometric property can be defined in a specific coordinate system or the default Cartesian system can be used.
Minimum thickness	A minimum thickness can be specified for both the forging and no interior holes geometry definitions. A forging geometry definition will always result in a minimum material thickness in the forging direction. The no interior holes definition will preserve features smaller than the given thickness.
Smooth transition Set ID	To define a smooth transition for a surface, the user firstly creates a node set definition defining the edge. Allowable set definitions are *SET_NODE and *SET_NODE_LIST.
Smooth transition width	The smoothing of a surface transition will be done over this specific width. At the edge specified by the node set, there will be no shape change; but the shape change will be fully effective at the smooth transition width away.

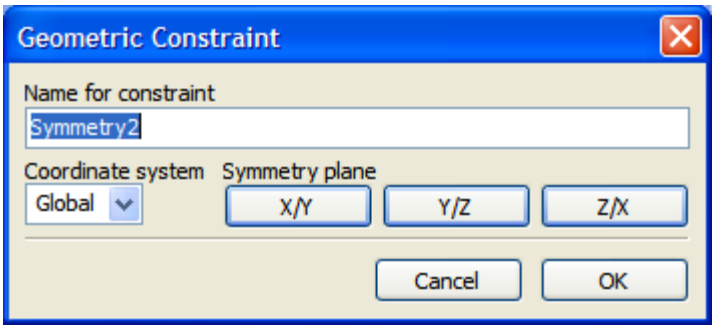

Figure 4-7: Creating a geometry constraint. A symmetry constraint is shown.

4.5.7. The Constraints Panel

The constraint panel contains the global constraints on the structure. See the following table and Figure 4-8 for more details.

Table 4-6: Constraint Data

Constraint data	
`Name`	Each constraint is identified with a unique name e.g., MAX_DISP.
`Case`	Each constraint is associated with a load case.
`Constraint type`	The important ones are NODOUT (stiffness), RCFORC (compliance), or EXPRESSIONS (see text), GLOBAL variables.
`Lower and upper bound`	The lower and upper bound are respectively the minimum and maximum values allowed for that the constraint. Leave this field empty for having no bound.
`ID`	This is the ID of the node in the FE model at which the results must be collected.
`Select`	This parameter indicates which value over time must be selected. It can be the last value, the maximum value, the minimum value, or at a specific time. A time, or a time interval can also be specified.
`Filtering`	If filtering is desired, select the type of filter, frequency, and time units. LS-PREPOST can be used to investigate the effects of filtering.

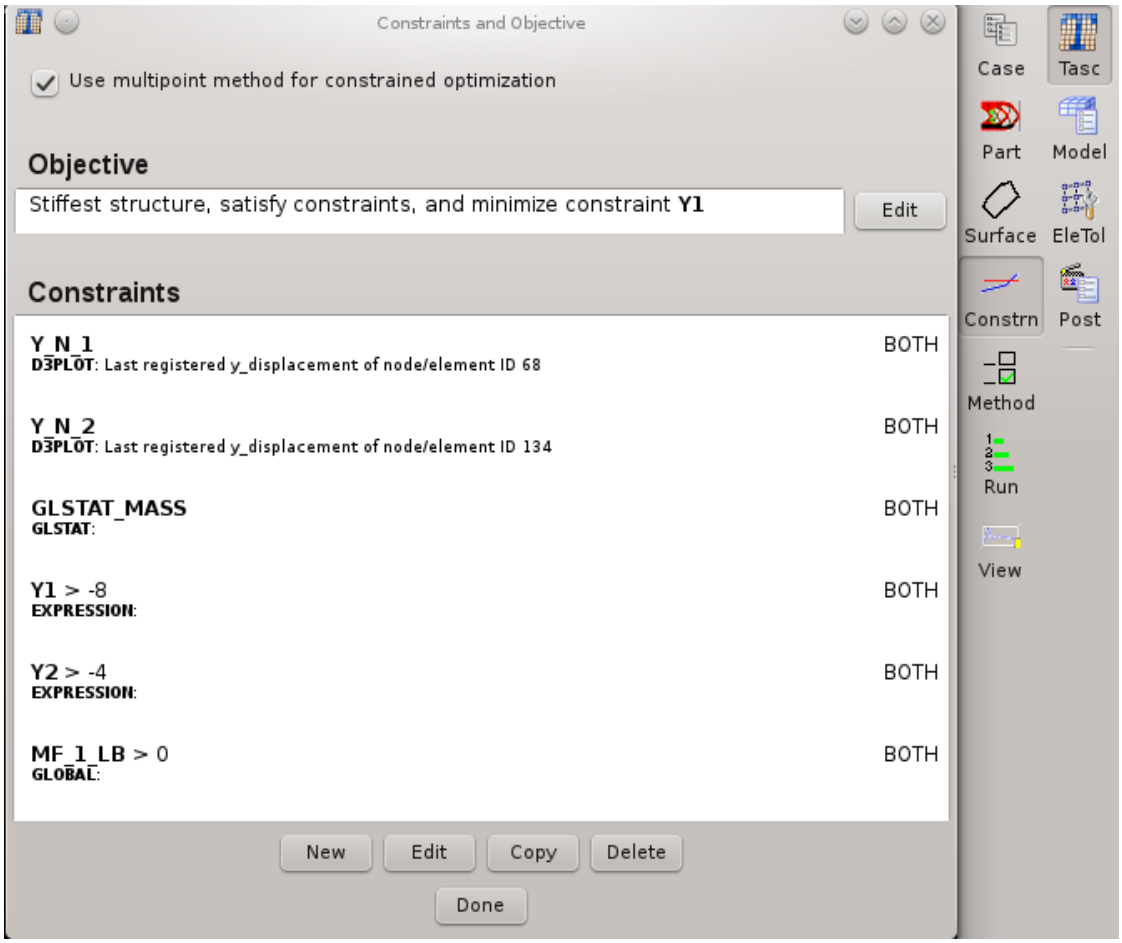

Figure 4-8: The constraints overview panel.

The EXPRESSIONS option allows you specify a mathematical computation with the other constraint values. Say you have responses defined using the names NOD1X and NOD44Y, you can define an expression using the values of the other constraints with the string “(NOD1X + NOD44Y)/2.0”.

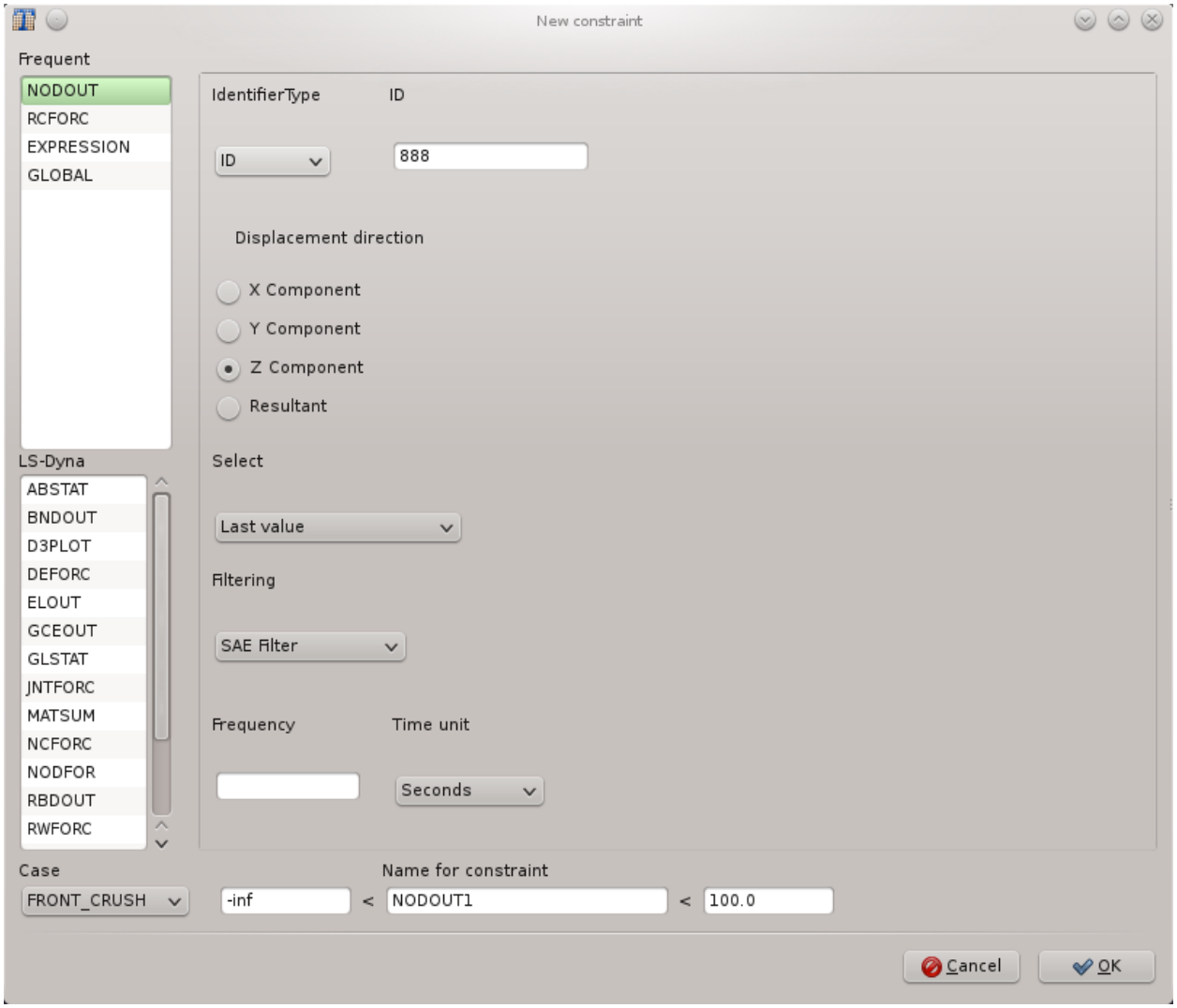

Figure 4-9: The constraints creation panel.

4.5.8. The Objective Panel

The objective panel specifies how the optimization problem will be solved. See the following table and Figure 4-10 for more details. The default for the multi-point optimization methods is to minimize the mass of the structure. The Fixed volume option is only valid for the older methods which will design for the fixed volume (user specified mass fraction) if no constraints are active. The Fixed volume option is also the default and only option for the older methods.

Table 4-7: Objective data

Objective data

`Objective`	This is the objective to be used. One of Default, Fixed volume, Minimum Mass, and Constraint.
`Constraint`	If the objective type is set to Constraint, then a constraint must be selected. The value of this constraint will be minimized.

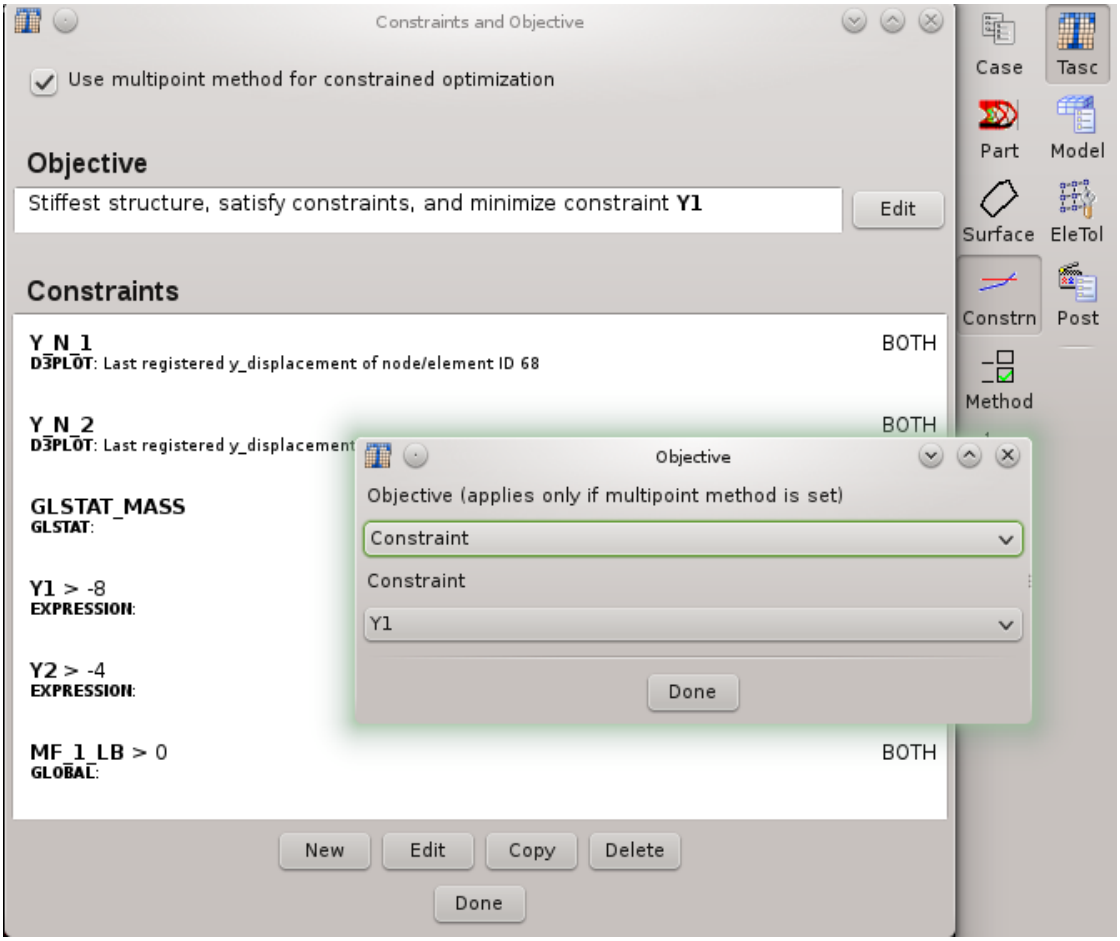

Figure 4-10: The objective panel.

4.6. Setting the Design Methodology

4.6.1. Algorithm and Convergence

The completion page in the method panel specifies how the termination criteria for the topology design. See the following table and Figure 4-11 for more details.

Table 4-8: Completion data

Completion data	
Number of design iterations	This is the maximum number of iterations allowed. The default value is 30.
Convergence Tolerance	For topology design the minimum mass redistribution is used to stop the search when the topology has evolved sufficiently. This convergence tolerance is compared with the *Mass_Redistribution* history variable displayed in the view panel. The default value is 0.002. For surface design the convergence tolerance indicates how much the surface stress was smoothed: a value of 1 indicates that a uniform surface stress was achieved, while a value of 0 indicates no improvement. The value is relative to the initial variation of the stress surface stress – so a value of 0.5 will indicate a 50% reduction of the initial surface stress variation.
Solidification	This is the convergence tolerance for the projected subgradient method. It measures which fraction of the structure is assigned to be either solid or void. The maximum value of 1 means that all of the variable values have been determined.
Design algorithm	This can be either the optimality criteria method (the only method in the previous releases) or the projected subgradient descent method (new to this release). Eigenvalue problems requires the use of the project subgradient method.
Topology variables move limit	This option provides a step size for the optimality criteria algorithm. The value can be reduced to prevent oscillations and improve convergence at the cost of having more iterations.
Normalize Case Data	This normalizes the cases data. This means that each cases will have an equal effect on the design – i.e. a 100N force load cases will have the same effect as a 1 N force load case. The load cases weights can still be used to differentiate the effect of the load cases. This option is only available for the Optimality Criteria algorithm. The Projected Subgradient algorithm always normalizes the gradient.

Desired mass flow	This is the optimization step size for the projected subgradient algorithm. The value can be reduced to prevent oscillations and improve convergence at the cost of having more iterations. Type in "Default" to obtain the default, recommended settings. Valid values include "0.5*Default", "0.1", "0.2*(0.8+0.2*exp(-lst_Iteration/20.))*lst_FreeVolFrac".
Descent acceleration factor	Acceleration factor or momentum for the projected subgradient method.
Objective Convergence Tolerance	A termination criteria when the multipoint constrained optimization scheme is used. Optimization is terminated if the change in objective is less than this value. It is computed as $\|O_{new} - O_{old}\|/\|O_{new}\|$.

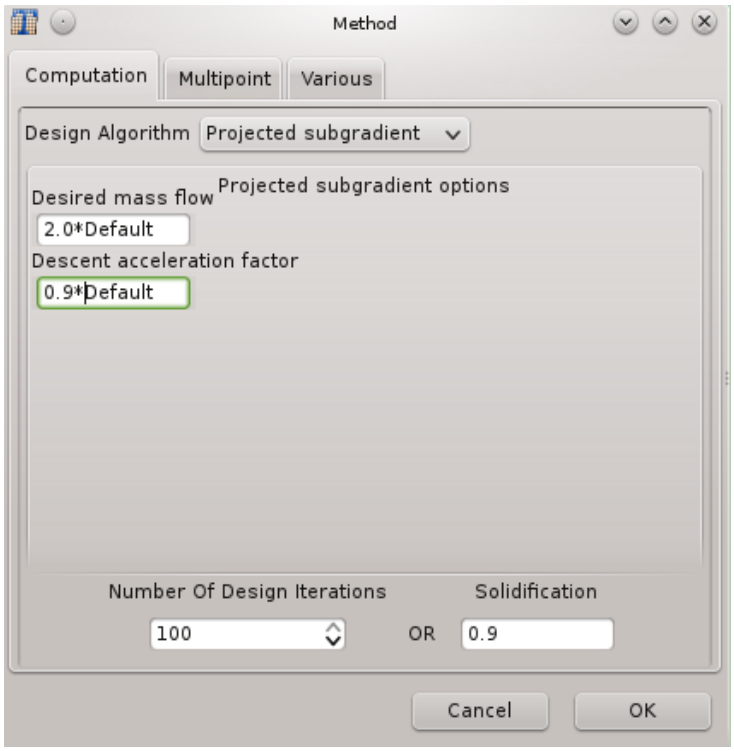

Figure 4-11: The completion options in the method panel

4.6.2. Multipoint options

The multi-point options can be set as shown in Figure 4-12. This is accessed through the method panel.

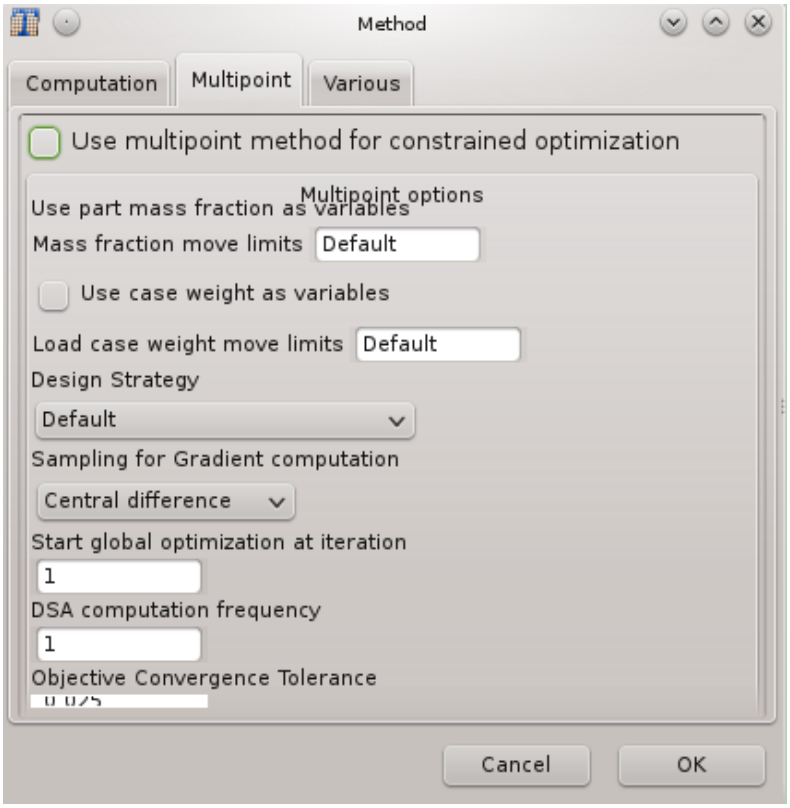

Figure 4-12 The multipoint options in the method panel

The available options are described in the following table.

Table 4-9: Multipoint options

Option	Description
Use multipoint method	For constraint optimization, this will results in the evaluation of several candidate designs per iteration. The candidate designs are computed considering different values of the mass fractions and load case weights.
Use case weights as variables	The load case weights can be used as global variables. This a powerful option to satisfy constraints across load cases.
Design strategy	This is the design strategy to be used. The standard option is the classical optimization (dynamic programming) using the global variables. Dynamic weighing or a random point selection scheme can also be selected.

Sampling	The gradients in the multipoint methods can be computed using finite differences, a linear response surface, or the best results can be selected from a random set. The cheapest option is forward differences together with a large value of the DSA frequency. Central differences is the most accurate. Monitor the derivatives in the history panel to see if the accuracy is sufficient.
Number of RSM points	The number of points to be used to construct either the linear response surface or in the random search. A value of -1 will results in a default value being used.
Starting iteration	The iteration in which optimization using the global variables will commence. Iterations before this will only contain standard topology optimization with the initial values of the global design variables.
DSA frequency	This is how often the multipoint method should evaluate the design sensitivity information numerically using the multipoint method. Monitor the derivatives in the history panel to see if the accuracy is sufficient.
Mass fraction move limit	Mass fraction move limits relative to the current value. This can be a number or an expression, e.g. *0.4* or *0.2*lst_Iteration*. For example, if this value is 0.2 and the mass fraction is 0.1, then the move limits on the mass fraction are [0.08,0.12]. The system variable *lst_Iteration*, the design iteration, can be used in the computations. The default is "0.05+0.05*exp(-(lst_Iteration-1)/10.)" for the mass fractions, which means that the move limit is decreased as the iteration count (lst_Iteration) increases.
Load case weight move limit	Load case weights move limits relative to the current value. This can be a number or an expression, e.g. *0.4* or *0.2*lst_Iteration*. For example, if this value is 0.2 and the weight is 0.1, then the move limits on the weight are [0.08,0.12]. The system variable *lst_Iteration*, the design iteration, can be used in the computations. The default is "0.1+0.1*exp(-(lst_Iteration-1)/10.)" for the load case weights, which means that the move limit is decreased as the iteration count (lst_Iteration) increases.

4.6.3. Miscellaneous Options

Other methodology options can be set as shown in Figure 4-13. This is accessed through the method panel.

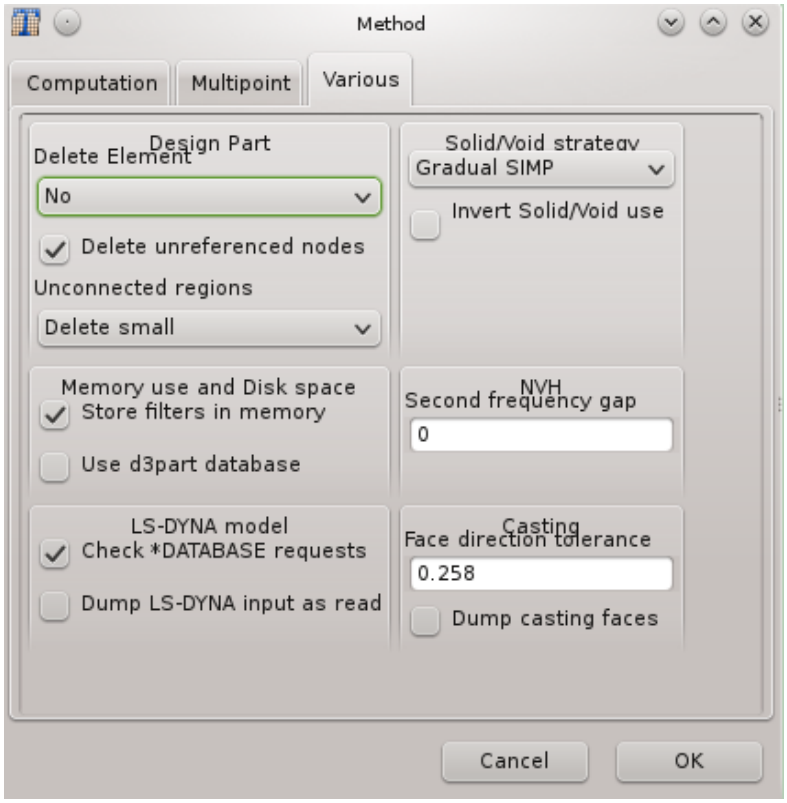

Figure 4-13 The various options in the method panel

The available options are described in the following table.

Table 4-10: Various options

Option	Description
Delete elements	Normally the program delete elements below a certain variable value, but the elements can be set to have a value of the minimum allowable. A third option is to delete the elements only for the explicit dynamics loads cases and to keep the elements at the minimum value for the implicit cases (including statics and eigenvalue problems).
Delete unreferenced nodes	The MPP LS-DYNA execution speed can be slowed down in later iterations because of the presence of many unreferenced nodes. Use this option to correct this. This option will delete unreferenced nodes in the interior of the

	design part. Note that a check is only done whether the design part still use these nodes; if these interior nodes are referenced by other FE parts or entities, then the LS-DYNA run will fail due to the absence of these nodes.
Unconnected region	Deleted, warn about, or ignore unconnected regions in the model. Unconnected means that the (group of) element(s) does not share a face with the main body. Elements sharing node(s) but not a face are unconnected. The default is to delete small (less than 10% of the main body) regions only.
Store filters in memory	This option can reduce memory use by a factor two, but extend the time required to extract results. The option is useful can cases where the elements have many neighbors such as tetrahedral models.
Use d3part database	The field results (IED values) will be read from the d3part database instead of the d3plot database. Use this option to save disk space.
Face direction tolerance	For casting definitions this is used to decide whether two elements face in the same direction. It is the sine of the allowable angle.
Check database requests	If constraints are defined, then a check is done to see that the FE deck include the required *DATABASE entries. This can be disabled using this option.
Dump casting faces	This advanced options dumps files showing the casting faces to an ASCII file. This can be viewed in LS-PrePost using the Fringe toolbar (select the User option).
Dump LS-Dyna input as read	This is a debugging option for support to investigate input deck reading.
Solid/void strategy	The “true mechanics” is recommended as well as the default. The gradual “true mechanics” ramps up the solid/void separation over several iterations – it is recommend where solidification problems are experienced using the standard approach. SIMP, the academic standard, is only recommended for linear problems, because it is prone to causing difficulties such as collapsed elements or incorrect energies during the initial iterations. The gradual SIMP approach, more commonly known as SIMP with continuation, is recommended for mixing explicit and eigenvalue

	analyses, this method gradually increases the p value from 1 to 3 – this helps to algorithm to be more correct prediction energy absorption or displacements during earlier iterations.
Invert solid/void use	The normal solid/void use can be inverted such that it is not used for solids, but used for shells.

4.7. The Run Panel

The run panel is used to submit the design problem for computing. In addition, the LS-DYNA® jobs can also be stopped, and old results deleted. Use this panel and the Viewer panel to monitor job execution. See Figure 4-14 for more details. Note that you can click on an entry in the PID column to see the LS-Dyna output for that specific job.

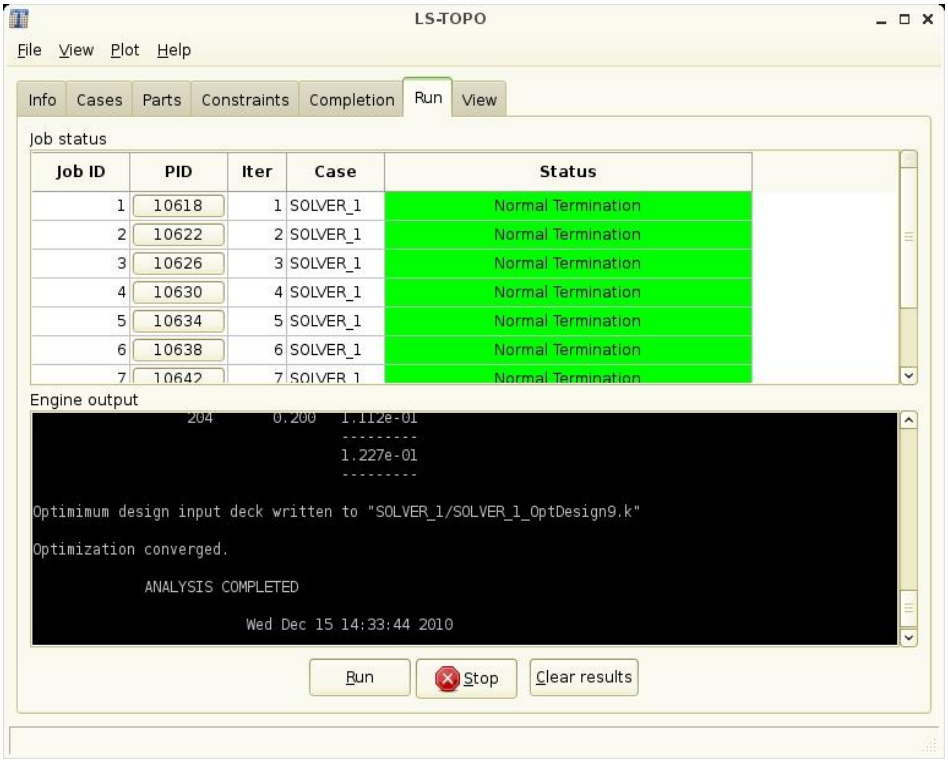

Figure 4-14: The run panel.

4.8. Viewing Results

The view panel can be used to monitor both optimization progress and optimization results. Both histories and plots in LS-PREPOST are possible. See Figure 4-15 and Figure 4-16 for more details.

For the histories note that:

- Multiple histories can be plotted simultaneously by holding down the Control key.
- The plot ranges can be set under the View pulldown menu.
- The histories can be printed or saved to file using the Plot pulldown menu.
- The history data can be exported and postprocessed using the scripting interface.

The available history variables are given in the following table.

Table 4-11: Histories

Histories	
Case/Constraint	This is the value of the Constraint of the named Case.
Case/Weight	This is the weighing applied to the named load Case. If dynamic load cases weighing is set then this value is changed to that effect.
Mass_Redistribution	This convergence criterion is the fraction of the total mass of the structure that has been redistributed per iteration.
Solidification (All Parts)	The fraction of the structure that has been classified as either solid or void. This is the termination criteria for the projected subgradient method. The value will be between 0 (no convergence) and 1 (full convergence).
P123/ElFrac	This is the element fraction for part 123. This value, only relevant for solids, is the fraction of elements in use (not deleted). At convergence this will be close to the mass fraction value (for solids).
P123/MassFrac	This is the mass fraction for part 123. This value is constant if no constraint bounds were set. If constraint bounds were set, then the part mass fraction will be adjusted to satisfy the constraints.
P123/Solidification	The fraction of the part that has been classified as either solid or void. This is only available for the projected subgradient method. The value will be

	between 0 (no convergence) and 1 (full convergence).
*Surface*_Field_Setpoint	This is the set point for the surface results.
*Surface*_Field_Smoothing	This is the smoothing of the surface results.
*Surface*_Field_Max	This is the maximum value of the surface results.
*Surface*_Field_Min	This is the minimum value of the surface results.
*Surface*_Field_StdDev	This is the standard deviation of the surface results.
*Surface*_Field_RangeReduction	This is the range reduction (difference between the maximum and minimum) of the surface results.

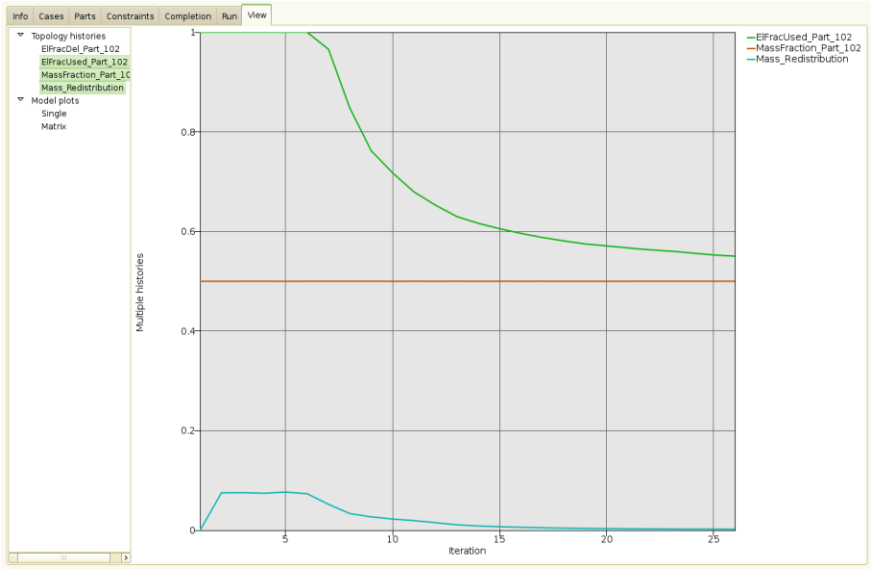

Figure 4-15: The view panel with histories.

For the LS-PrePost plots you can plot either the design of a single iteration or a matrix plot showing the evolution of the design over several iterations. The available field variables are giving in the following table.

Table 4-12:

Field	
`Variable Fraction`	The value of the *design variable* for the element.

`Material utilization`	The extent to which the material in the element is used in the application. These are the values actually used in the redesign and consider multiple load cases and geometry definitions such symmetry. The value is high for parts of the structure heavily used and low for structural elements not useful in the application. This information is only available after the design has been analyzed using LS-Dyna.
`Solid density`	The *material density* in a solid element. This is related to the *Variable Fraction* field.
`Solid IED`	The *Internal Energy Density* for solid elements. This is related to the material utilization.
`Shell IED`	The *Internal Energy Density* for shell elements. This is related to the material utilization.
`Shell thickness`	The *shell thicknesses*. This is related to the *Variable Fraction* field.
`Projected subgradient`	The *gradient* of the load case objective as used in the computation. The gradient is projected such that it does not change the mass of the structure. Negative values means material will be removed, while positive values means mass will be added.
`Contributing case`	This is the index of the load case(s) contributing material at that location. The plot in effect shows the load paths being created at that iteration. The values must be interpreted in hexadecimal. A value of 1 (0x0001) means the first case is responsible for the redesign of that point, a value of 2 (0x0010) means load case 2 is responsible, while 3 (0x0011) means that both load cases needs more material at that location. So a value of 11 (0x1011) will mean that load cases 1, 2 and 4 need material at that location, but not load case 3.

Value	Hex	Index of load case contributing the load path / material
0	0x0000	None
1	0x0001	1

2	0x0010	2
3	0x0011	1,2
4	0x0100	3
5	0x0101	1,3
6	0x0110	2,3
7	0x0111	1,2,3

This plot is not available if a constraint on a frequency is specified.

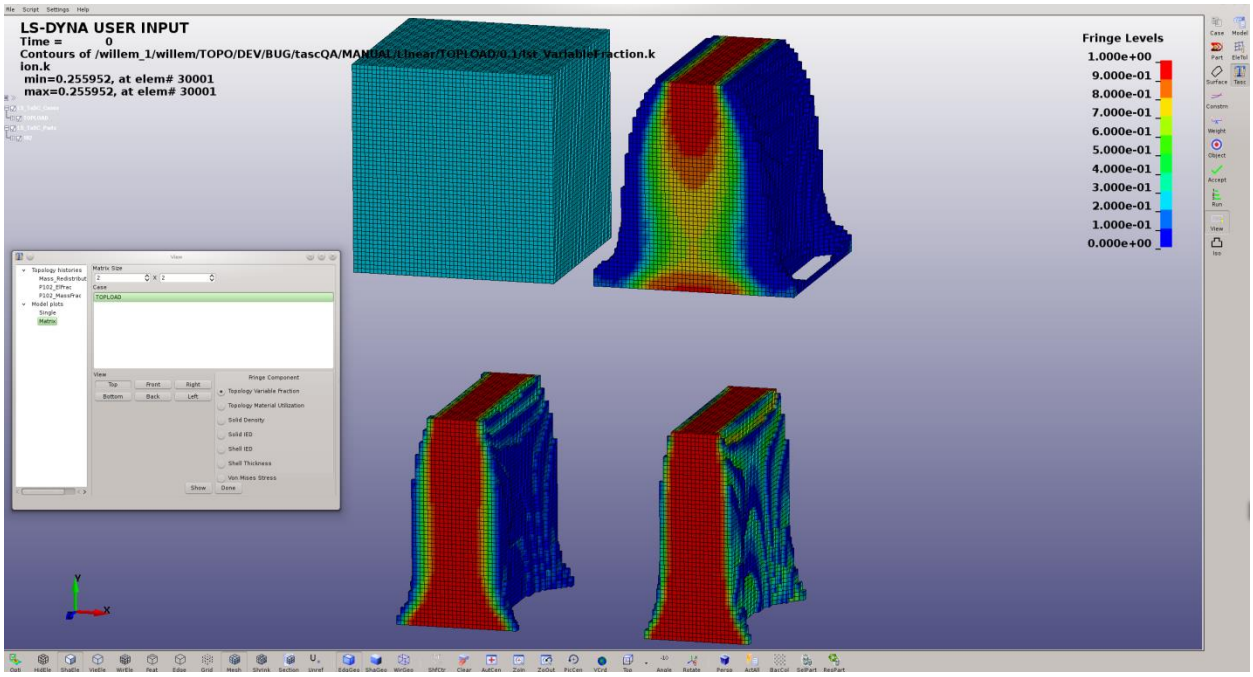

Figure 4-16: Viewing the model evolution in LS-PREPOST.

4.9. Iso-surface plots of a design

You can create surface containing the final optimal structure. The iso-surface is created at a specific value of the design variables. You can select both the value of the design variables and a file to which to save the data. The data is saved in the LS-Dyna input format. This is available only for the topology design of solid elements in the current version. The results are averaged at the nodes using the average of all the values of the elements attached at the node.

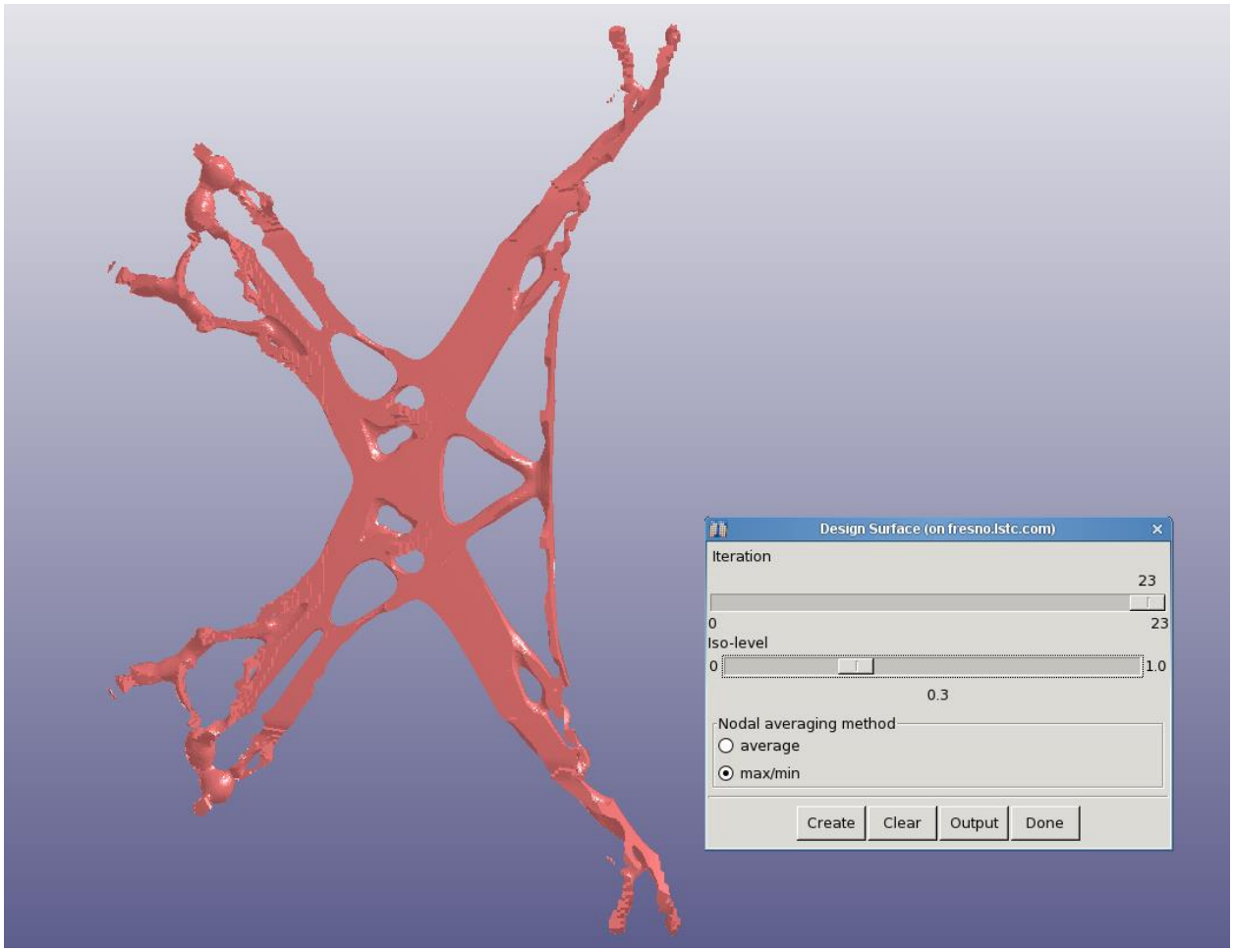

Figure 4-17: Creating an iso-surface plot of a design. The design shown is that of the interior of a bonnet.

4.10. Databases and Files

The important files and directories are shown in the figure below. Four files are important to know about:

- The project database
- The project results in the *lst.binout* binary file
- The optimal design in the case directory
- The d3plot files in the run directory inside the case directory

4.10.1. The Optimal Design file

The file containing the optimal design is named OptDesign<*iteration_no*>.k and can be found in the case directory. This is basically the original LS-Dyna input deck without the deleted elements (the elements associated with the unused material). The file has various uses; for example, to export the design, or it can be remeshed to be used as a new starting design, or in a mesh refinement strategy, or in a strategy reducing the part mass fraction in stages.

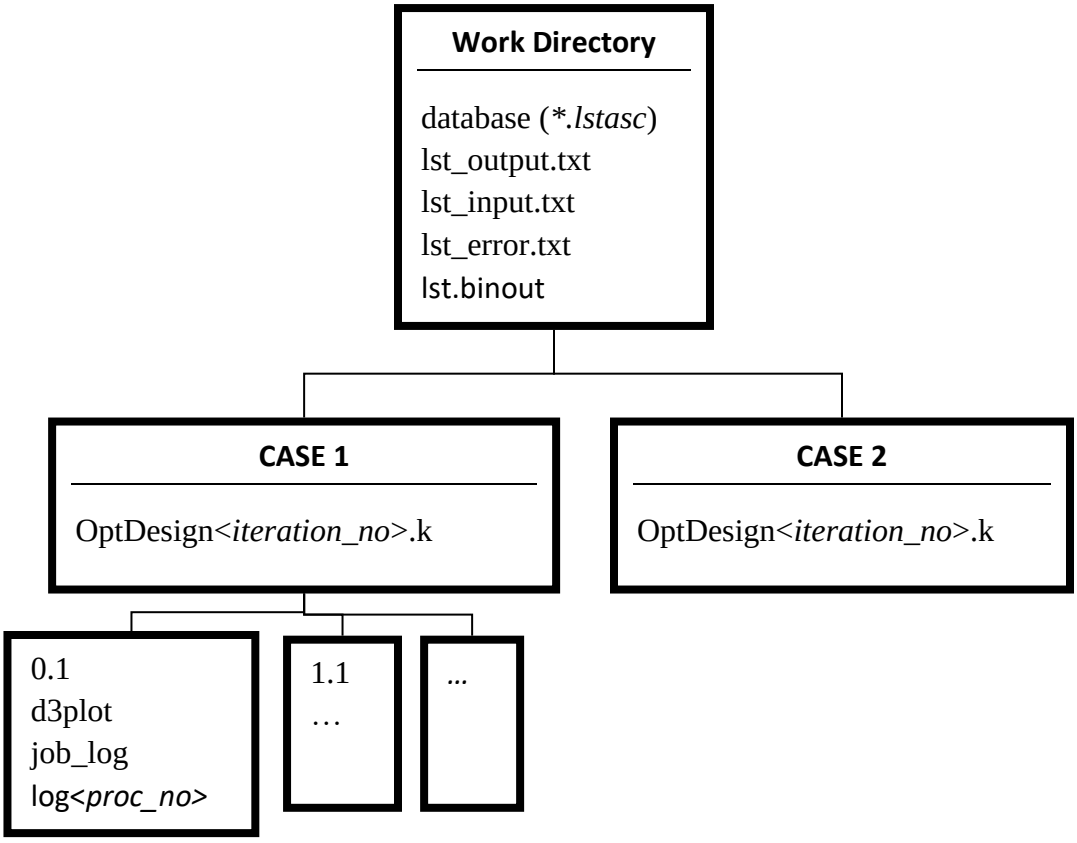

Figure 4-18 Directory structure

4.11. Restart

The program always attempts to restart from the existing results. To prevent a restart, you have to delete the previous run directories and the LS-TaSC runtime databases (use the Clear Results button on the run panel). Do not delete any files if a restart is required, unless you suspect the file has been corrupted.

If a larger number of LS-TaSC iterations are desired, then it can be restarted from the last iteration. Simply set the number of iterations higher and run the LS-TaSC job. The successfully completed iterations will not be rerun.

If the LS-TaSC job has been interrupted, then it can be restarted using the same procedure. Simply rerun the LS-TaSC job in the same directory.

You can add certain minor edits to the LS-DYNA input deck between restarts. Say the optimization stops at iteration 12 due to a convergence problem. If you modify the input and restart, then it should resume LS-DYNA analysis at iteration 12 after reading the results for the previous iterations. This will work for minor model changes like contact definitions, but not for major changes to nodes and elements of design part like re-meshing.

The *lst.binout* file is used for the restart if it exists, but it can be corrupted. It contains (i) the values of the design variables computed and (ii) results stored for plotting such as histories and constraint values. It is safe to delete the file. The values will be extracted again from the d3plot files and the design variables computed. So the restart will be done without rerunning LS-DYNA. The restart will take longer though, specifically if the advanced options are set not to store filters in memory. An LS-DYNA job will be restarted for a specific iteration if the "finished" file in the run directory is deleted or missing for this iteration.

You cannot use restart to change the bound on a constraint. This will change the designs computed and analyzed. In this case, begin in a clean directory.

You can add a constraint with neither a lower bound nor an upper bound and use restart to extract the constraint values purely for monitoring, because this does not affect the design computed.

Restart can be used to write out the <SOLVER_NAME>/OptDesign<iteration>.k file at an earlier iteration. For this simply set the iteration number to desired value in the Completion tab of the Method panel. The file will then be written for this design iteration. Files for later design will not be affected, and this file can therefore be written for multiple iterations.

4.12. Script Commands

The script commands issued to create the database can be viewed from the Script pull-down menu. Use these commands as a template for scripts.

5. Troubleshooting

This chapter lists some of the most common errors and suggested remedies.

5.1. LS-Dyna execution

5.1.1. Executable failing or no output

For the example problems: check that you changed the name of the LS-DYNA executable in the example problem to what is used on your computer.

For eigenvalue design the LS-DYNA version must be R10 or newer.

Provide the complete path for the solver command instead of using *alias*. You may also specify necessary DYNA options in the command, e.g.,

```
/home/Tushar/bin/ls971_single memory=100m
```

5.1.2. Nonconvergence of the LS-Dyna run

This can be many things. Most common are hourglassing, too large a time step, and negative volumes.

It may also be that the strain levels in the base iteration is excessive. The original model may be able to resist the load, but in the base iteration with a mass fraction of say 0.1 the load may be too large for the structure (or element technology).

5.1.3. Hourglassing

Specify hourglass control in the LS-Dyna input deck. Stiffness form, hourglass type 4, is usually effective. Refining the mesh could also help prevent the issue.

5.1.4. Negative Volumes

While care has been taken to avoid running into negative volume errors, sometimes the simulation terminates due to negative volume errors.

A user can take several actions to correct this error. Some options, in approximate order of importance, are:

- Reduce the time step or time step scale factor.
- Check for hourglass issues and apply hourglass control.
- Increase minimum density fraction (default 0.05 for dynamic problems).

- Check the CONTACT cards. Note that the failed run probably has elements with soft material interface with elements with harder material; hence care must be exercised in defining master and slave penalty stiffness factors.
- Specify SOFT=2 option on the control card.

5.1.5. The LS-DYNA analysis fails if a smaller mass fraction is requested

Possibly the structure is not strong enough to support the load.

Inspect the d3plot results in the failed iteration to understand what happens in the LS-DYNA analysis.

Fixes are to reduce the load, increasing the mass fraction, changing the FE model to be more robust, using a finer mesh, modify your approach keeping in mind that you cannot get a solution from that starting mass fraction, or accepting that a design does not exist at that mass fraction.

5.1.6. Mysterious Error when/after calling LS-DYNA and/or Errors involving the LSOPT Environment Variable

Make sure the queuing is set correctly. Specifying the use of a queuing system when none is available may cause (i) mysterious errors or (ii) the LS-DYNA execution not to return after finishing.

Make sure the LSOPT environment variable is not set.

5.1.7. Job submission errors using queuing options

Using Runqueuer/wrapper option: Make sure LSOPT_PORT and LSOPT_HOST environment variables are passed to the job submission script where “wrapper” executable is defined. Refer to “Job_log” file for errors.

Using Blackbox option: Make sure that the LsoptJobCheck script outputs a valid statement (WAITING, RUNNING N/M, FINISHED or FAILED). Refer to “job_log” file to debug the errors.

The job “Job_log” is in the run directory (e.g. “./SOLVER_1/3.1” and should be checked before checking the file “lscheduler.debug” in the root directory (the one where LS-TaSC was executed).

5.2. LS-TaSC execution

5.2.1. Design Part

The design part is not found: check that the DYNA input deck has the same part id for the design part as specified in the input file. In the case of the multiple load cases, the design domain must remain the same.

5.2.2. Extrusion Set

The extrusion set is not found: check that the set of elements on the extruded face are grouped under the *SET_SOLID option in the DYNA input deck. The ID of the set is same for all load cases as specified in the input file.

Unable to find all the slaved elements: if the node numbering order is different for some elements, then the algorithm may fail. Using a different node number will, for example, cause face 1 to be the top face on one element and to be the left face on another element; the algorithm depends on this not happening.

5.2.3. Oscillations of the LS-TaSC solution

For some problems, the code does not converge; instead, oscillations set in. The user must look at the geometry to understand why oscillations are observed. Mostly, oscillations indicate that there is more than one possible optimal solution.

One fix is to reduce the move limit on the design variables using the advanced settings.

5.2.4. Casting definitions

Using the Advanced Options in the File pull down menu, you can set a debug flag, which will dump a definition of the faces to a file for display in LS-PREPOST. Investigating the casting faces in LS-PREPORT can help debug the issue.

5.3. LS-PREPOST

You may need to install another version of LS-PREPOST into the LS-TaSC installation directory. Please follow the instructions on the LS-PREPOST web site. The name of the executable must be *lsprepost*.

6. User Results

This chapter describes how to import results from other analysis software.

6.1. Background

For importing user results the user must compute a either the design sensitivity information or optimality criteria value for each element. The design sensitivity information can for example be the derivative of a displacement or eigenfrequency with respect to the element variables. An example of an optimality criteria would be the energy density of each element.

See the later sections for the requirements regarding the load case, derivatives, settings, and files.

6.2. Steps in a user analysis

For every iteration, LS-TaSC will call the user analysis in the run directory. The following steps will be performed:

3. LS-TaSC: LS-TaSC writes a file describing the variable values to the"*VariableValues.txt*" file.
4. USER: The user program reads the variable values from the "*VariableValues.txt*" file.
5. USER: The user program analyzes the structure
6. USER: The user design sensitivity information are written to the "*UserResults.txt*" file.
7. USER: The user program, as a final step, write "N o r m a l t e r m i n a t i o n" to the standard output.
8. LS-TaSC: LS-TaSC checks for "N o r m a l t e r m i n a t i o n" in the output from the user program (this output goes to a log<<*pid_number*>> file).
9. LS-TaSC: LS-TaSC read the "*UserResults.txt*" results from the file.

6.3. Details of the load cases

The first load case must be a normal LS-Dyna load case. LS-TaSC reads the input deck associated with the first load case, looking for various information like set definitions. If required, make this a dummy load case by setting the weight to very small (but non-zero).

Only one objective per load case is allowed. If you evaluate three performances in a single analysis, then you have to set up the other two analyzes to refer to the data in the first one.

The load case name should be start with the letters "USER_" ; for example, "USER_freq1" or "USER_A1".

The executable name should be the name of the user-defined program. The program can be a script scheduling any number of other programs or post-processing.

6.4. Details of the objective and design sensitivity

A good choice of the objective is a global quantity; for example, the external work, compliance, or a fundamental frequency. A local quantity such as an element stress can be difficult to handle.

The input to LS-TaSC is the derivatives of the objective with respect to the density variable for each element in the design domain as described in the theory manual. So these element values are a user-defined design sensitivity value for each element. For example, the derivative of the objective *P* with respect to the *i*th design element $\frac{\partial P_j}{\partial x_i}$.

The design process minimizes this objective, and the derivative of this objective with respect to the element variables must be provided. For example, if you minimize F(x) with F(x) the compliance, then the user must provide dF/dX1, dF/dX2, …, dF/dXn.. But if you maximize F(x) with F(x) the natural frequency, then the user must provide -dF/dX1, -dF/dX2, …, -dF/dXn.

6.5. Format of the variable value file

The name of the file is "*VariableValues.txt*". The user code must read this file to obtain the values of the variables assigned to each file.

The file contains two columns: the element ID and the variable value. Here is a sample "*VariableValues.txt*" file:

$ 32 vars given as element id, and variable value

$ all numbers are 10 characters wide

1 0.001

2 0.001

…

31 0.86

32 0.89

$end

Lines starting with a $ should be ignored.

6.6. User results

The name of the file is "*UserResults.txt*". LS-TaSC will read this file.

LS-TaSC will print the first and last value read to the screen and the lst_output.txt file for debugging purposes.

The file must have two columns: the first column is the element id and the second is the element DSA value. Here is a sample "*UserResults.txt*" file for 32 elements:

1 4.88738e-06

2 5.87231e-06

…

31 0.00468277

32 0.0114259

6.7. LS-TaSC default input values

The defaults values are not implemented keeping user results in regard. The user should check all values.

6.8. Debugging

You should be able to do most of the debugging of the user analysis outside of LS-TaSC. You may want to run LS-TaSC once to get the "*VariableValues.txt*" file.

Once you are running LS-TaSC, then the output from your program will go to a file named log<<*pid_number*>>. This file is located in the run directory.

If you are running over a queuing system, then the "lscheduler.debug" file in the top level directory may be helpful.

LS-TaSC will print the first and last value read from the "*VariableValues.txt*" file to the screen and the lst_output.txt file for debugging purposes.

6.9. Solid/Void (SIMP)

Firstly, set LS-TaSC to explicitly use SIMP. The user program must compute the derivatives considering the SIMP constants.

6.10. Element vs material volume

This is only for density values such as the Internal Energy Density in an Element. Right now LS-TaSC scales the Internal Energy Density values from LS-DYNA from element volume based to the material volume based. This is not done for user results. For a typical case (say for the case of minimize compliance) simply provide dF/dX1, dF/dX2, …, dF/dXn.

6.11. Symmetry and extrusion definitions

This requires no special handling. The required information is read for the input file associated with the first case, as usual.

7. Other LS-TaSC MANUALS

The functioning of LS-TaSC is described in a number of manuals. The standard user will only be interested in the users's manual. The more advanced topic are therefore supplied as separate manuals to keep the size of this manual down to what the normal user will require.

7.1. Example problems manual

The example problems manual is available in the same location as your LS-TaSC executable.

7.2. Theory manual

The theory manual is available in the same location as your LS-TaSC executable.

7.3. Scripting manual

The scripting manual is available in the same location as your LS-TaSC executable.

7.4. Queueing system installation

The queuing system installation manual is available in the same location as your LS-TaSC executable.