The LS-TaSC™ Tool

Topology and Shape Computations

User's Manual

Version 3.1

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

Disclaimer

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

A heuristic topology optimization method developed at the University of Notre Dame, known as hybrid cellular automata, as described in the LS-TaSC Theory Manual, showed potential in handling topology optimization problem for crashworthiness problems. This method updates the density of elements based on the information from its neighbors. No gradient information was required. The simplicity and effectiveness of this method for both two- and three-dimensional problems made it an attractive choice for our initial implementation. The methodology has however been enhanced using more general approaches as well; currently, amongst others, it gives mesh independent results. Our methodology is therefore best referred to as simply the LS-TaSC 3.1 methodology (the product name together with the version number).

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 typically the stiffest structure for the given weight (minimum compliance design), but with an upper bound on the internal energy density.

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.

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 should be consulted for feedback on how to modify the material stress-strain curve in the input deck.

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.

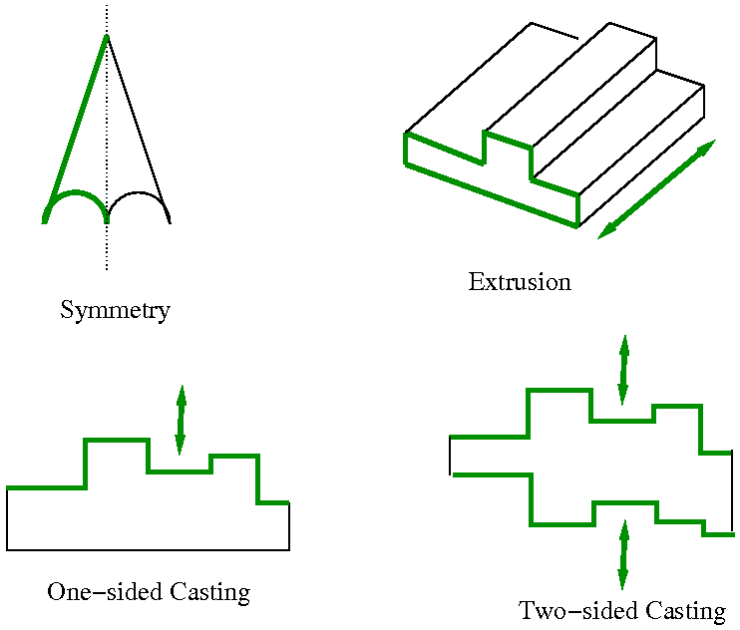

Figure 2-1: Geometry definitions

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.

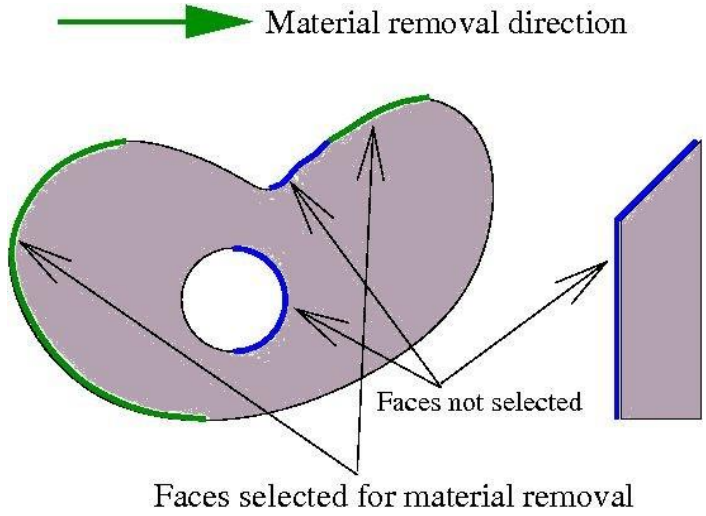

Figure 2-2: The faces selected for design in a casting definition are all the faces facing the material removal direction. The algorithm will not consider the faces shown in blue.

2.3. Convergence

The 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.

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.

Typically the problem is converged in less than 30 iterations, but this is not guaranteed.

2.4. Design Variables

2.4.1. 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.

2.4.2. 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.

2.4.3. 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 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.

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

2.5. 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.

2.5.1. The Contact Definition

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.

2.5.2. Part Definition

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

2.5.3. 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.

2.5.4. 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.

2.5.5. 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 *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.

2.5.6. 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.

2.5.21. 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 4.4.2.

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. Dynamic Load Cases 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 $k_1C_1 + offset_1 = k_2C_2 + offset$ 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 are too large.

2.7. General Constrained Optimization

Constrained optimization uses the part mass fractions and load case weights as 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 always 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.

2.7.1. 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.

2.7.2. 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.

2.7.3. 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. 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.

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.

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.

Other (user-defined) responses can be defined by specifying a string. The only allowable commands are the *D3PlotResponse* and *BinoutResponse* commands as defined in the LS-OPT manual. Use LS-OPT to create these strings.

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

This can be specified in conjunction with dynamic weighting.

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.

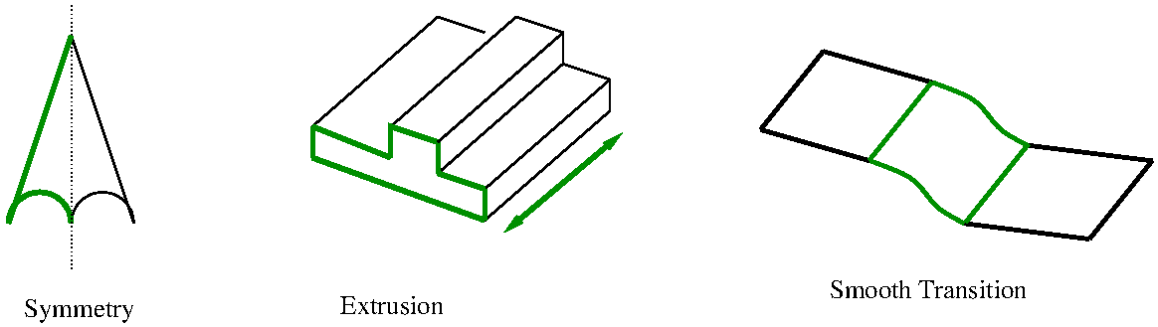

Figure 3-1: Geometry definitions

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.

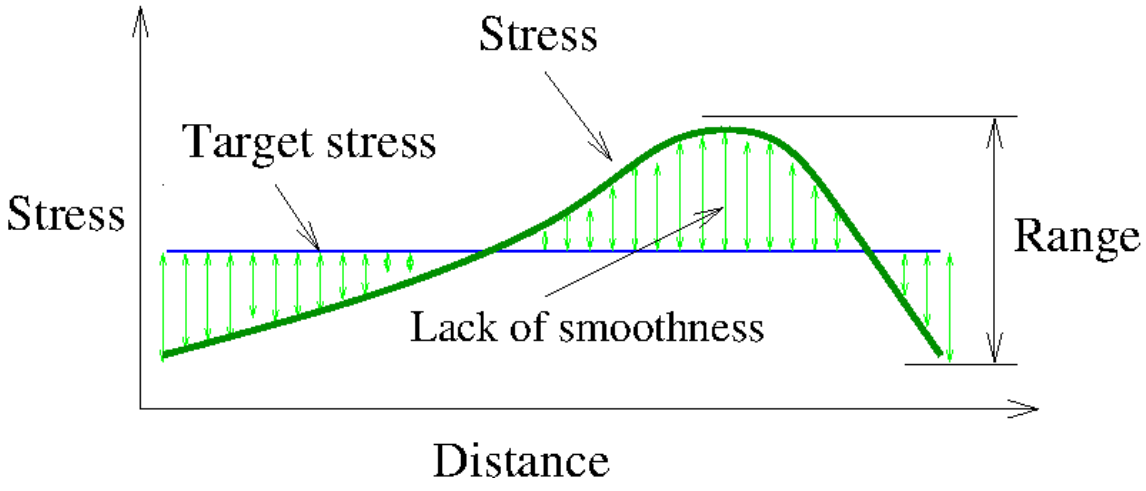

Figure 3-2 Convergence for shape design

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 Execution

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:

5. With the execution command

   $ lstasc myProject.lstasc

6. 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.

4.3.1. 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 4.4.2.

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 4.5.1) to read results from the *d3part* database instead.

4.4. 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.

4.4.1. 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.

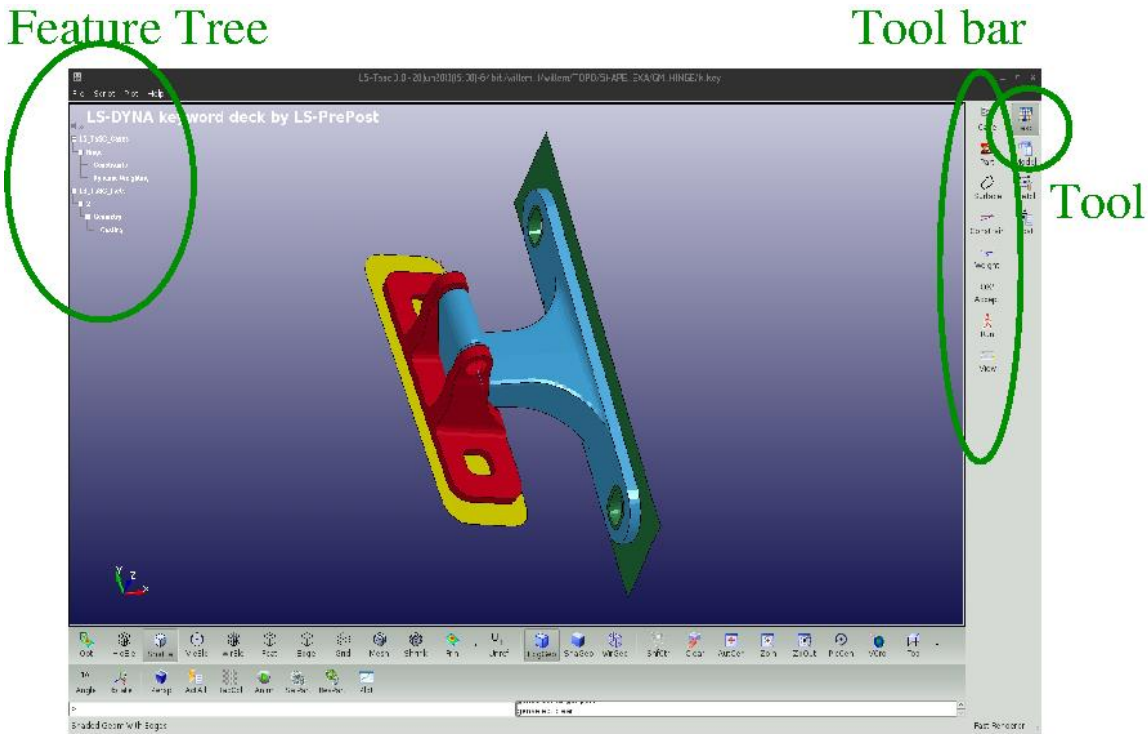

Figure 4-1: The toplevel GUI

4.4.2. 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 processes to be run simultaneously. A value of zero indicates all processes would

`jobs`	be run simultaneously. This parameter only makes sense if multiple cases must be evaluated. The program will allow as many processes as defined for the current case being evaluated.
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.

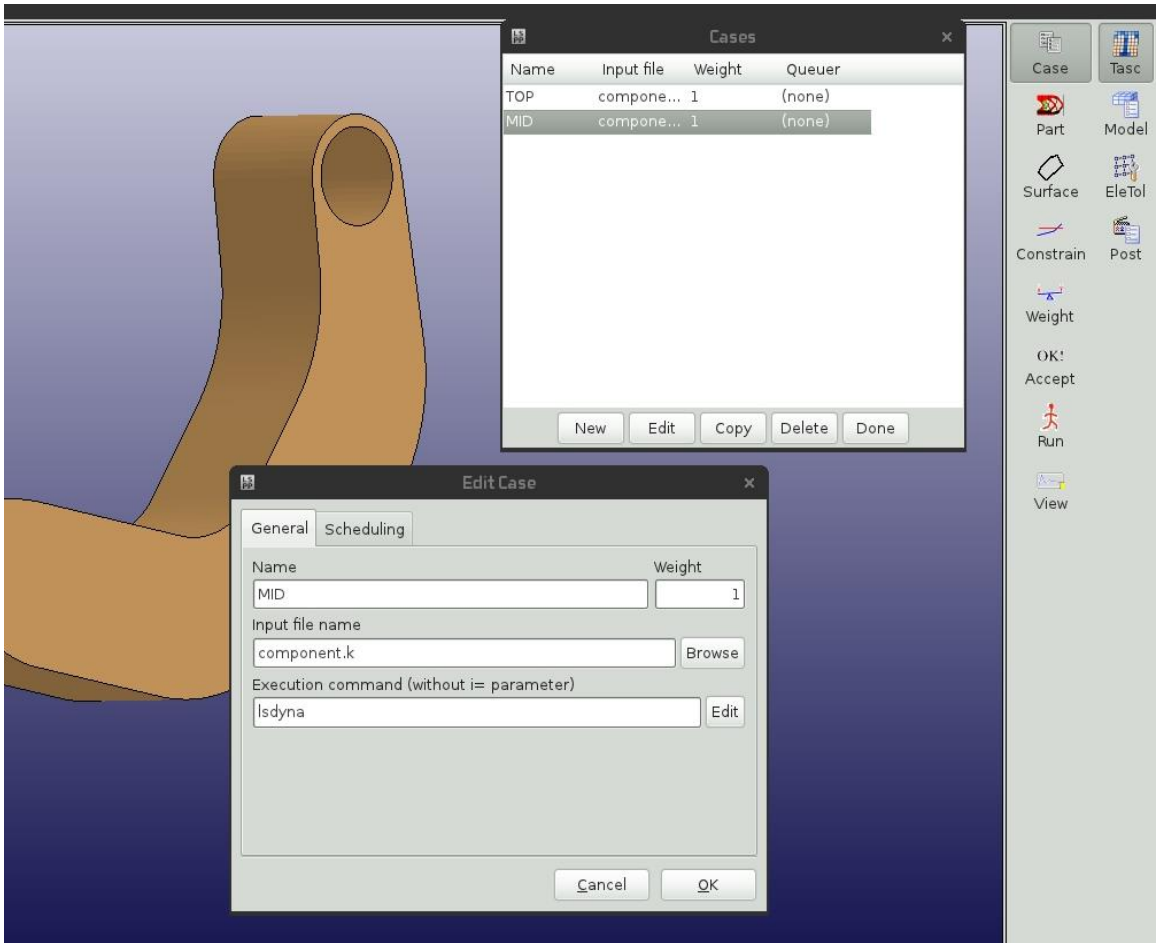

Figure 4-2: The cases panel.

4.4.3. 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 4-3, and Figure 4-4 for more details.

Table 4-2: 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.

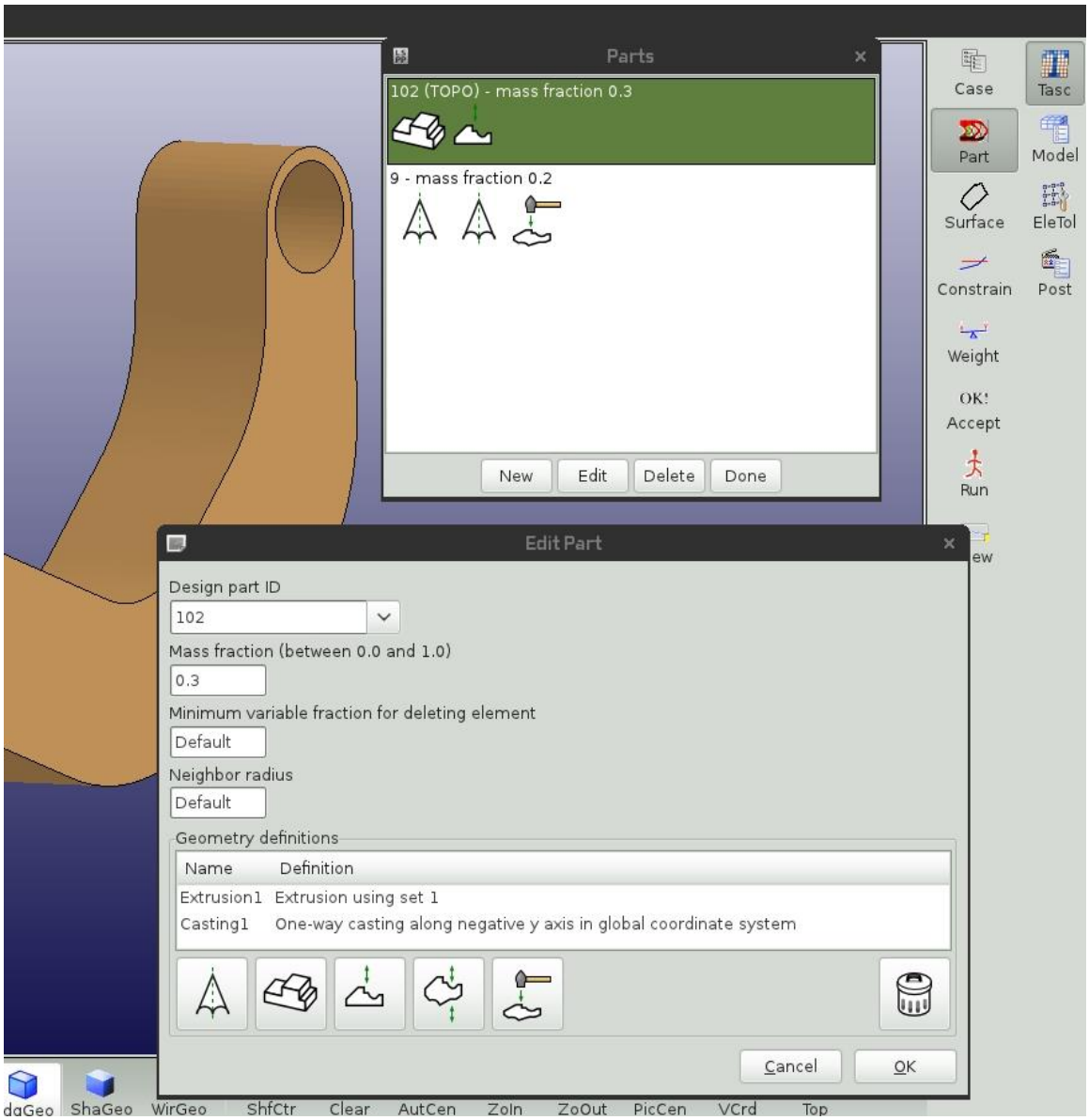

Figure 4-3: The parts panel.

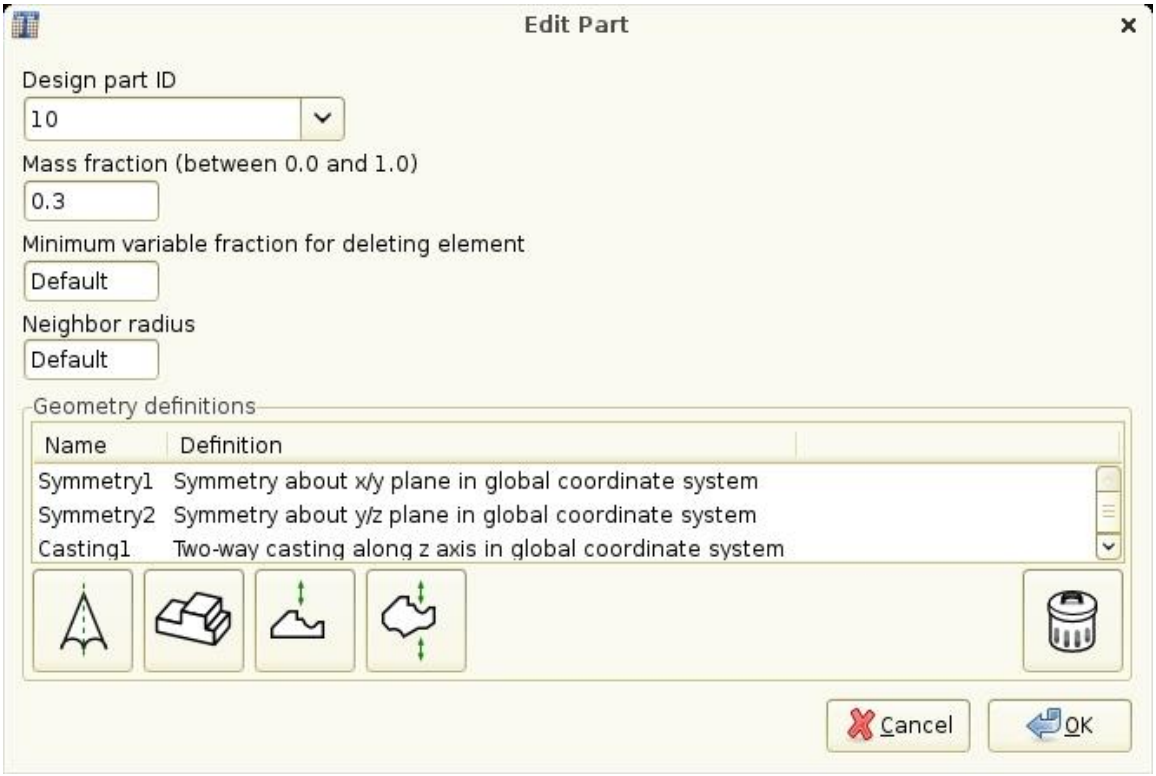

Figure 4-4: The panel to create part and geometry.

4.4.4. 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 4-5 for more details.

Table 4-3: 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 in an iteration.
Remesh depth	This is the number of elements that should be considered in re-meshing after a shape change was done.

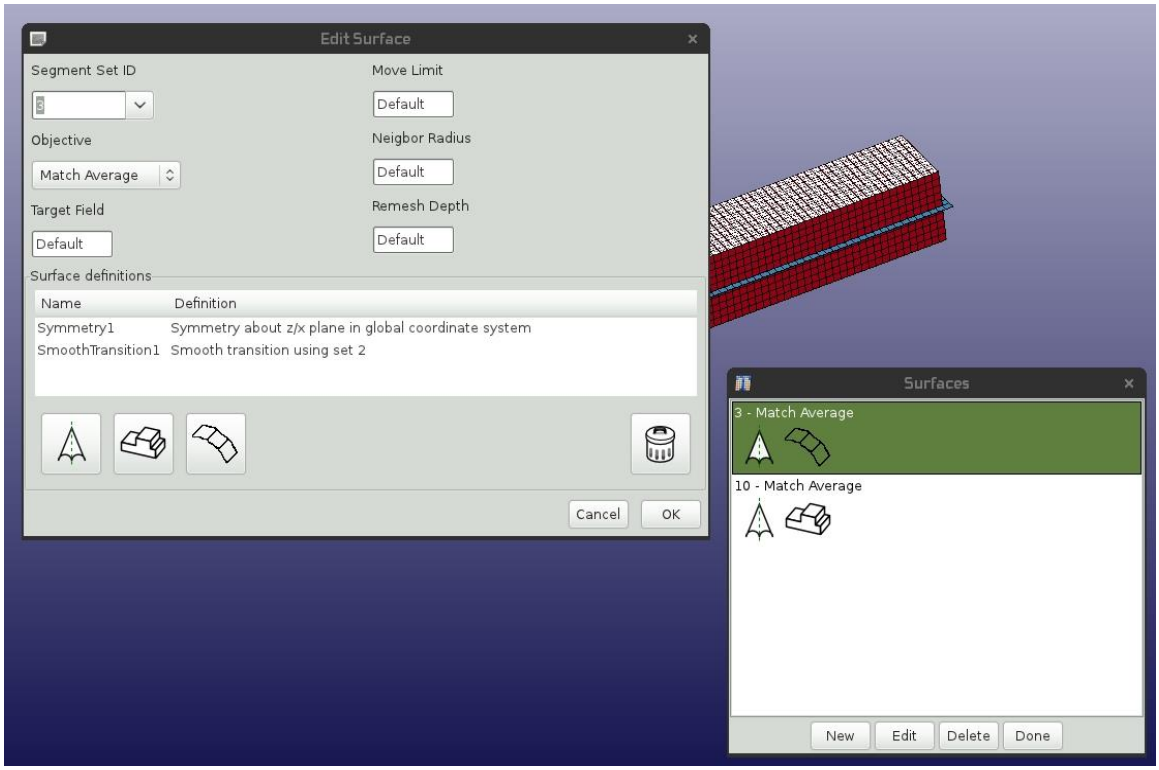

Figure 4-5: The surface panel.

4.4.5. Part and Surface Geometry

The geometric properties can be defined for every part and surface. See the following table and Figure 4-6 for more details.

Table 4-4: 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 forging geometry definition will always result in a minimum material thickness in the forging direction.
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.

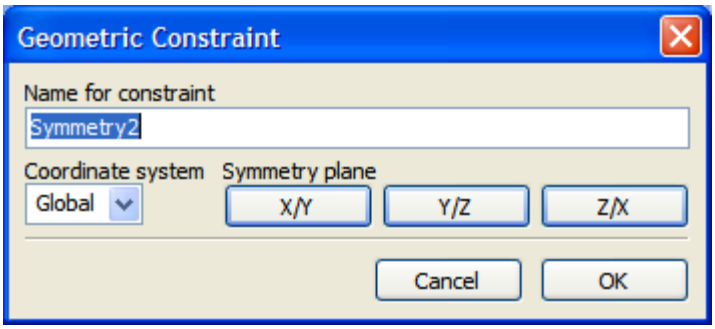

Figure 4-6: Creating a geometry constraint. A symmetry constraint is shown.

4.4.6. The Constraints Panel

The constraint panel contains the global constraints on the structure. See the following table and Figure 4-7 for more details.

Table 4-5: 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.

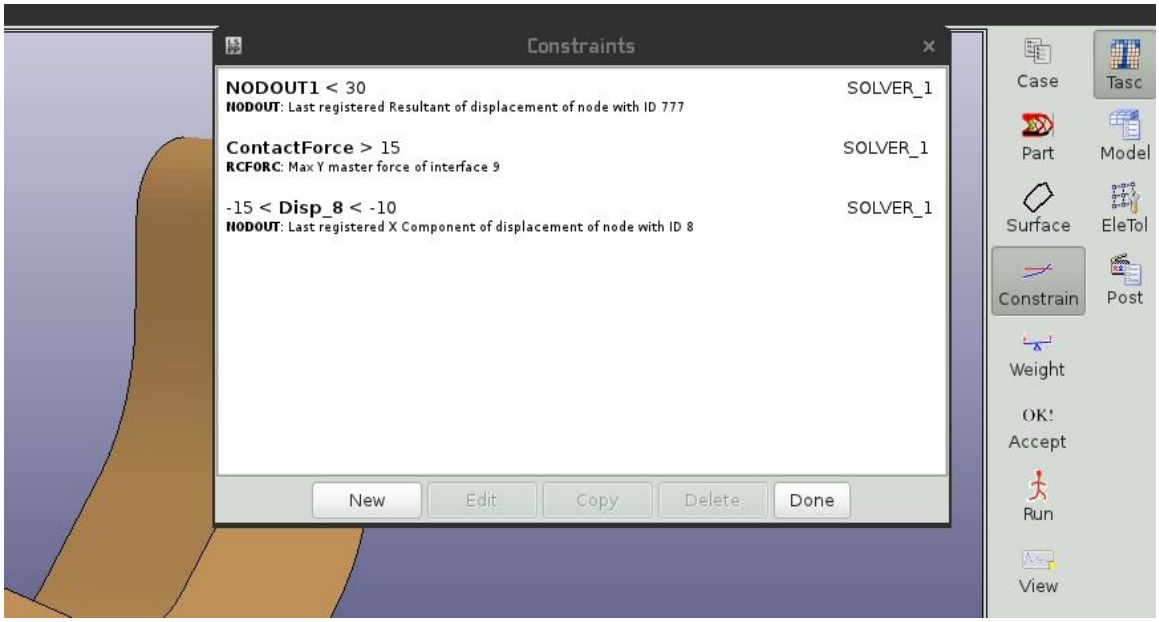

Figure 4-7: The constraints overview panel.

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”.

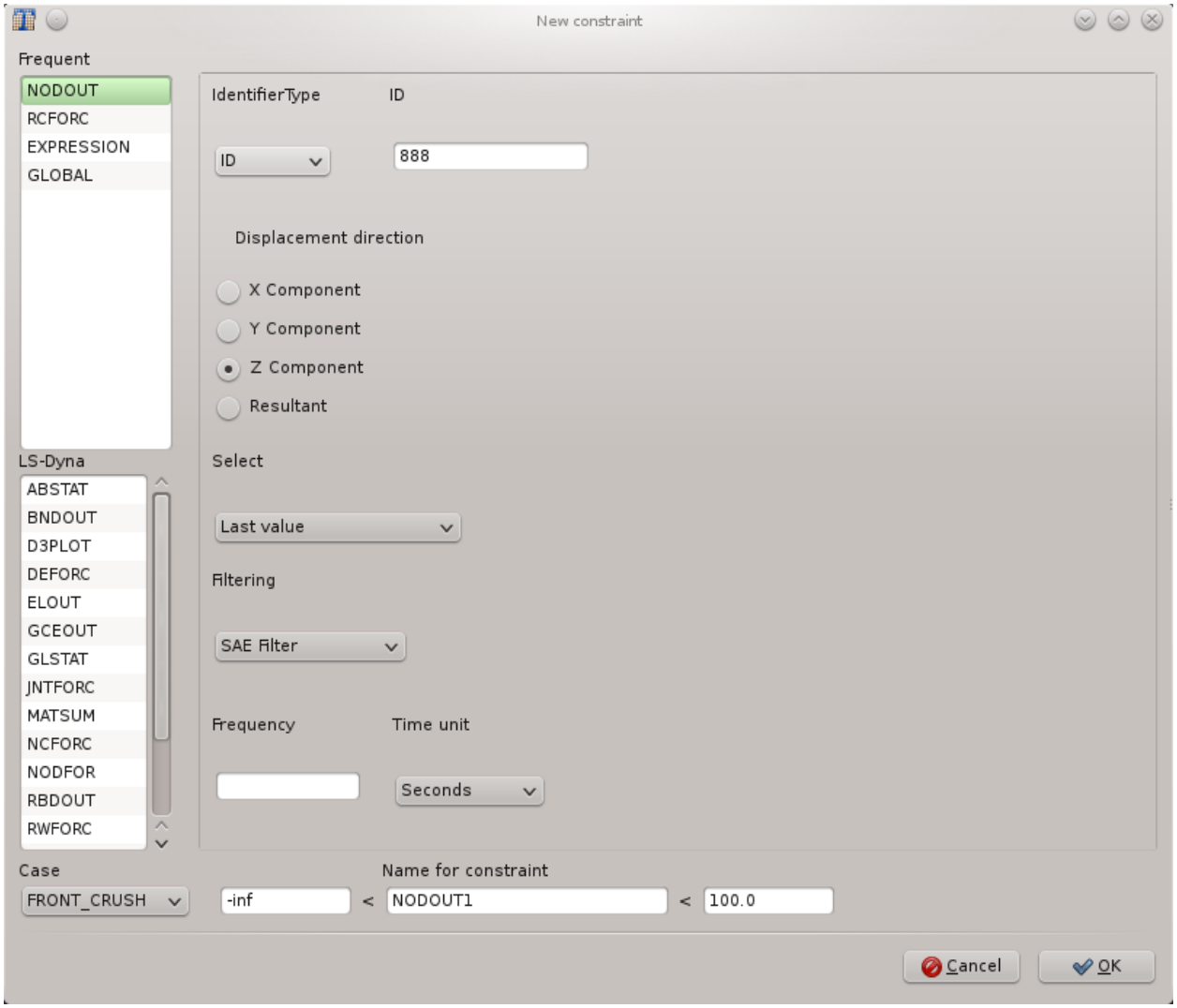

Figure 4-8: The constraints creation panel.

4.4.7. The Dynamic Weighing Panel

The dynamic weighing panel is used to specify the equation defining the dynamic weighing. See the following table and Figure 4-9 for more details. The load case weights will be adjusted to satisfy this equation.

Table 4-6: 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.

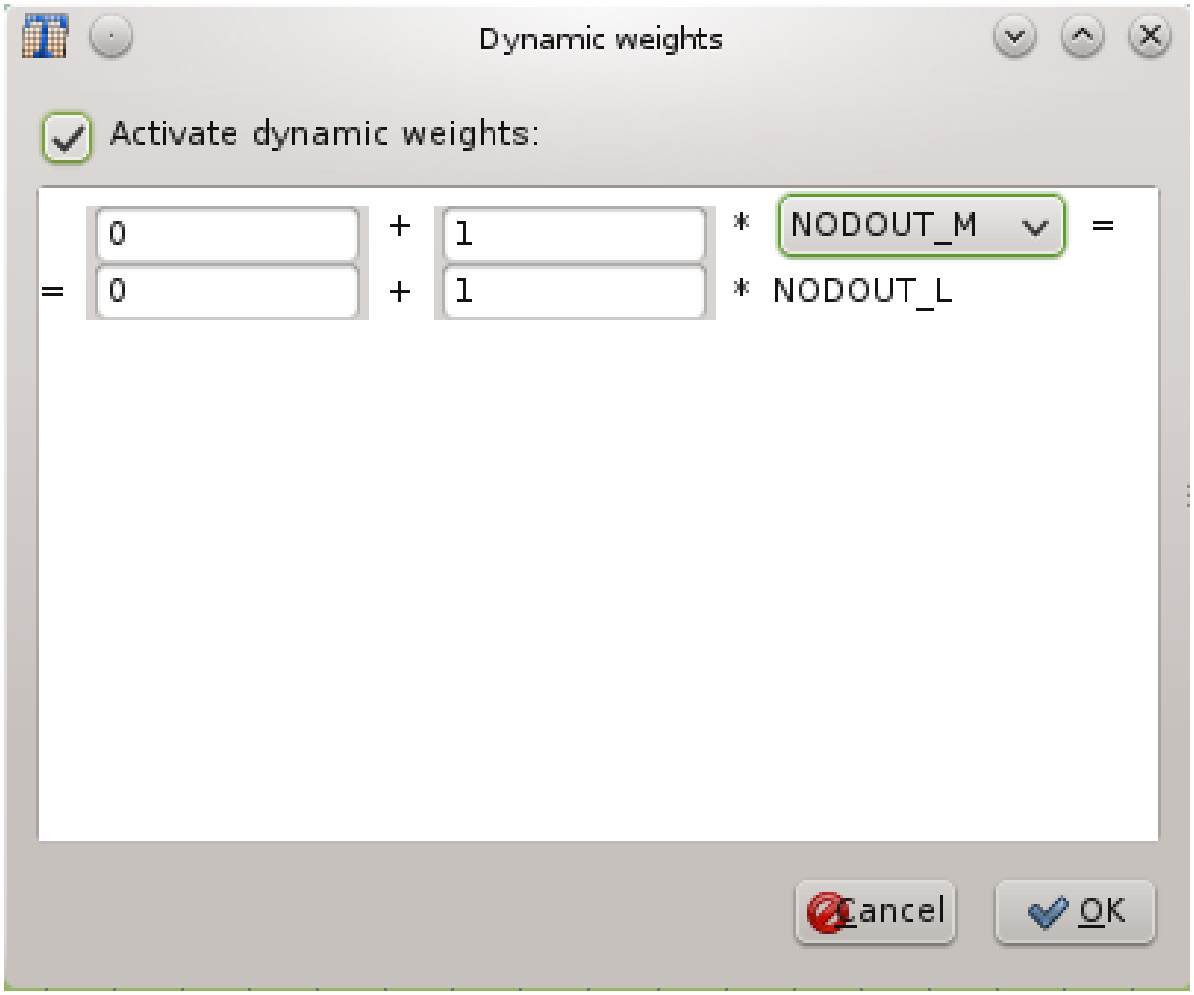

Figure 4-9: The dynamic weighing panel.

4.4.8. 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.

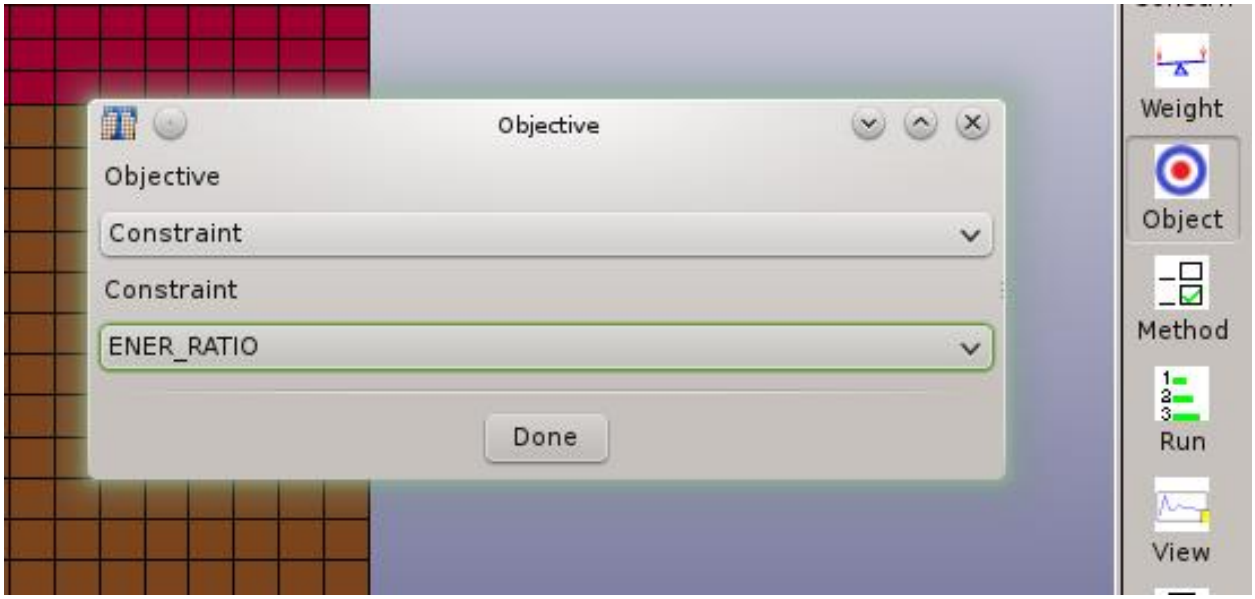

Figure 4-10: The objective panel.

4.5. Setting the Design Methodology

4.5.1. Job Completion 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.

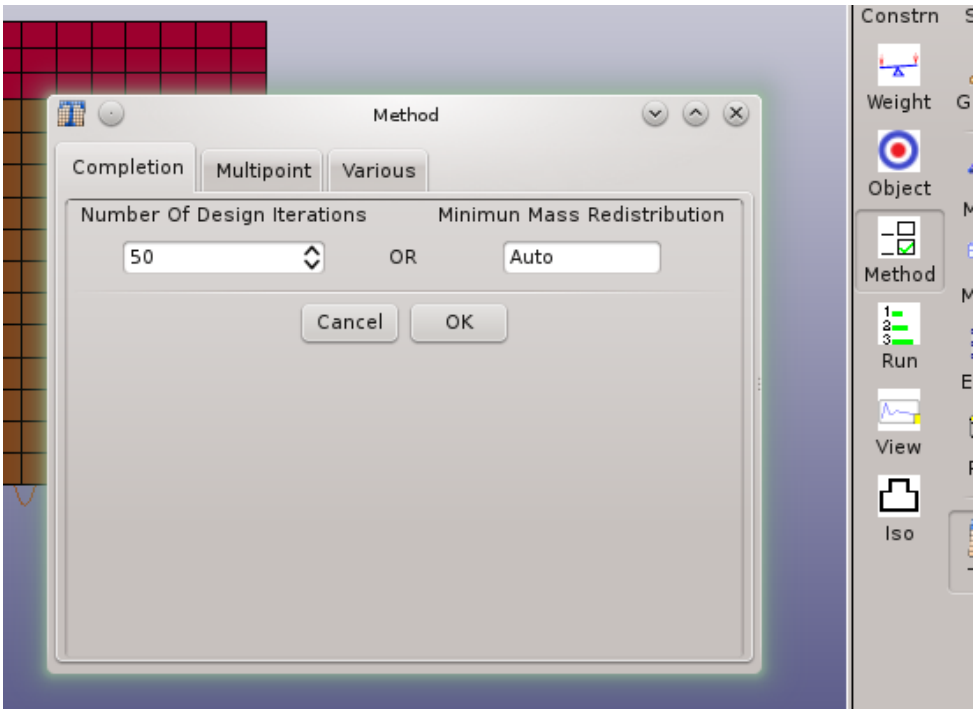

Figure 4-11: The completion options in the method panel

4.5.2. Multipoint options

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

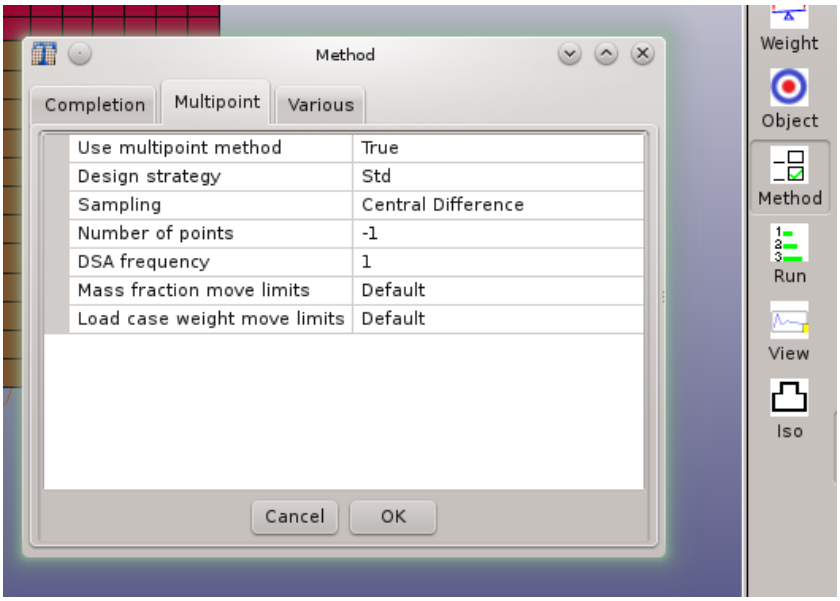

Figure 4-12 The multipoint options in the method panel

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.
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.
Number of points	The number of points to be used to construct either the linear response surface or in the random search.
DSA frequency	This is how often the multipoint method should evaluate the design sensitivity information numerically using the multipoint method.

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.

4.5.3. Miscellaneous Options

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

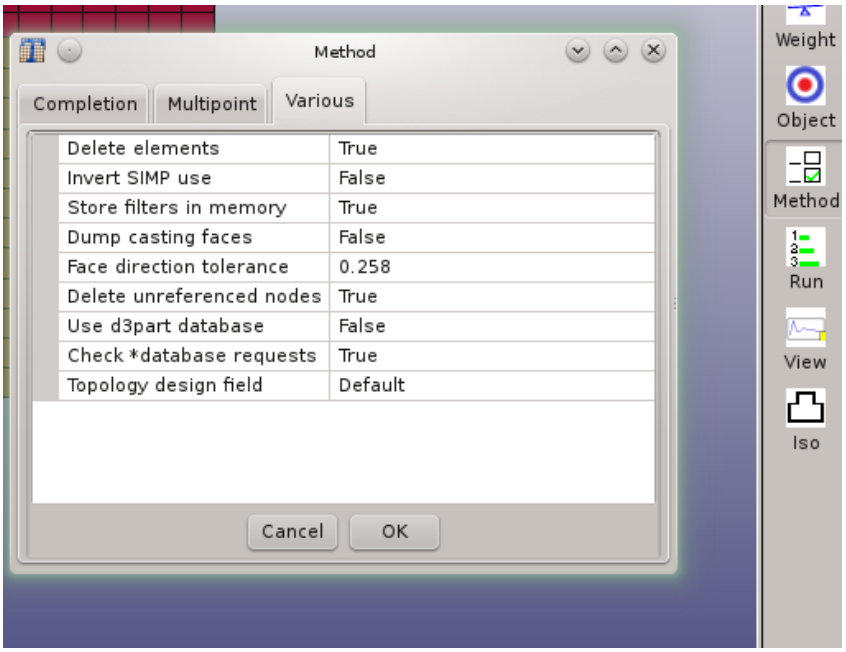

Figure 4-13 The various options in the method

The available options are described in the following table.

Table 4-10: Advanced 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.
Invert SIMP use	The normal SIMP use can be inverted such that it is not used for solids, but used for shells.
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).
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.
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.

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.
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.
Check database requests	If constraints are defined, then a check is done to see that the FE deck include the required *DATBASE entries. This can be disabled using this option.
Topology design field	This is the criteria used to decide whether the material in an element is utilized. One of Internal Energy Density or Von Mises.

4.6. 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.

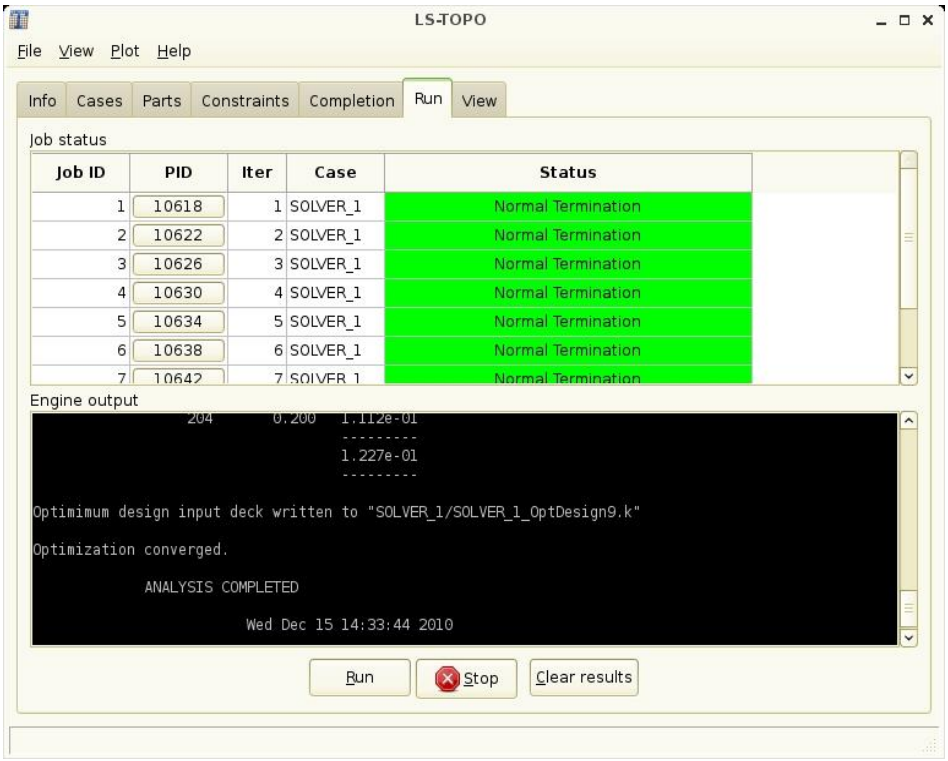

Figure 4-14: The run panel.

4.7. 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.
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.
*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.

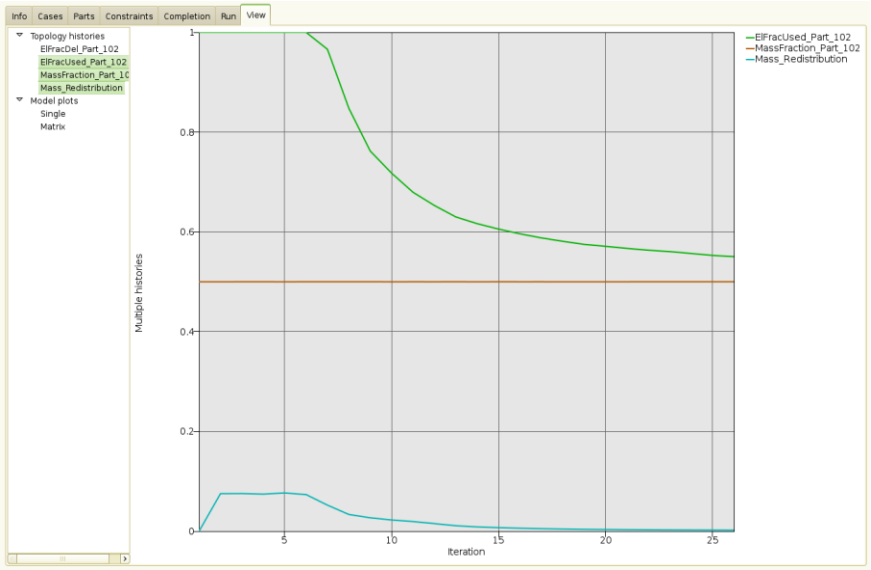

Figure 4-15: The view panel with histories.

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.

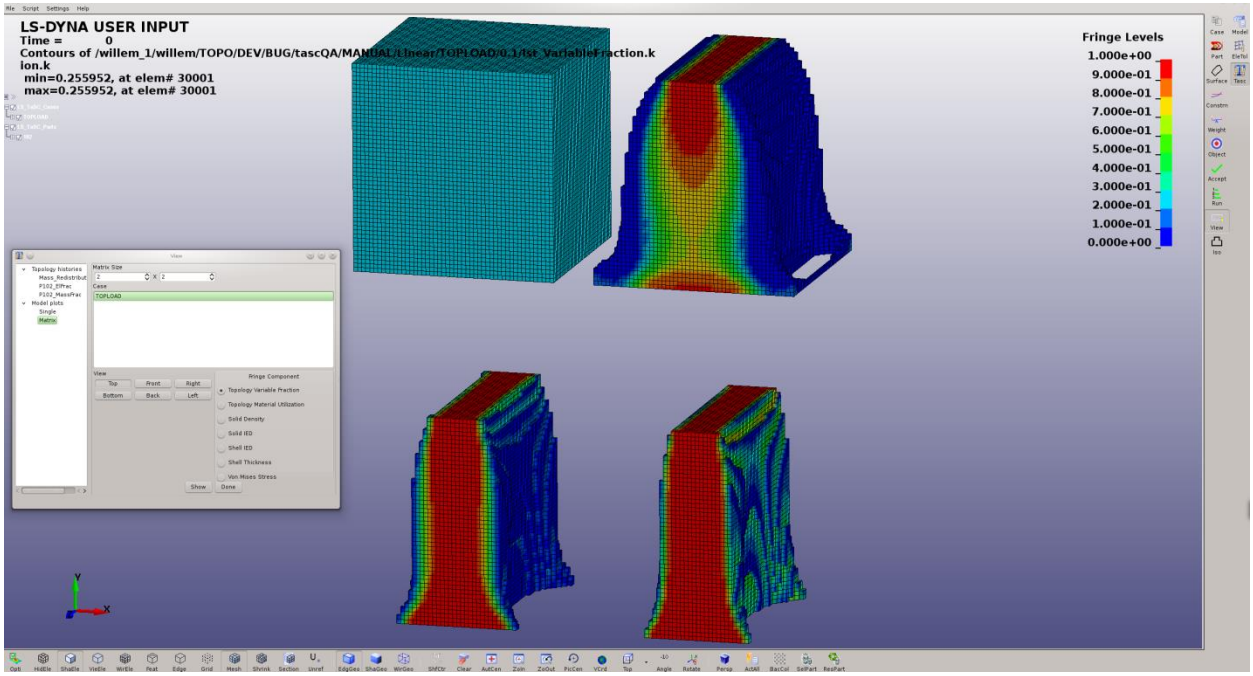

Figure 4-16: Viewing the model evolution in LS-PREPOST.

4.8. 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.

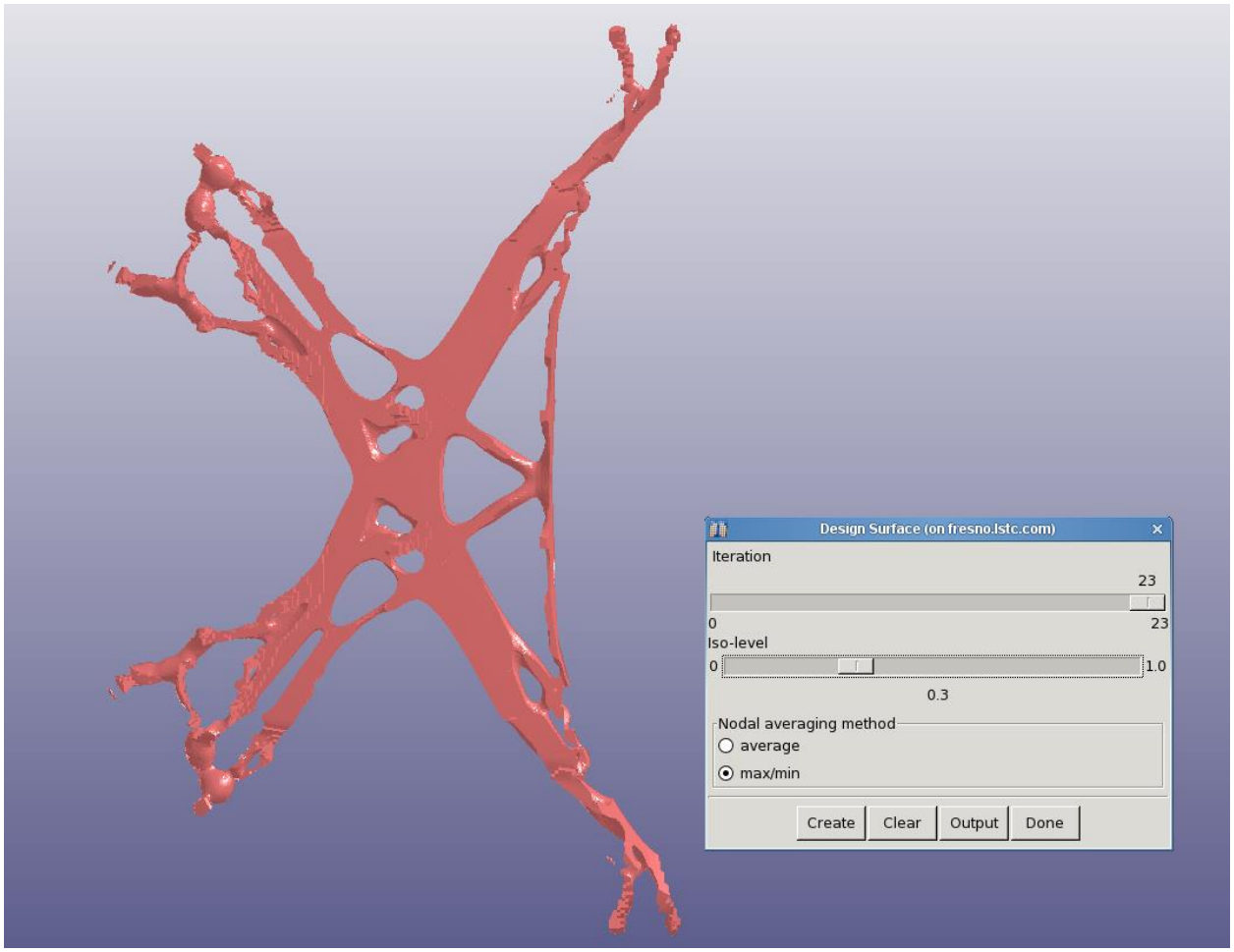

Figure 4-17: Creating an iso-surface plot of a design. The design shown is that of the interior of a bonnet.

4.9. 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

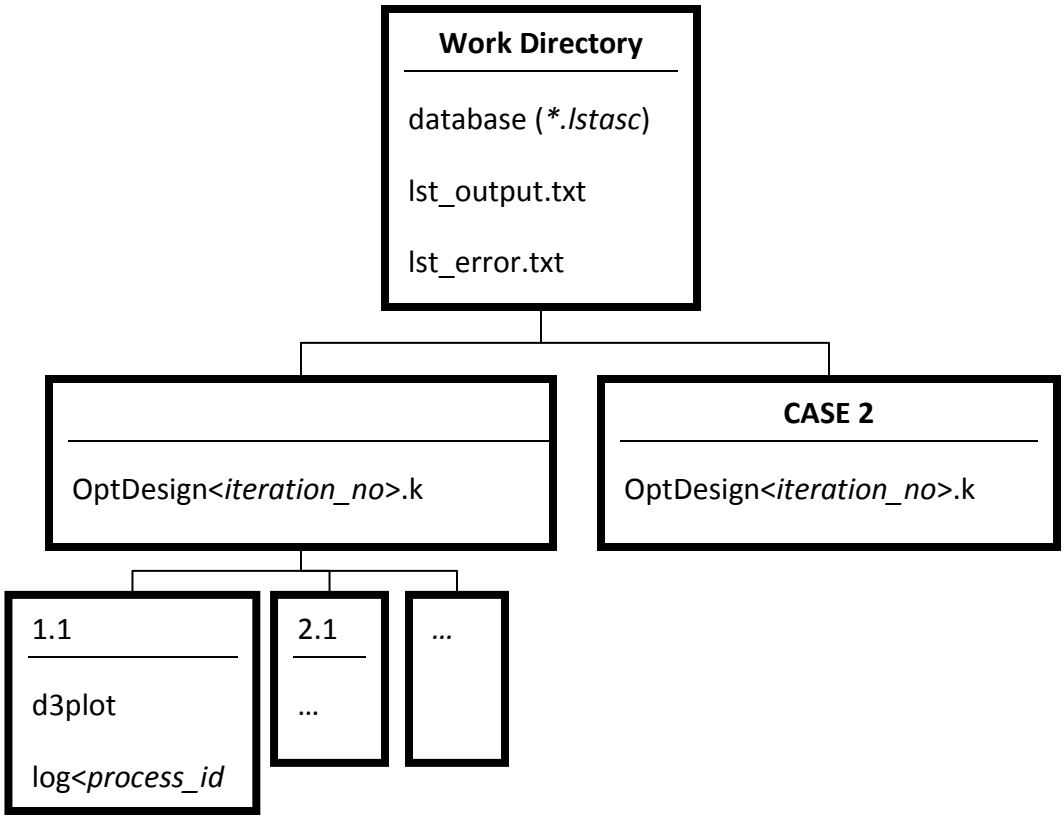

Figure 4-18 Directory structure

4.10. 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.

4.11. 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. Example Problems

The applications of the topology code is demonstrated with the help of a few test examples below. The examples are supplied together with the software executables (manual_examples.tar).

5.1. Fixed Beam with Central Load

This example demonstrates

1. how to define a problem,
2. how to add a load case,
3. how to define the design part,
4. how to run the problem, and
5. the analysis of the results.

The related files are available in MANUAL/Beam.

5.1.1. Problem Description

This example simulates a beam that is fixed on both ends. A pole with assigned initial velocity of 10m/s hits the beam in the center. The design part is meshed using $5mm^3$ brick elements. The symmetry of the problem is used to design only half-section of the beam. The geometry and loading conditions of the beam are shown in Figure 5-1.

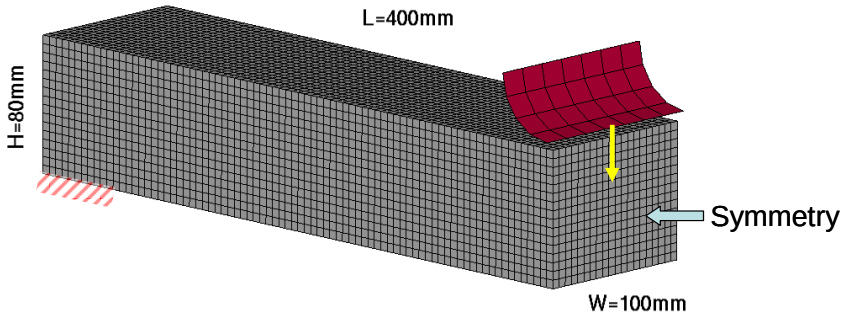

Figure 5-1: Geometry and loading condition.

5.1.2. Problem Setup

The project input data is saved to the file *lst_project.lstasc* as provided in the examples distribution. First, the *Case* icon from the main LS-TaSC GUI has to be selected, Figure 5-2. Specify the name of the load case, the LS-DYNA input file *Beam.dyn* and the LS-DYNA executable. The next step is to define the part to be optimized, Figure 5-3. Select the design part ID 101 and a desired mass fraction of 0.25. A maximum of 50 iterations are selected to find the optimal topology, Figure 5-4. Then run the optimization, Figure 5-5.

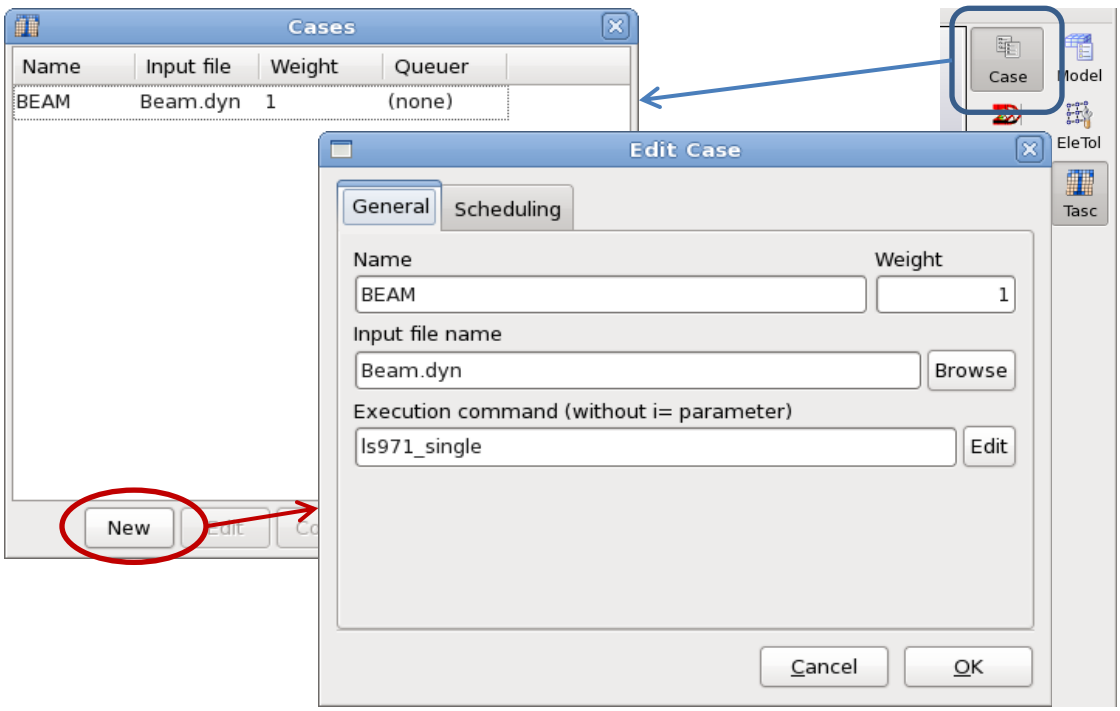

Figure 5-2: Definition of load case; specification of load case name, LS-DYNA input file and execution command.

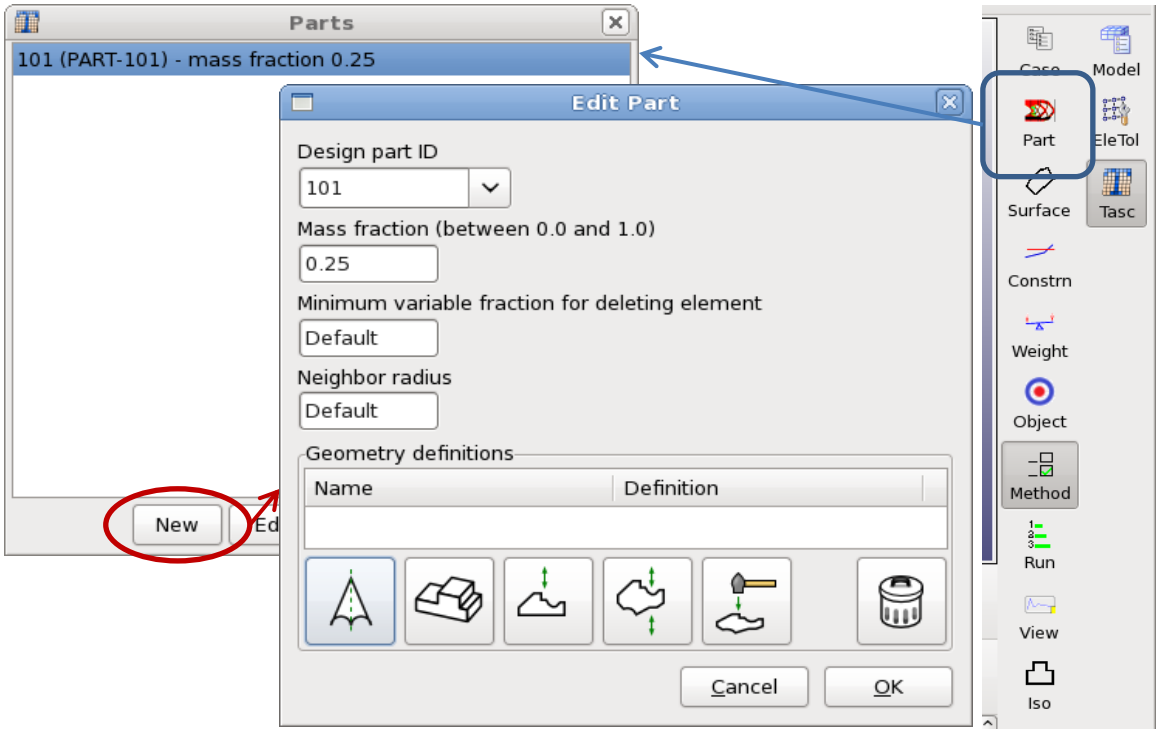

Figure 5-3: Definition of design part; specification of design part ID and desired mass fraction.

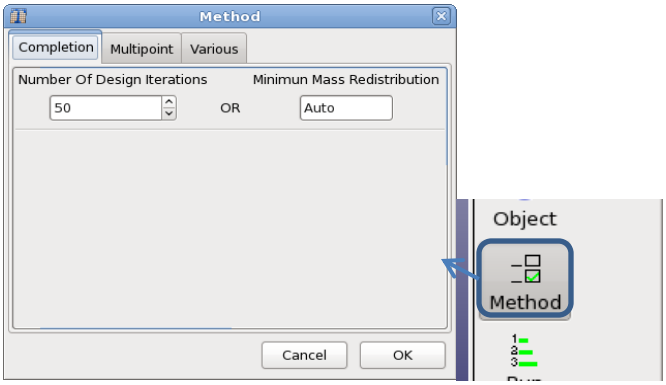

Figure 5-4: Definition of maximal number of iterations.

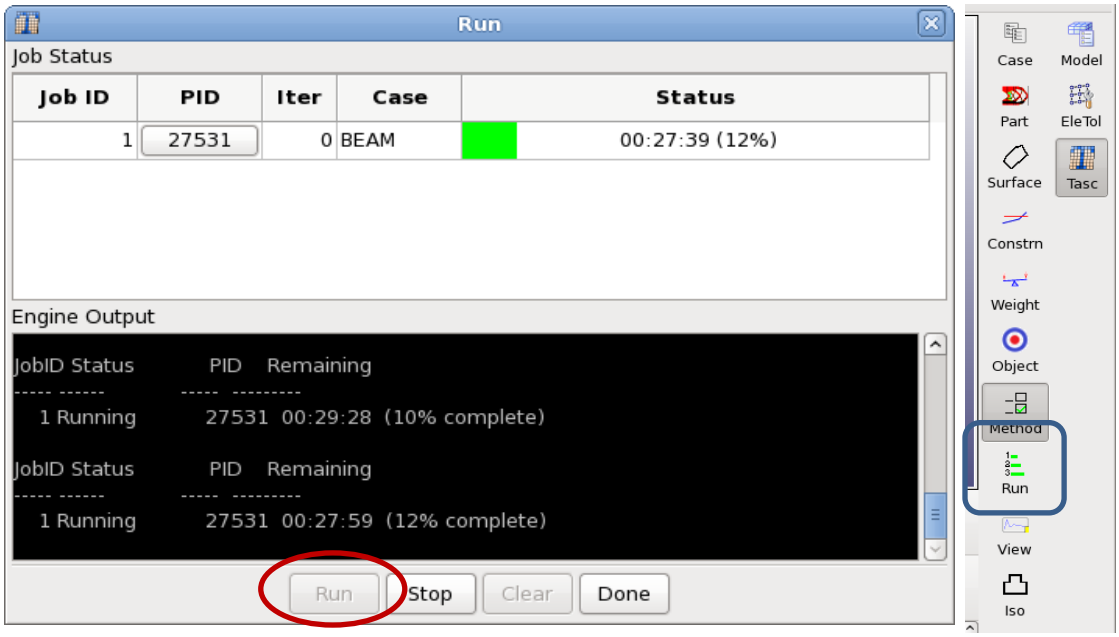

Figure 5-5: Run dialog

5.1.3. Results

The optimization converged after 35 iterations. The results can be visualized using the Topology history and Model plot options available in the *View* dialog, Figure 5-6.

The convergence is quantified using the change in topology, characterized by the normalized mass redistribution as shown in Figure 5-7. It was observed that initially there were significant changes in the topology (up to 17 iterations). Afterwards, small changes were made in the topology.

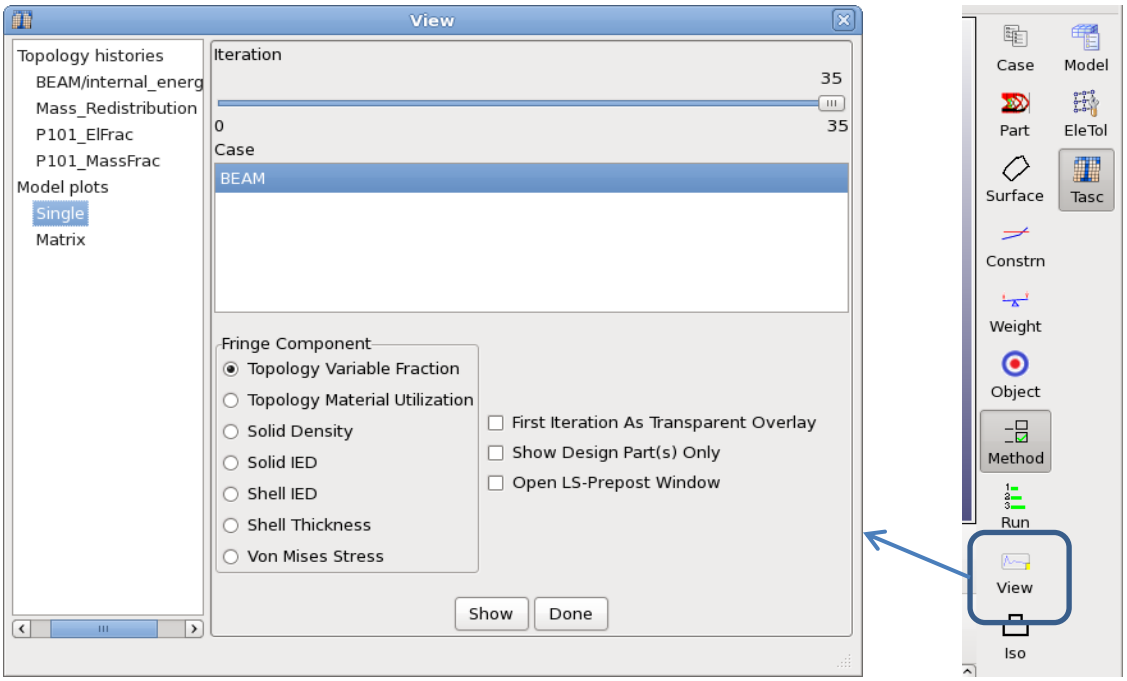

Figure 5-6: View dialog, visualization of optimization results

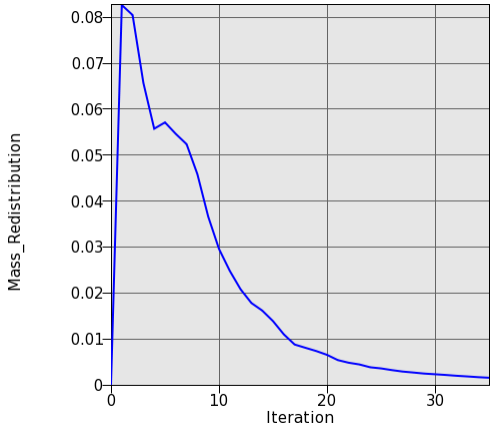

Figure 5-7: Topology history mass redistribution

The final topology is visualized in Figure 5-8. The topologies at different iterations during the evolution process are shown in Figure 5-9. The final topology evolved in a truss-like structure. Many holes were carved to satisfy the mass constraint while reducing the non-uniformity in the distribution of the internal energy density. The final structure was also found to have a reasonably homogenous distribution of the material as was desired. Topologies at different stages of the evolution process show that the main features of the structure were evolved by iteration 14 (row 2, column 1). Further iterations were necessary to bolster the structure by removing the material from relatively non-contributing zones and redistributing it to the desirable sections such as a 0-1 type topology was evolved.

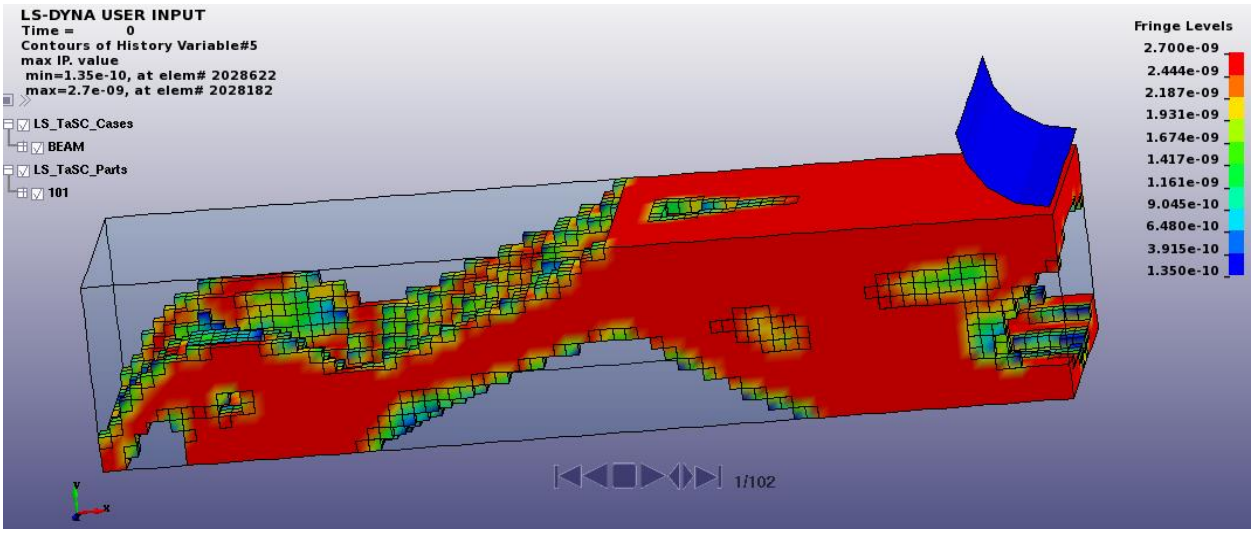

Figure 5-8: Initial and final design, fringe component solid density.

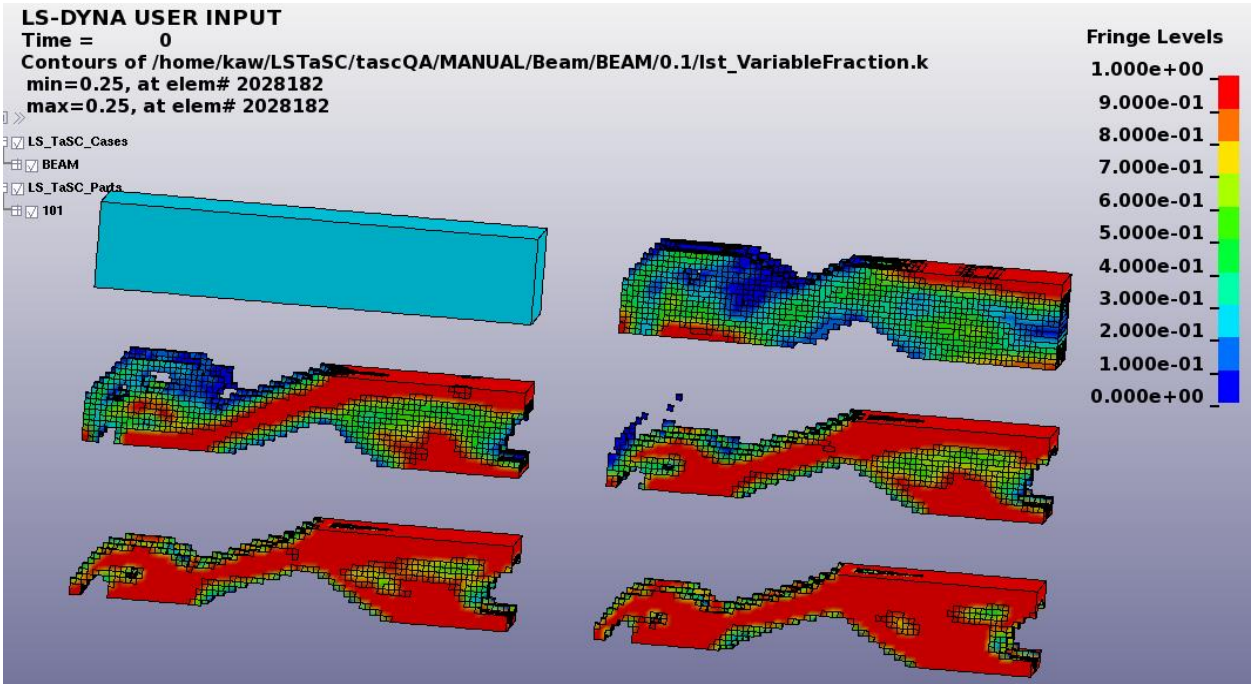

Figure 5-9: Evolution of the geometry shown using density contours.

5.2. Beam using geometry definitions

This example demonstrates

- how to set up a problem with extrusion definitions, and
- how to set up a problem with casting definitions.

The related files are available in MANUAL/Beam_extr_cast.

5.2.1. Problem Description

The same fixed-beam as described in section 5.1.1 is analyzed with an extrusion and casting definitions. The symmetry face is also defined as the extruded face. In the input deck file, the elements on the extrusion face were grouped in a solid set (*SET_SOLID). Two different casting conditions were applied in two separate design runs:

(i) in the first run casting definition was applied in the Z direction, and

(ii) in the second run a two-sided casting definition was applied in the Z direction.

All other parameters were kept the same.

5.2.2. Problem Setup

The project input data is saved to the file *Extr_Cast.lstasc* and *Extr_Cast2.lstasc* as provided in the examples distribution in the directory *Beam_extr_cast*. Additionally to the setup explained in section 5.1.2, the extrusion and casting definition has to be specified, Figure 5-10.

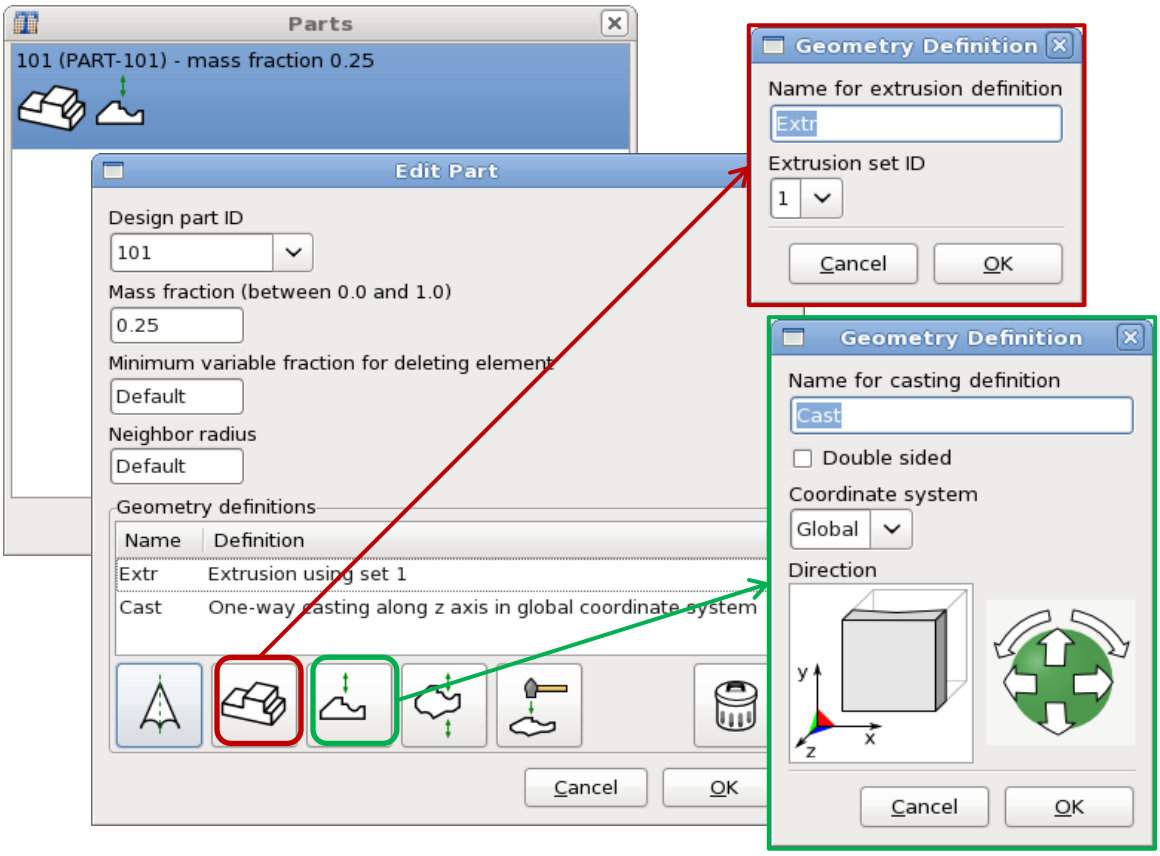

Figure 5-10: Definition of design part with extrusion and casting constraint.

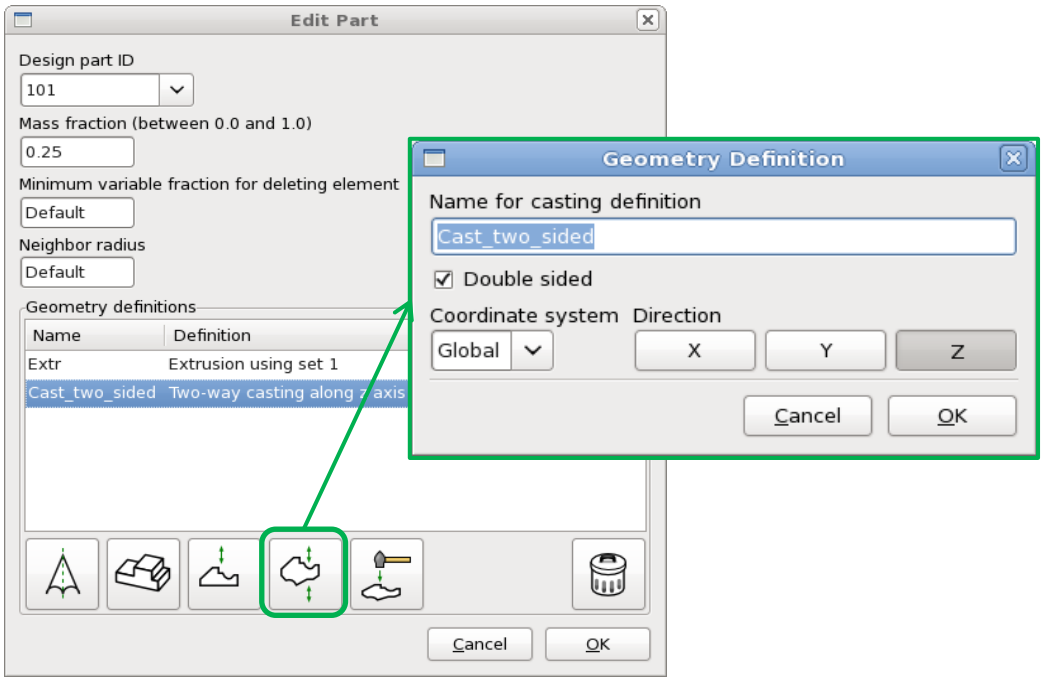

Figure 5-11: Definition of design part with extrusion and 2-sided casting constraint

5.2.3. Results with extrusion and casting

The optimization converged after 36 iterations. Different phases in the evolution are depicted in Figure 5-12. One can see that a lot of material was removed early. The final geometry evolved by considering the geometry definitions was significantly different than the case when no manufacturing constraints were considered. The C-section evolved makes intuitively sense.

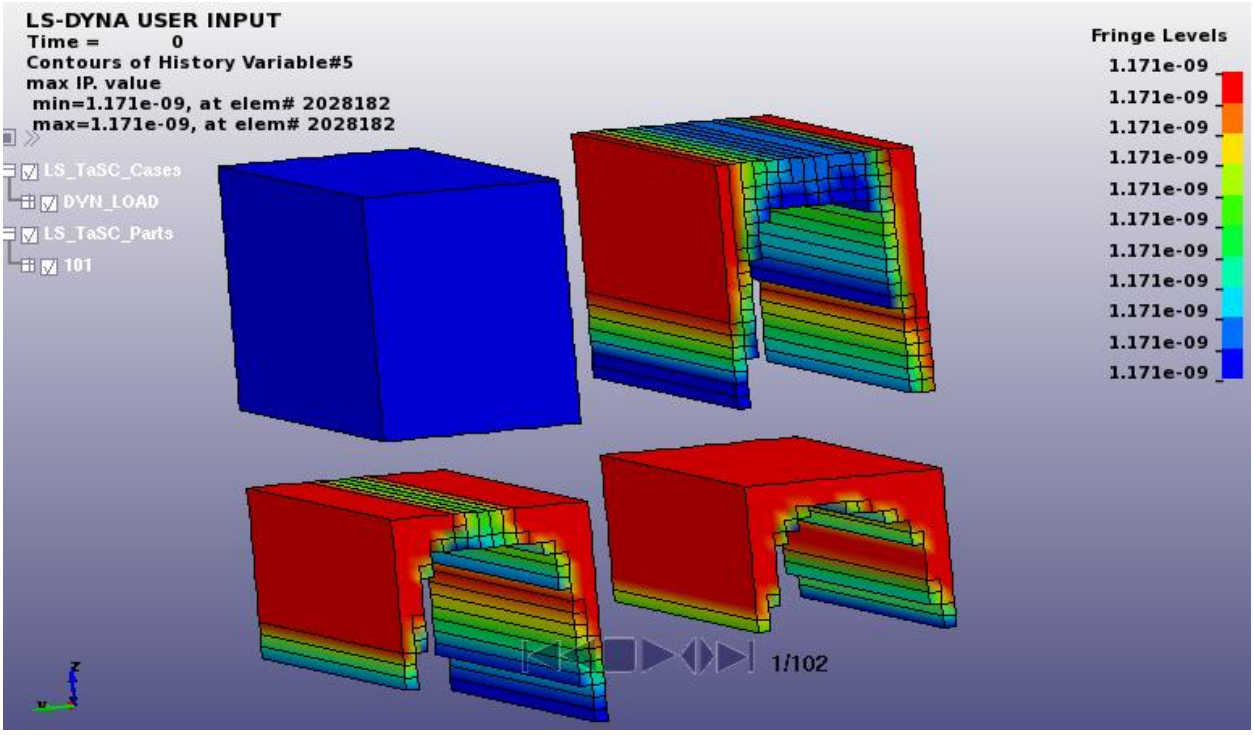

Figure 5-12: Evolution of the beam using extrusion and single-sided casting constraints

5.2.4. Results with extrusion and two-sided casting

Different phases in the evolution are depicted in Figure 5-13. One can see that a lot of material was removed early. The final geometry evolved by considering the geometry definitions was significantly different than the case when no manufacturing constraints were considered. The I-section evolved makes intuitively sense.

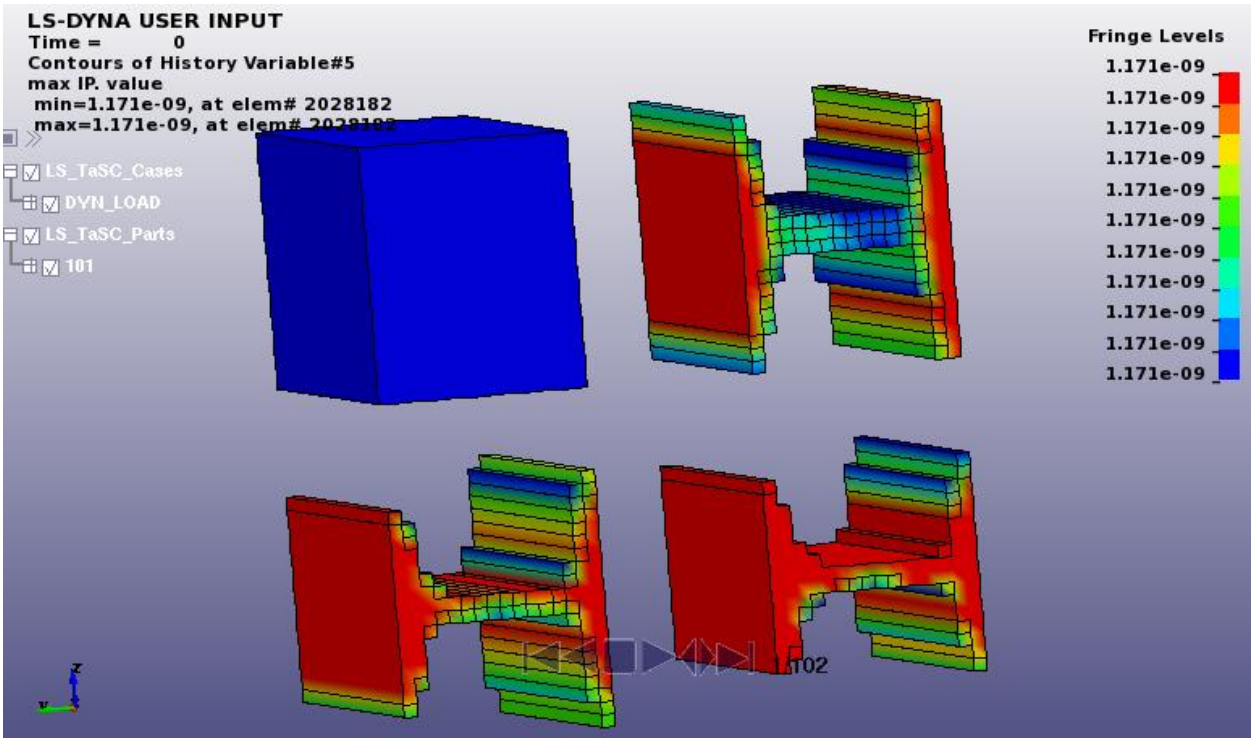

Figure 5-13: Evolution of the beam using extrusion and two-sided casting constraints.

5.3. Force-Displacement Constraints

This example demonstrates

- a topology optimization subject to force and displacement constraints.

The related files are available in MANUAL/Force_Disp_Constraints.

5.3.1. Problem Description

The geometry and loading conditions for the example are shown in Figure 5-14. This is a fixed-fixed beam with a central load. The design part was meshed with $(10mm)^3$ elements. The center load was assigned at the location of the pole hitting the beam.

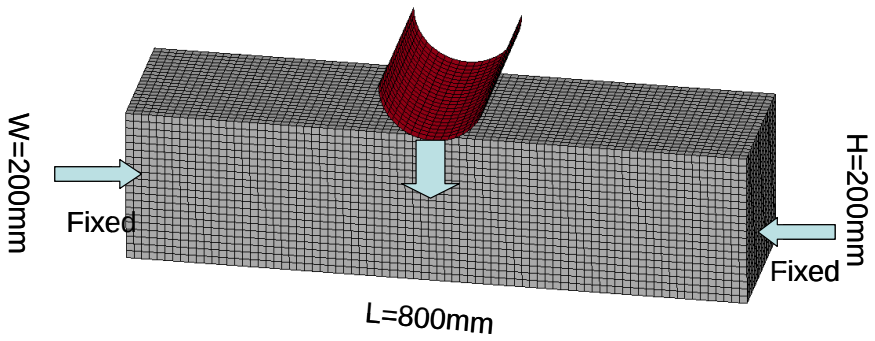

Figure 5-14: The geometry and loading conditions of the multiple constraints example.

5.3.2. Problem Setup

The project input data is saved to the file *force_disp_con.lstasc* as provided in the examples distribution. The definition of the design part is displayed in Figure 5-15. The desired mass fraction for this example was 0.2. The definition of the constraints is displayed in Figure 5-16. The maximum displacement of the indenter was constrained at 34 units and the maximum y-component of the interface force was limited at 1.45e6 units. A maximum of 50 iterations were allowed, Figure 5-17.

In case the user wants to conduct the simulations on a cluster using the LS-DYNA MPP version and a queuing system, *force_disp_con.pbs.lstasc* can be used as an example. An example script named *submit_pbs* to be used as execution command for the PBS queuing system is provided. The setup is displayed in Figure 5-18.

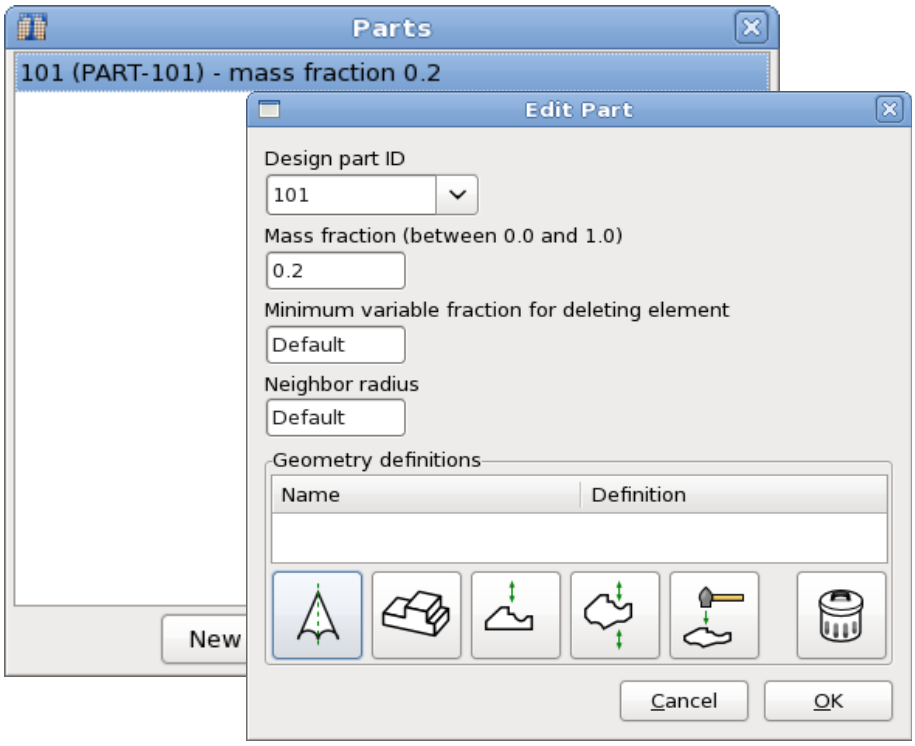

Figure 5-15: Design part definition with desired mass fraction 0.2.

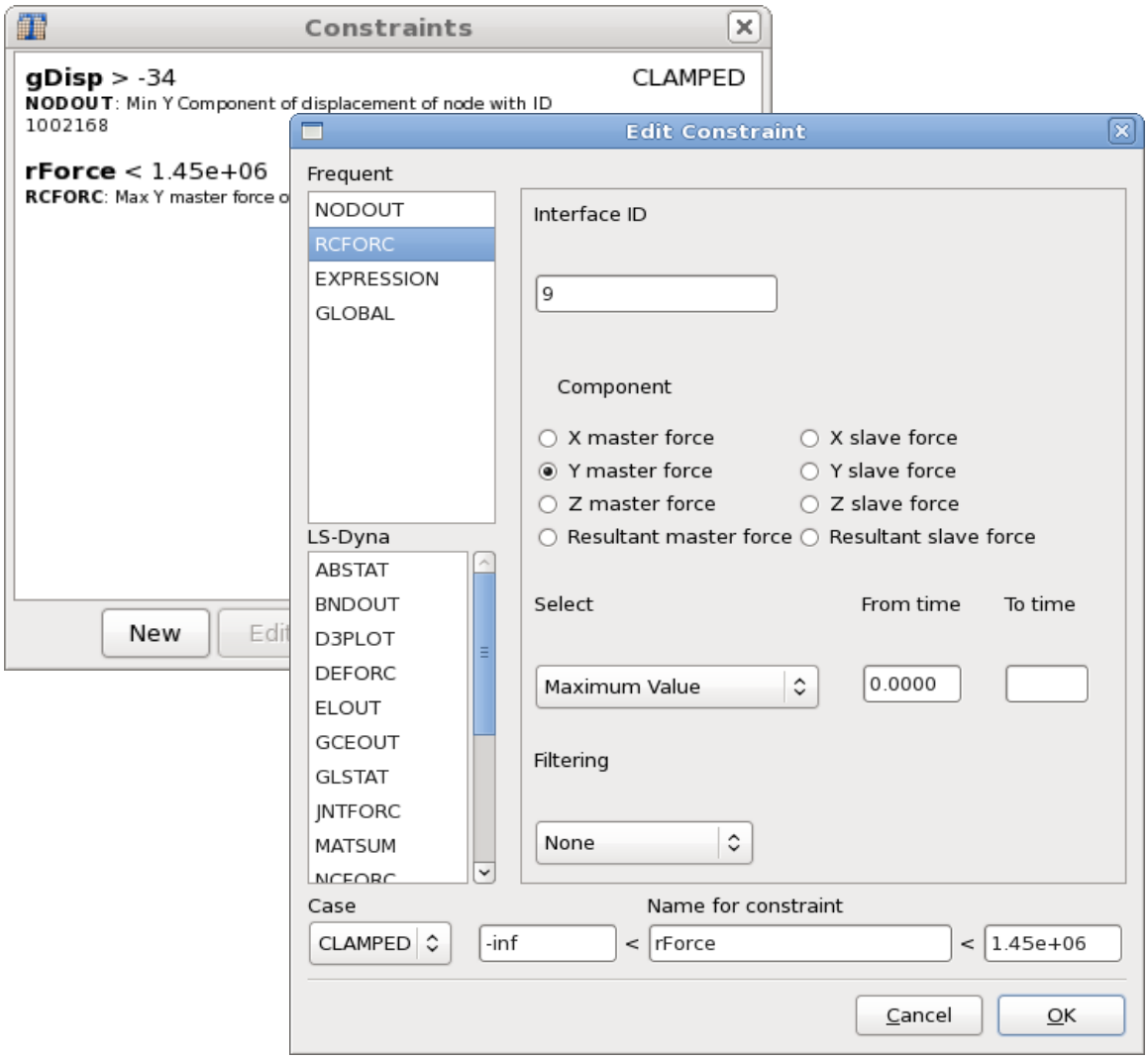

Figure 5-16: Definition of force constraint using the RCFORCE interface.

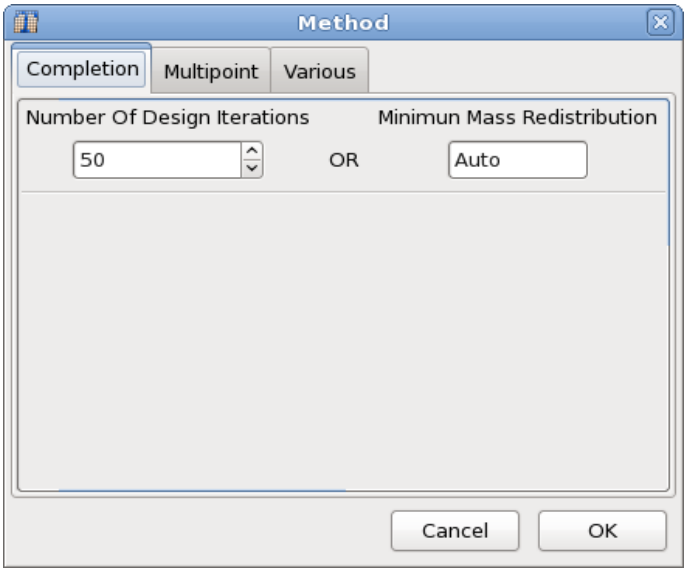

Figure 5-17: Termination Criteria

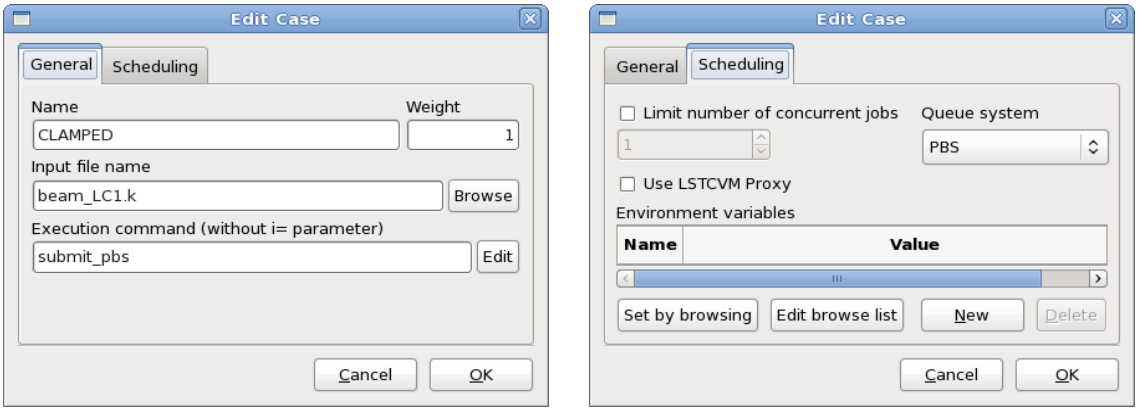

Figure 5-18: Case Setup using a queuing system. The execution command has to be replaced by an appropriate script (left) and the queuing system interface has to be selected (right).

5.3.3. Results

The optimization converged after 36 iterations. The convergence history for the multiple-constraints example is shown in Figure 5-19. There were minimal changes in the geometry after 25 iterations. While there was largely monotonic reduction in the mass redistribution, the constraints were oscillatory in the behavior. The oscillatory behavior of the constraints was due to their conflicting nature where an increase in displacement required an increase in the mass fraction which resulted in higher forces. At optimum, a balance between the two quantities was obtained. It is important to note that the mass fraction for this example was not held constant. Instead, it was automatically adjusted to satisfy the force and displacement constraints though the final mass fraction was fairly close to the desired value.

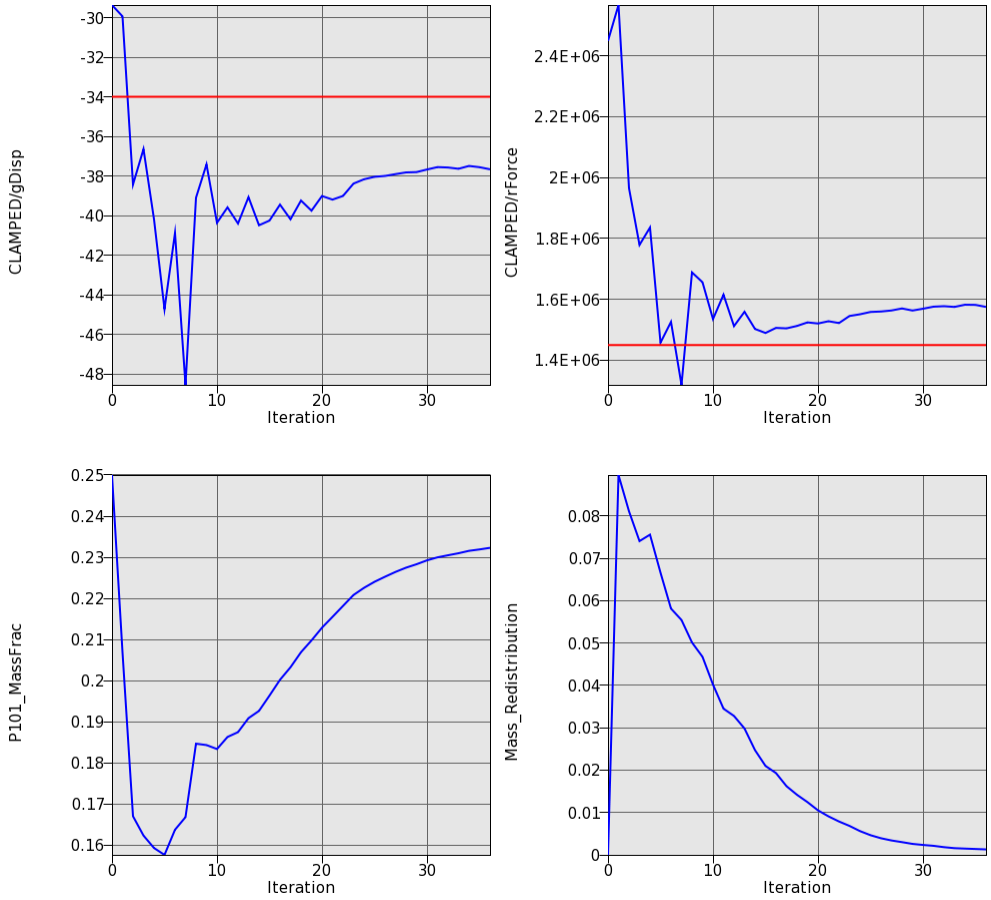

Figure 5-19: Convergence history for the example with multiple constraints. Constraints, mass fraction and mass redistribution.

The evolution of the topology of the clamped beam with multiple constraints is shown in Figure 5-20. The final structure had many cavities and resembled an optimized truss-like structure. The main cavities in the structure were formulated by the 12th iteration and the structure was fully developed in a largely 0-1 type structure by the 24th iteration. Further redistribution of the material refined this structure.

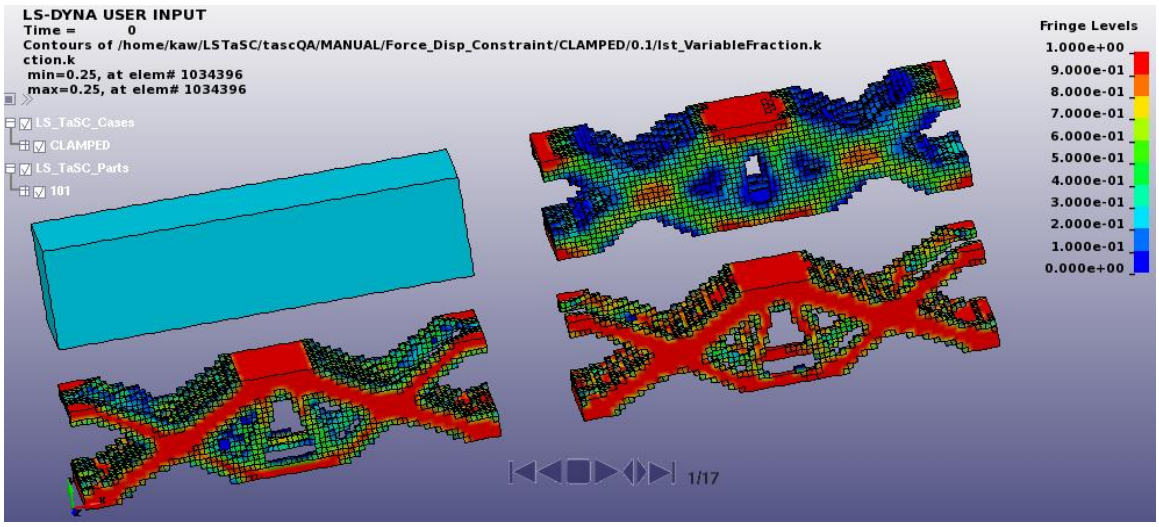

Figure 5-20: Evolution of the geometry for multiple-constrained clamped beam.

5.4. Linear Static Loading

This example demonstrates

- the topology optimization of a statically loaded structure.

The related files are available in MANUAL/Linear.

5.4.1. Problem Description

In this example, a unit load is applied in the center of the structure. The structure was fixed on the bottom. The geometry and loading conditions for the example are shown in Figure 5-21. The design part was meshed with $(1.05\text{mm})^3$ elements such that there were approximately 125,000 elements. The simulations are carried out using the double precision SMP version of LS-DYNA.

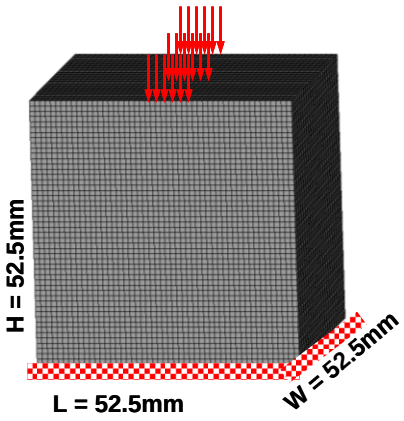

Figure 5-21: The geometry and loading conditions of a statically loaded structure.

5.4.2. Problem Setup

The definition of the interface to LS-DYNA is displayed in Figure 5-22. The name of the load case as well as the LS-DYNA input file *LinearStructure.dyn* and command has to be specified. Part 102 is the design part with the desired mass fraction of 0.3, Figure 5-23. A maximum of 30 iterations are used to find the optimal topology, which is the default, Figure 5-24.

The project input data is saved to the file *lst_project.lstasc* as provided in the examples distribution directory Linear.

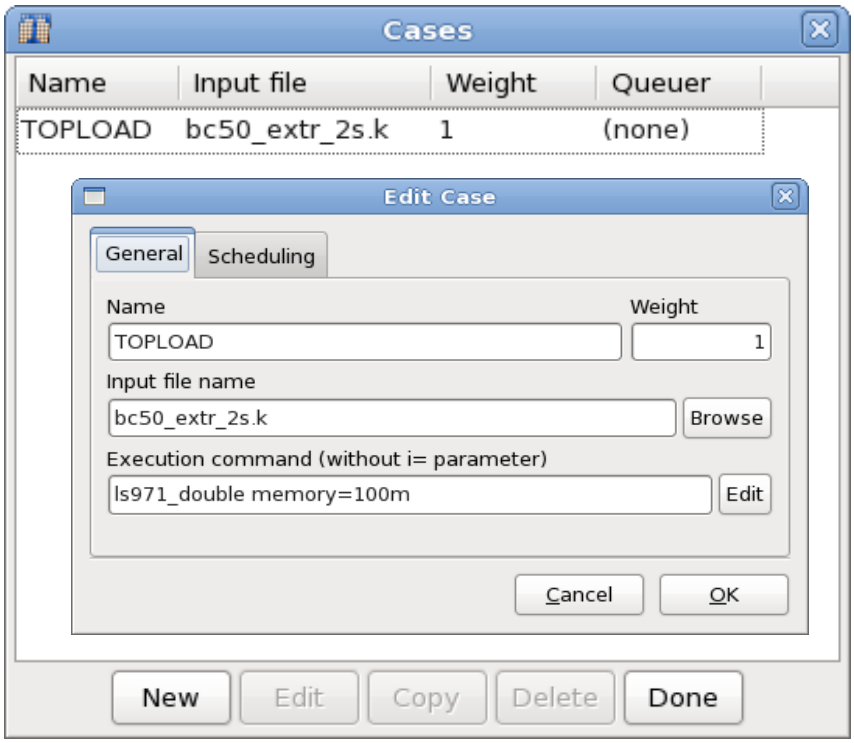

Figure 5-22: Definition of Case TOPLOAD

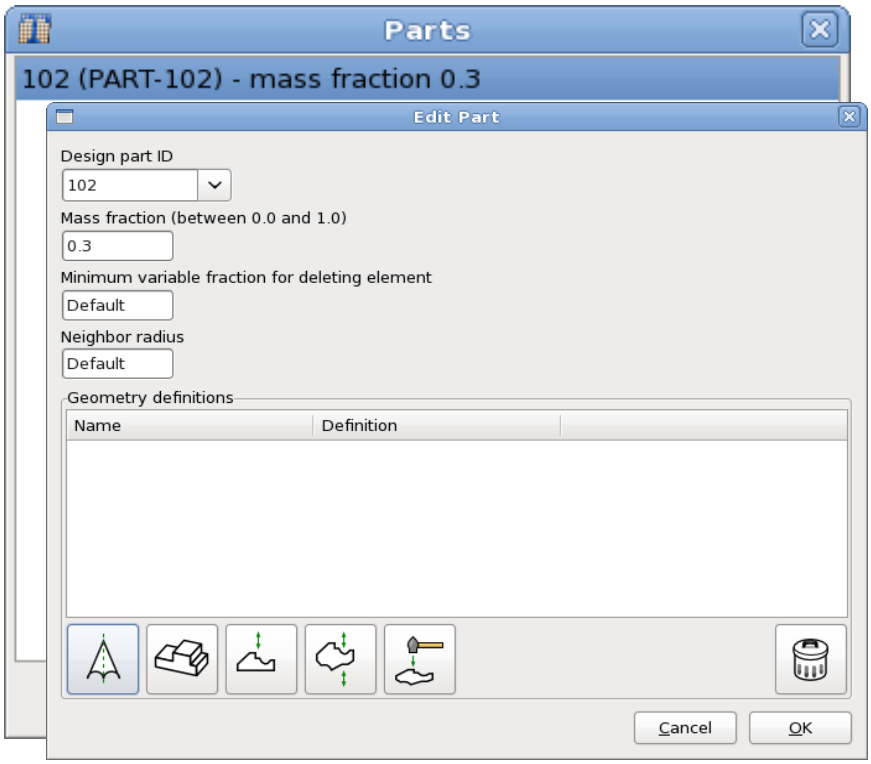

Figure 5-23: Definition of Design Part 102 and mass fraction 0.3

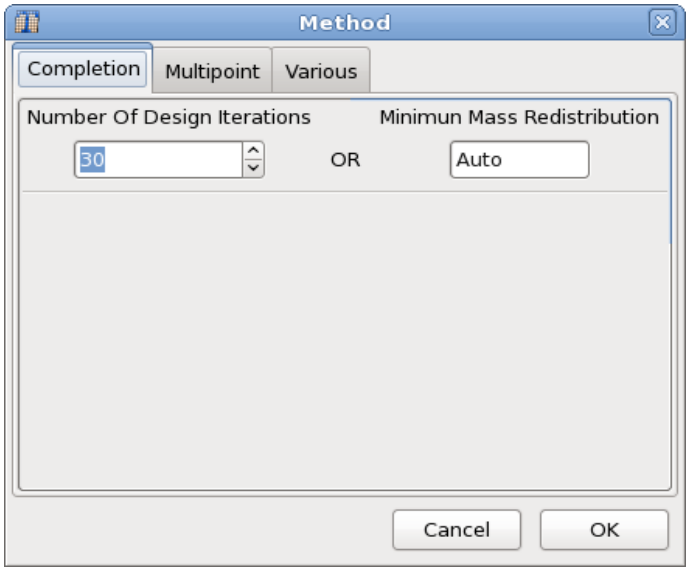

Figure 5-24: Termination Criteria

5.4.3. Results

The simulation converged after 21 iterations. The convergence history for the statically loaded structure topology optimization example is shown in Figure 5-25. As observed before, monotonic reduction in the change in topology was observed.

The initial and final structures are shown in Figure 5-26. The final structure evolved in a column-like structure with wider supports on the faces. The shape of the structure also resembled the best-stress design.

The evolution of the topology under the static loading conditions is shown in Figure 5-27. While the final form of the structure was largely evolved by 13th iteration (first structure in the second row), the material was re-distributed to remove the low-density elements that were not contributing sufficiently to support the load and obtain a homogenous material distribution such that the simulation converged after 21 iterations.

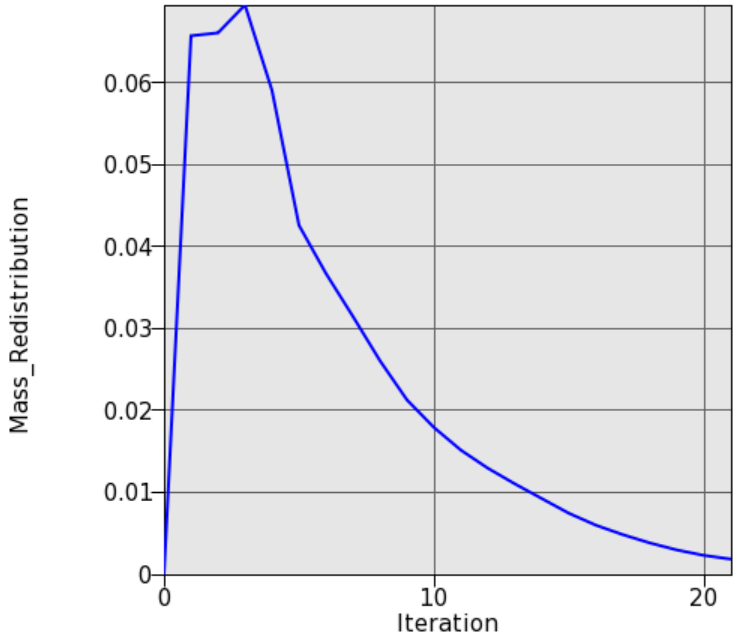

Figure 5-25: Mass Redistribution; Convergence history for linear-static example.

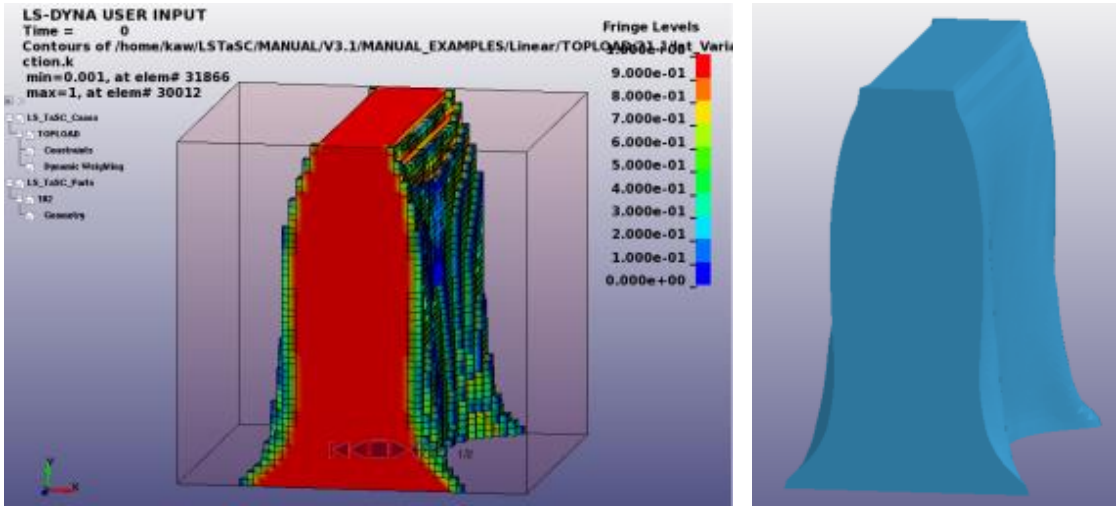

Figure 5-26: Initial and final density contours (left) and iso-surface of final design (right).

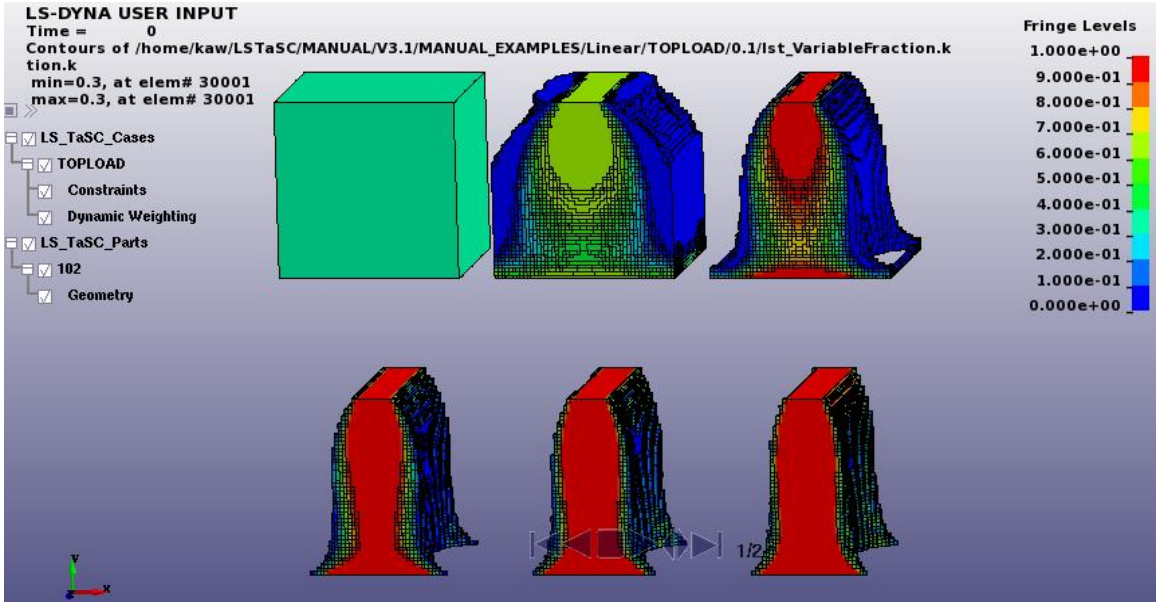

Figure 5-27: Evolution of the geometry for statically loaded structure.

5.5. Shell Example

This example demonstrates

- the optimization of a shell structure.

The related files are available in MANUAL/Shell.

5.5.1. Problem Description

The geometry and loading conditions for the example are shown in Figure 5-28.

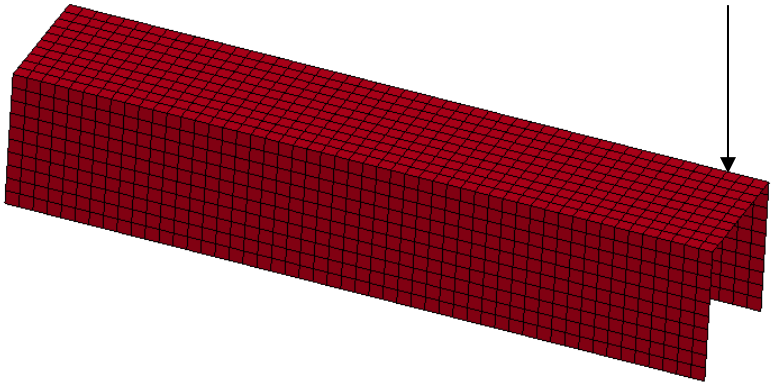

Figure 5-28: The geometry and loading conditions of the shell example. The left side is built-in, while a downward load is applied to the right, back corner.

5.5.2. Problem Setup

The project input data is saved to the file *Shell.lstasc* as provided in the examples distribution. The definition of the load case is displayed in Figure 5-29. The input file name and the LS-DYNA execution command has to be specified. Figure 5-30 shows the definition of the design part. The design part ID is 1 with a desired mass fraction of 0.3. The variable fraction for deleting elements was increased to 0.05. The convergence tolerance was set to 0.01, Figure 5-31.

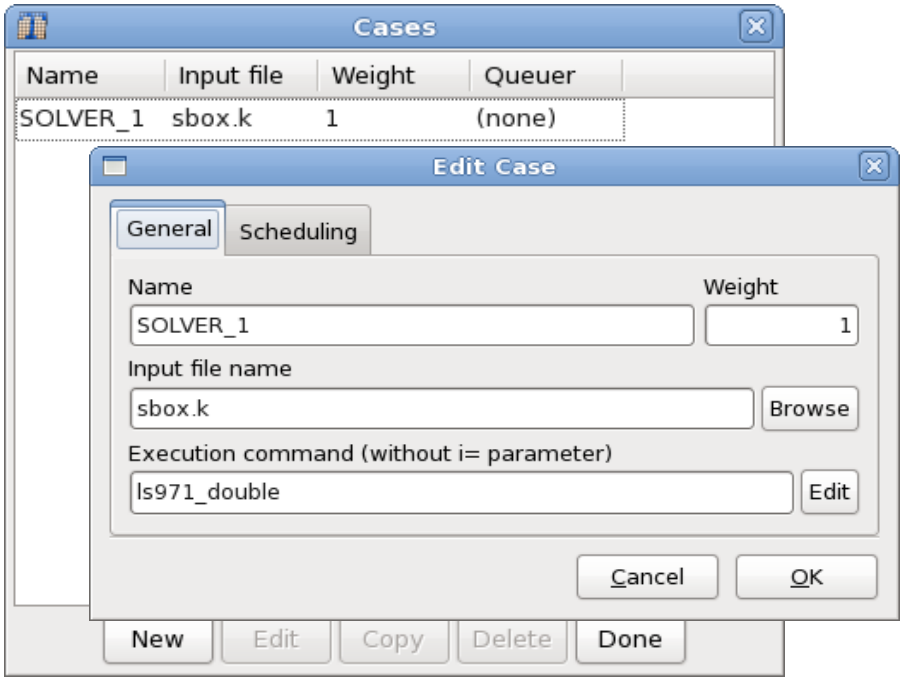

Figure 5-29: Definition of load case

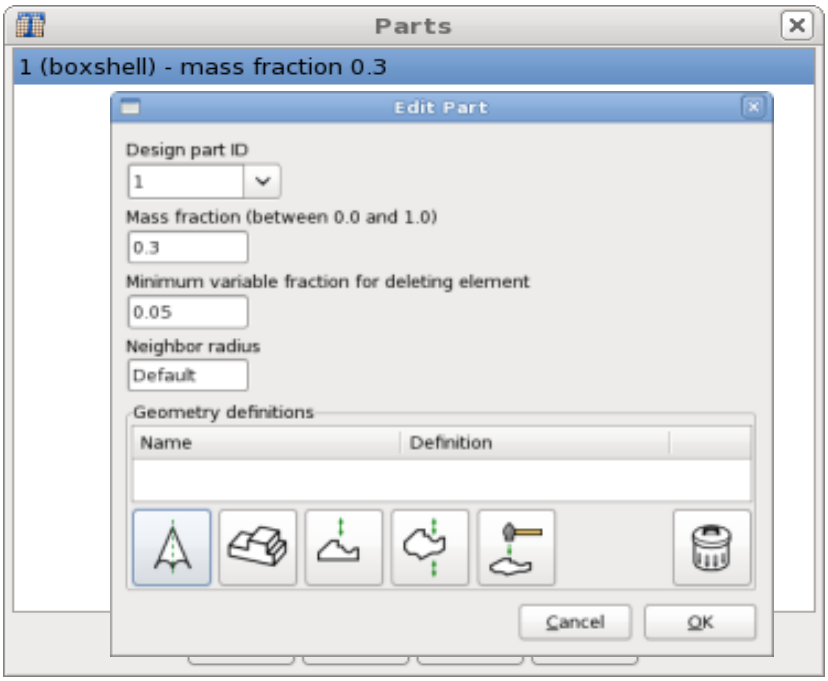

Figure 5-30: Definition of design part and mass fraction

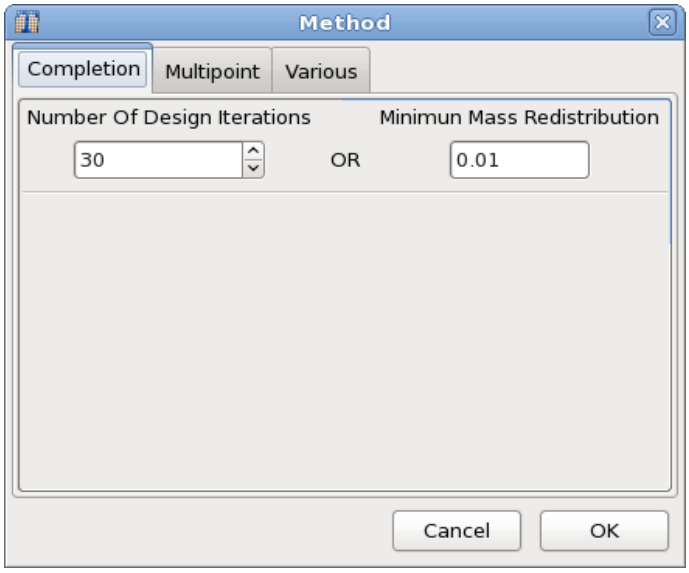

Figure 5-31: Termination Criteria dialog; the convergence tolerance was increased to 0.01

5.5.3. Results

The simulation converged after 15 iterations. The convergence history for the shell example is shown in Figure 5-32. There was largely monotonic reduction in the mass redistribution.

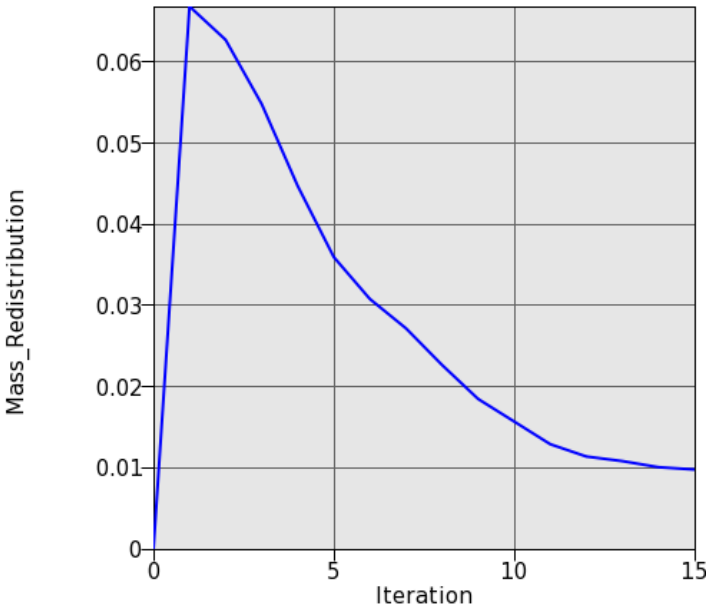

Figure 5-32: Mass Redistribution - Convergence history for the shell example.

The final design is shown in Figure 5-33. The final structure had many cutouts and resembled an optimized truss-like structure.

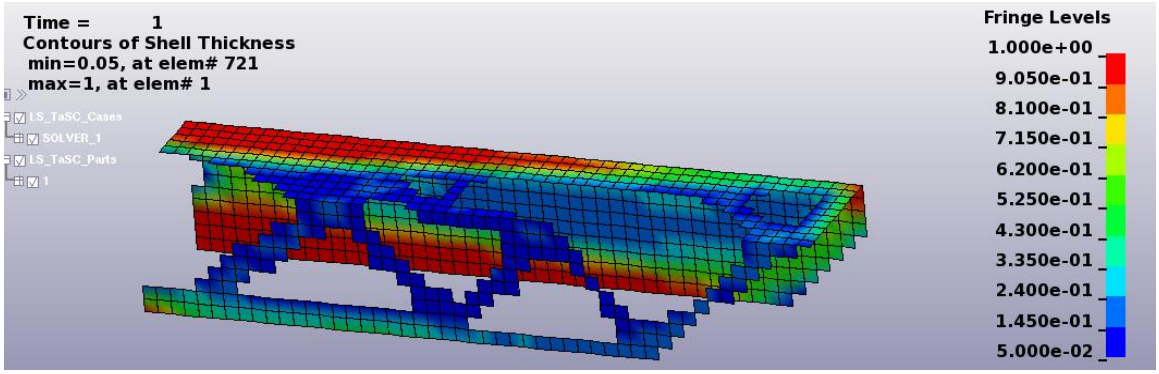

Figure 5-33: Shell thickness fringed on final geometry for the shell problem.

5.6. Optimization of a Bottle Opener considering Multiple Load Cases

This example demonstrates

- the optimization of multiple load cases, and
- the definition of extrusion constraints.

The related files are available in MANUAL/Opener

5.6.1. Problem Description

The geometry and loading conditions for the example are shown in Figure 5-34. Two loadcases are considered, **LC1** which opens the bottle and **LC2** which causes bending on the bottle opener. The design part was meshed with $(0.4mm)^3$ elements.

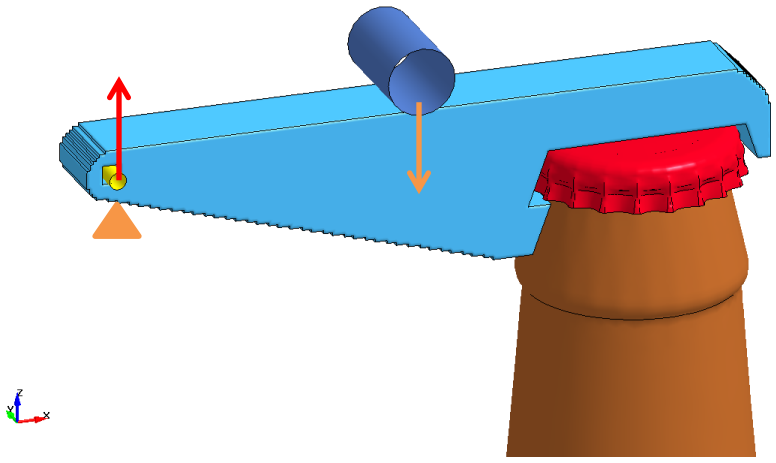

Figure 5-34: The geometry and loading conditions of the bottle opener example.

5.6.2. Results

The optimization stops after 50 iterations due to the limit of iteration. But the history for the mass fraction and element fraction shows a good convergence as shown in Figure 5-35. Figure 5-36 shows four designs from Iteration 0 to Iteration 50. The principal stress under loading is shown in Figure 5-37.

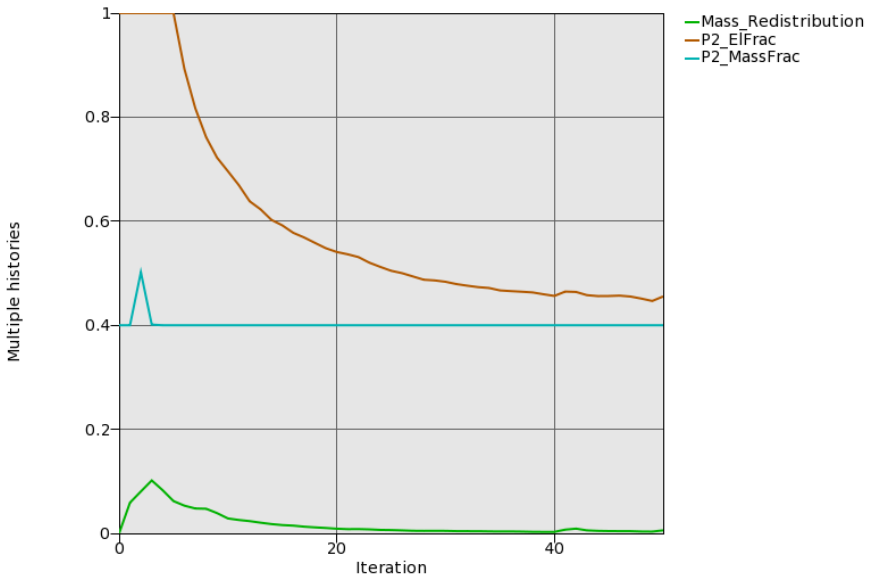

Figure 5-35: Convergence history – Mass Fraction, Element Fraction and Mass Redistribution

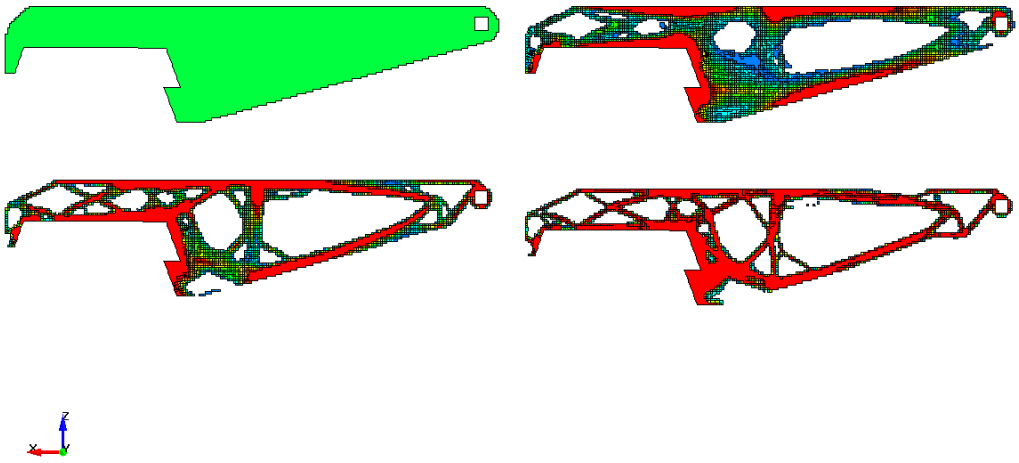

Figure 5-36: Design space to final design

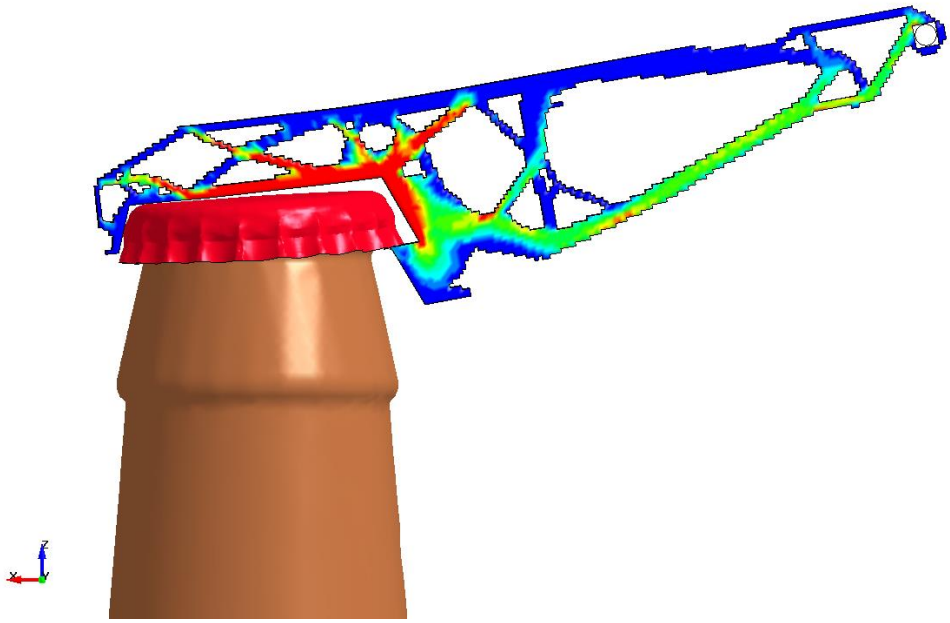

Figure 5-37: Principal stress

5.7. Optimization of Multiple Load Cases

This example demonstrates

- optimization of multiple load cases,
- a symmetry geometry definition,
- constraints,
- constant and dynamic weighting of load cases and
- constrained optimization using multi-point method.

The related files are available in MANUAL/MLC.

5.7.1. Problem Description

The geometry and loading conditions for the example are shown in Figure 5-38. This is a fixed-fixed beam with three loads. The three load cases were identified according to the location of the pole hitting the beam. The design part was meshed with $(10mm)^3$ elements.

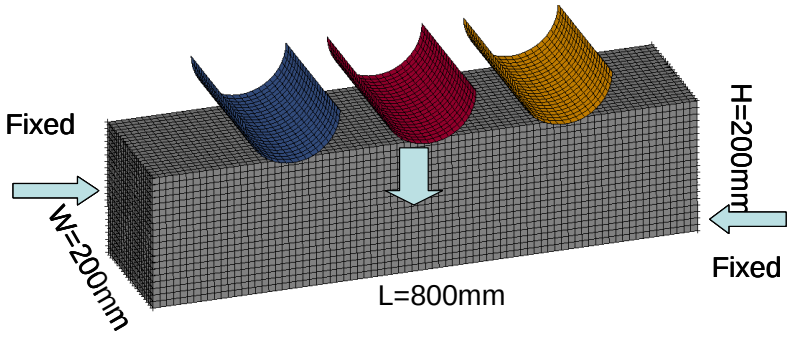

Figure 5-38: The geometry and loading conditions of the multiple load case example.

5.7.2. Problem Setup

Figure 5-39 displays the design part definition. The problem is symmetric, so only two load cases are therefore used and symmetry is defined, Figure 5-39. The desired mass fraction for this example is 0.3. The maximal displacements at the centers of impact for both load cases are constrained to be less than 110, Figure 5-40. A maximum of 50 iterations are allowed. All simulations of both load cases of an iteration are run simultaneously.

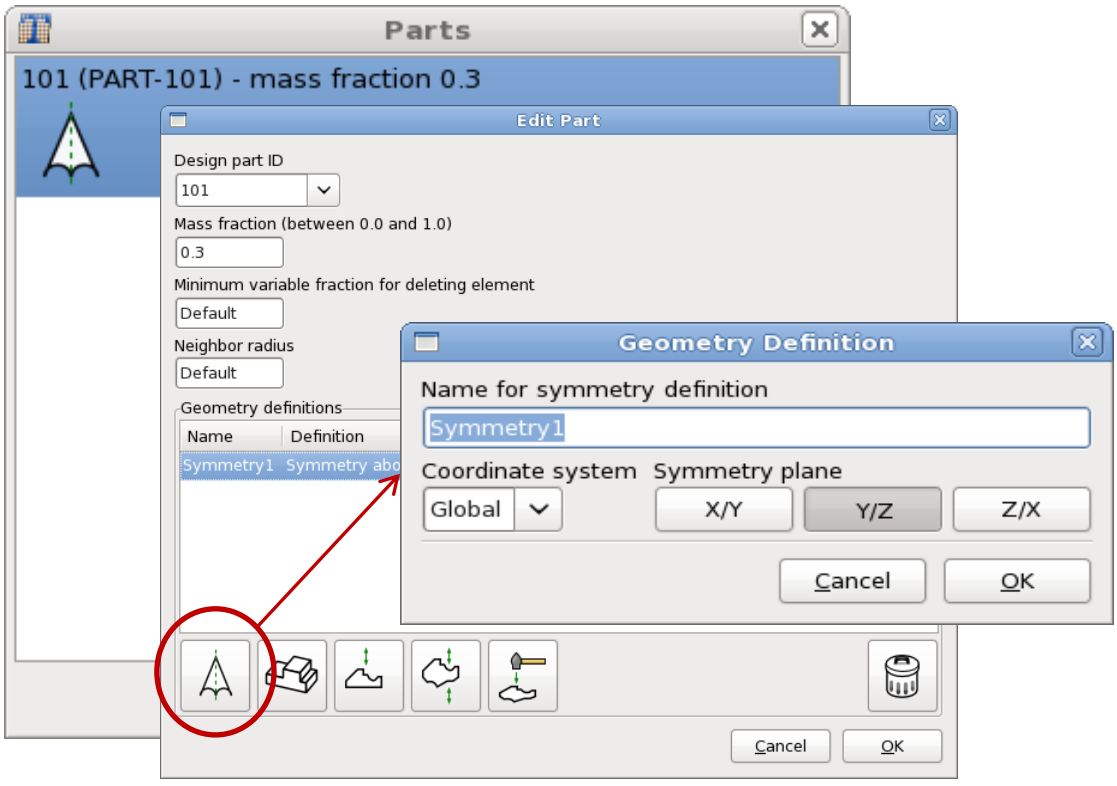

Figure 5-39: Definition of design part with symmetry condition and mass fraction 0.3.

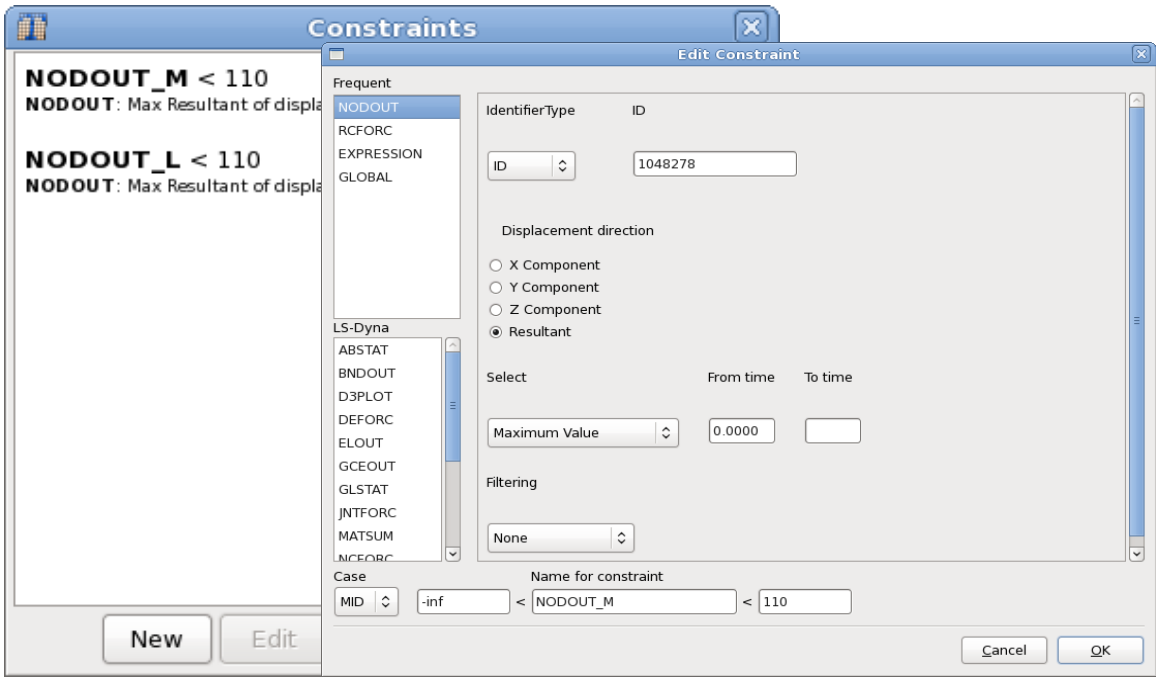

Figure 5-40: Definition of constraints – displacement of centers of impact < 110.

Three approaches to solve this optimization problem are executed. The problem is analyzed using constant (*mlc_constantweight.lstasc*) and dynamic weighting (*mlc_dynweight.lstasc*) of the load cases as well as the multi-point method (*mlc_multipoint.lstasc*).

Constant weights for each load case are defined in the *Case* definition dialog, Figure 5-41. Dynamic weighting can be activated in the *Weight* dialog, Figure 5-42. The multi-point method can be switched on in the *Method* dialog *Multipoint* tab, Figure 5-43. The response surface methodology (RSM) is used to optimize the global variables, the mass fraction and the load case weights. Note that the load case weight move limit was increased to get faster convergence.

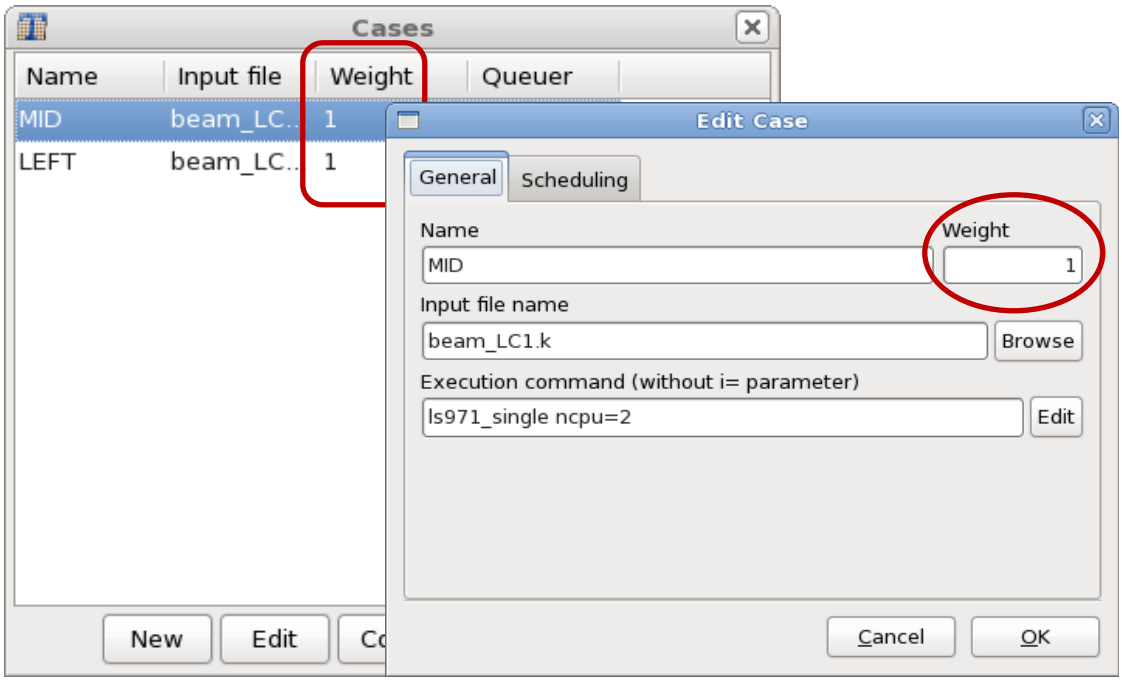

Figure 5-41: Definition of constant weights for each load case

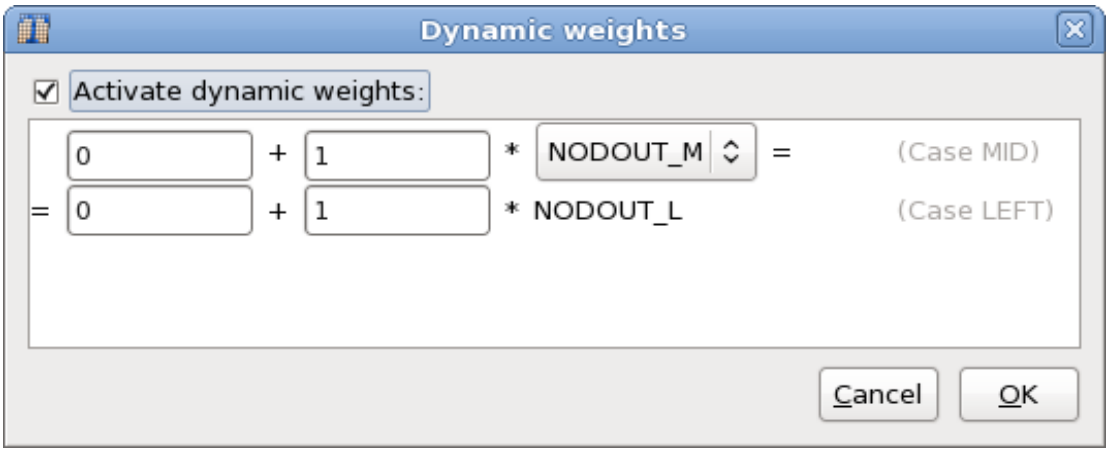

Figure 5-42: Definition of dynamic weights

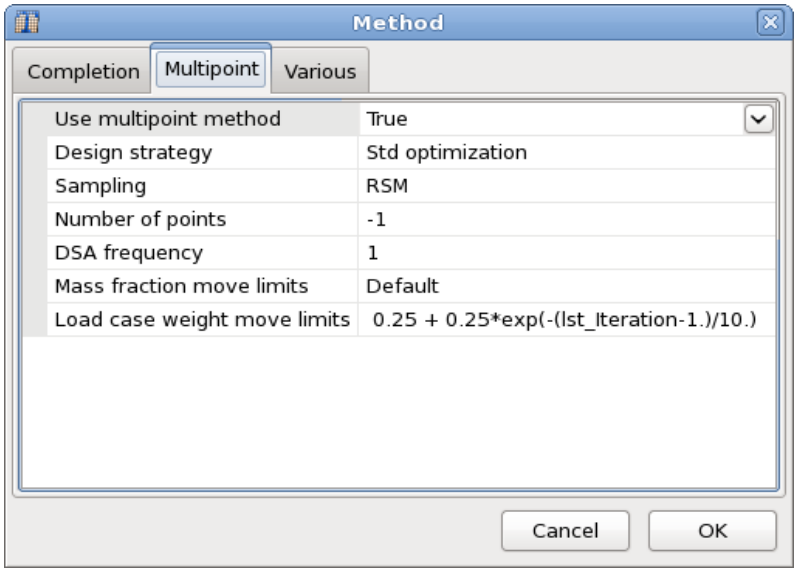

Figure 5-43: Settings for multipoint method

5.7.3. Results with constant weights

The optimization converges after 27 iterations. The results are as shown in Figure 5-44 to Figure 5-46. The resulting structure is much stronger in supporting the side loads than the center load with the resulting poor outcome for the constraint values as shown in Figure 5-44.

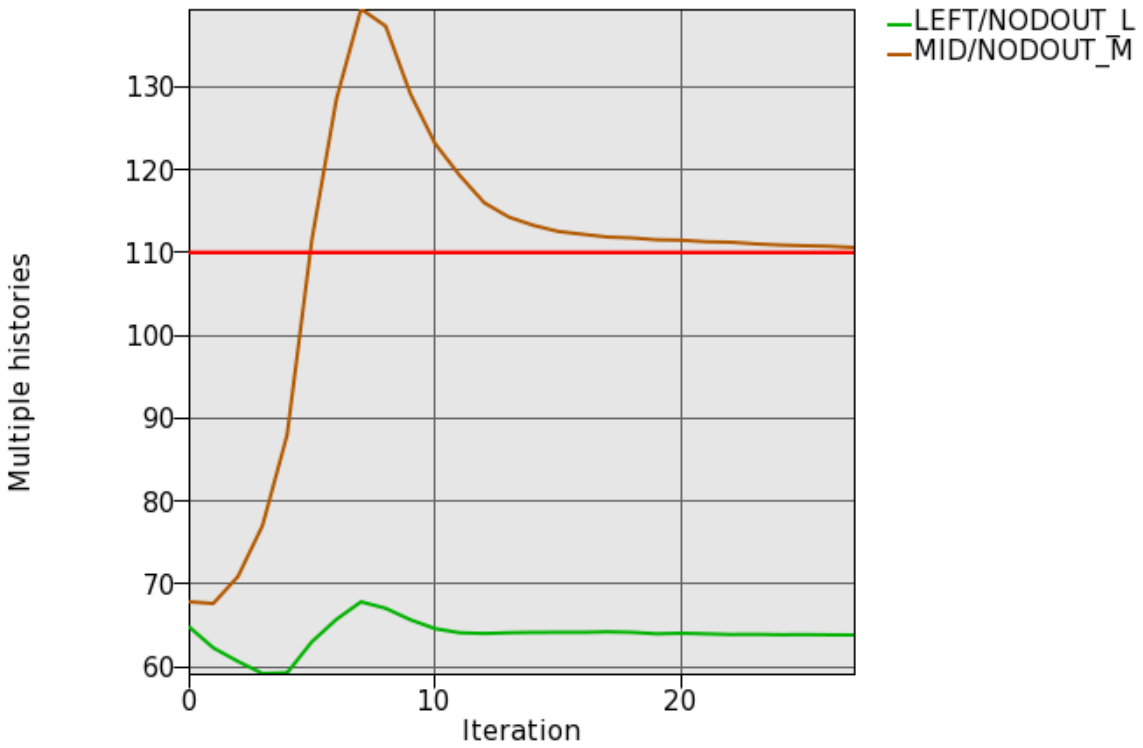

Figure 5-44: Constraint convergence history for multiple-load case example with constant weights.

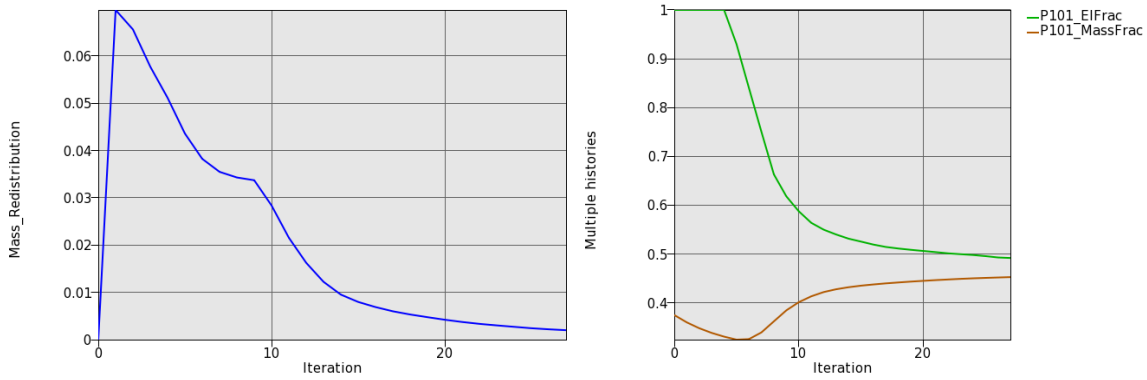

Figure 5-45: Various histories of the load case weight for multiple-load case example using with constant weights: mass redistribution (left), the fraction of elements kept, and the mass fraction (right).

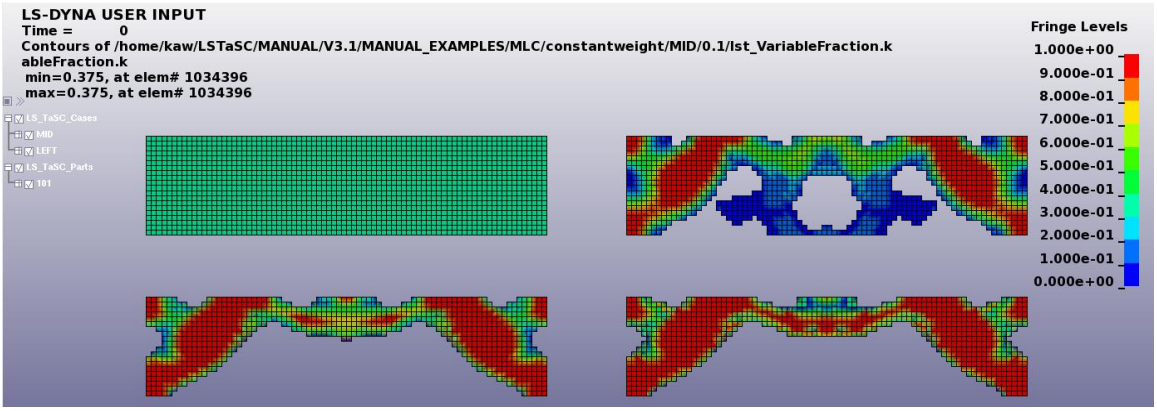

Figure 5-46: Evolution of the geometry for multiple-load case structure using constant weights, colored by topology variable fraction.

5.7.4. Results with dynamic weighing

The optimization converged after 48 iterations. The convergence history for the multiple-load example solved with dynamic weights is shown in Figure 5-47. Results are much improved by the dynamic weighting. The constraints are reasonably close to the bound as shown in Figure 5-47 due to the load case weighting computed also shown.

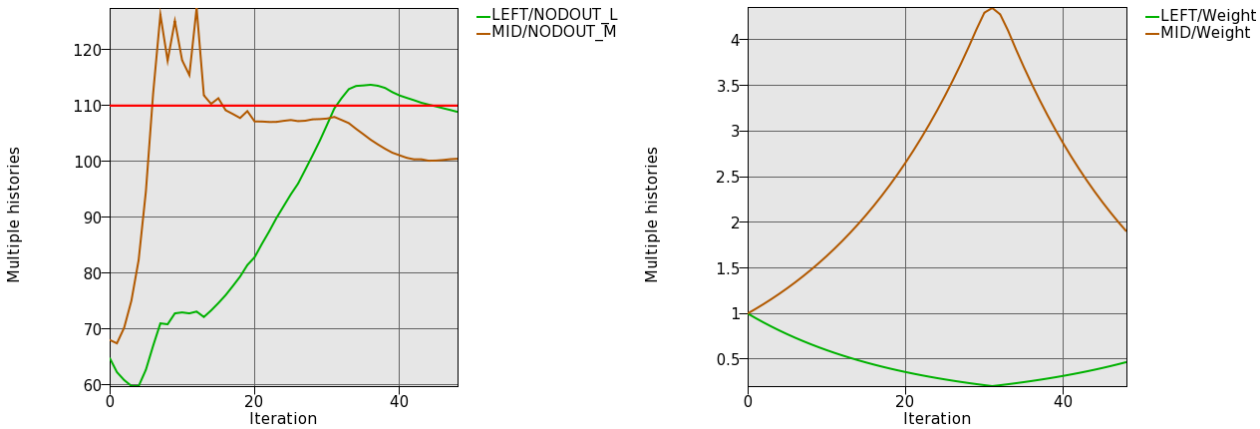

Figure 5-47: Constraint convergence history for multiple-load case example using dynamic weighting is shown on the left. Note the improvement with respect to not using dynamic weighting. The corresponding weight factors are shown on the right.

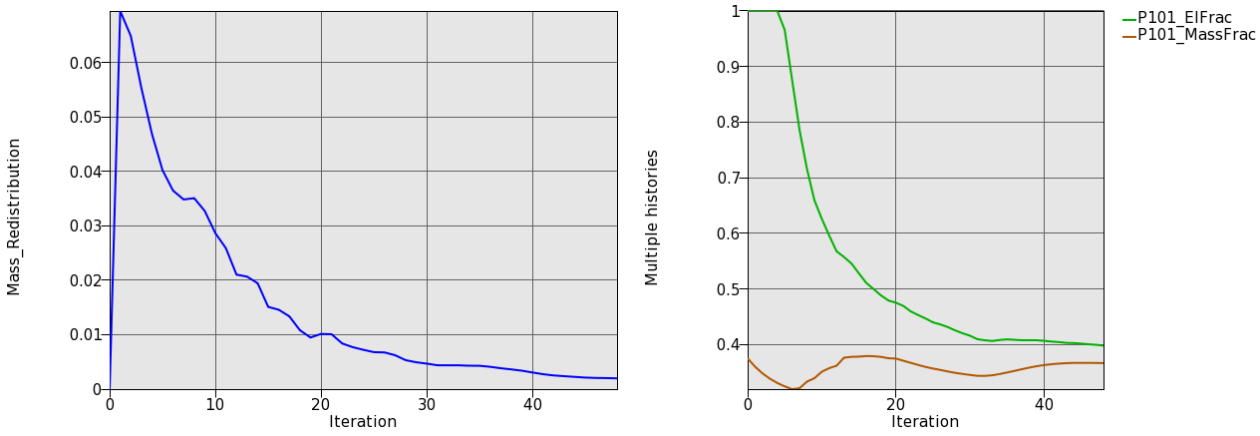

Figure 5-48: Various histories of the load case weight for multiple-load case example using dynamic weighting: mass redistribution, the fraction of elements kept, and the mass fraction.

The evolution of the topology under multiple loading conditions is shown in Figure 5-49. The final structure evolved in a tabular structure with the two cross-members as legs. The structure had more material in the center section due to the high importance assigned to the center weight. There were many cavities in the structure such that the final structure could be considered equivalent to a truss-like structure as one would expect.

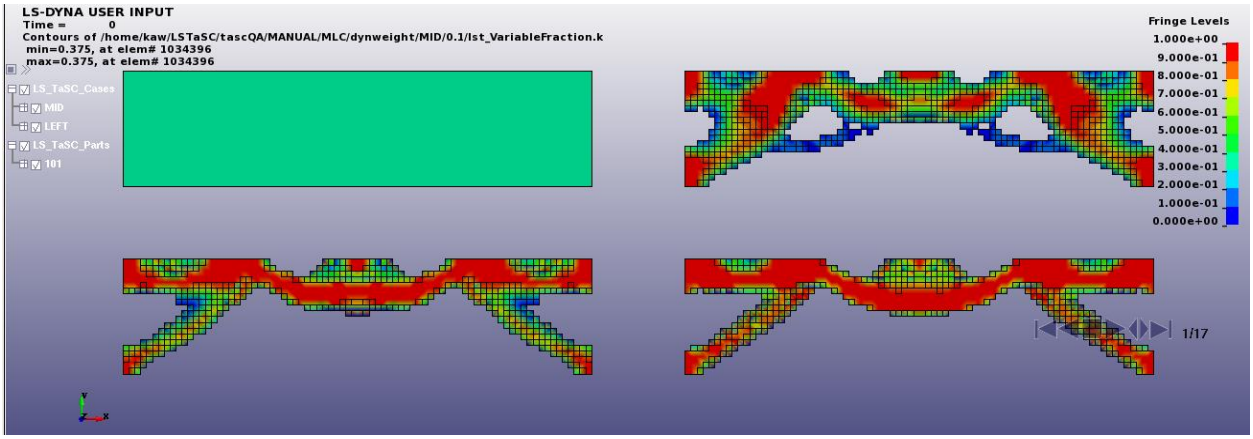

Figure 5-49: Evolution of the geometry for multiple-load case structure using dynamic scaling of the weights. The design is improved with respect to not using dynamic weighting by strengthening the portion of the structure carrying the center load.

5.7.5. Results using multi-point optimization

The optimization converged after 43 iterations, 4 simulations were performed per iteration. The results are as shown in Figure 5-50 to Figure 5-52. The displacement of the offset load case was made bigger than the center load case by decreasing the weight of that load case. The load case weight values usually overshoots, which why the displacement values are not exactly the same. The problem can be set up using different constraints to prevent this.

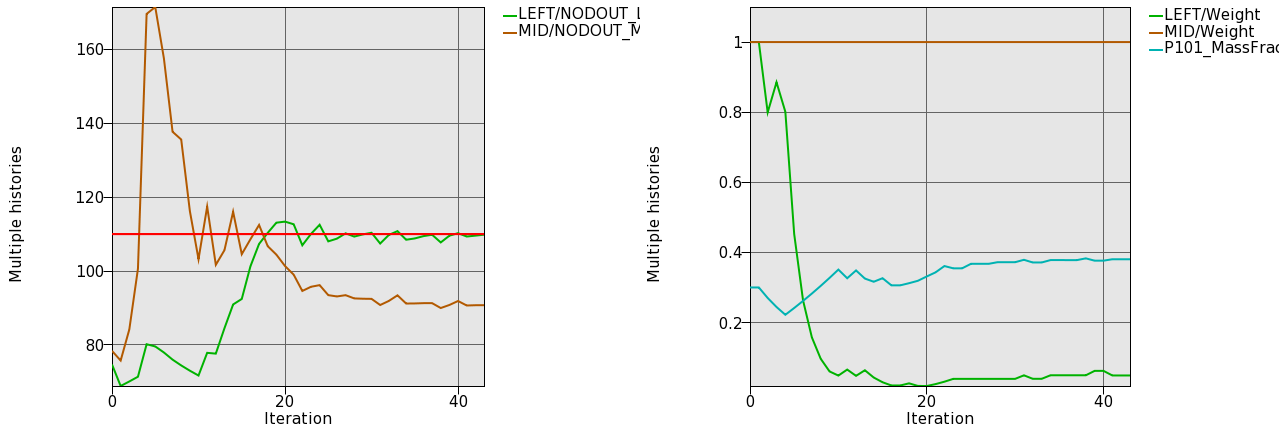

Figure 5-50: Constraint convergence history (left) and global variables (right) for constrained optimization with multiple load cases.

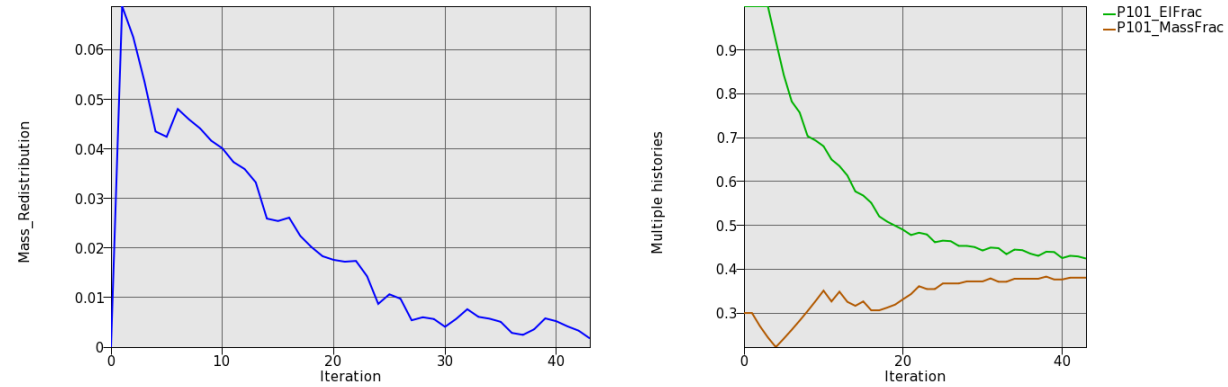

Figure 5-51: Mass redistribution and element and mass fraction.

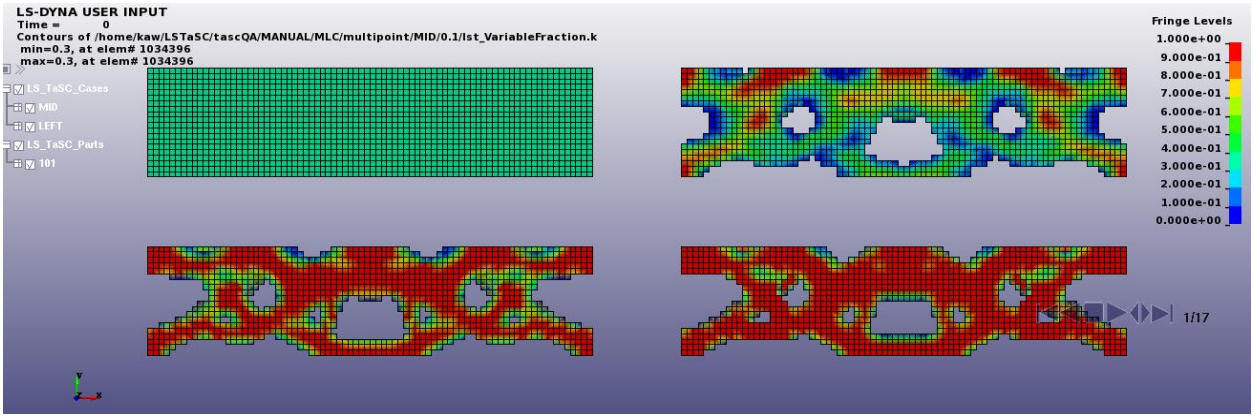

Figure 5-52: Evolution of the geometry for multiple-load case structure using multi-point method

5.8. Multiple Part Design and Mathematical Expressions

This example demonstrates:

- Design considering multiple parts,
- Constraints depended on multiple parts,
- Using the energy absorption of a part as a constraint,
- Defining a constraint as a mathematical expression of other constraints, and
- The multi-point methodology.

The related files are available in MANUAL/MultiPart_Expressions.

5.8.1. Problem Description

The geometry and loading conditions for the example are shown in Figure 5-53. The structure represents the floor plan of a vehicle. The load case consists of an applied displacement to the front of the vehicle while the rear of the passenger compartment is fully supported. The two design parts are part 3 which is the design of the engine compartment and part 5 which is the design of the passenger compartment. The problem is set up such that the intrusion into the passenger compartment is strongly dependent on the mass fraction of part 3 relative to part 5.

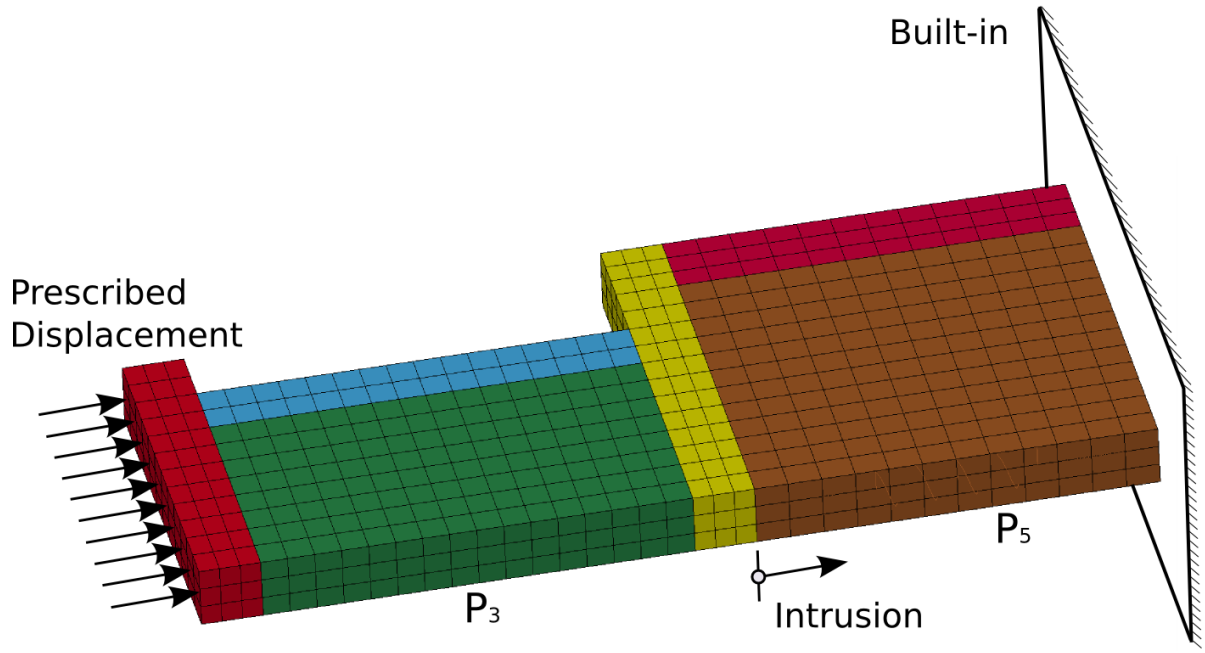

Figure 5-53: The geometry and loading conditions of the multiple part example.

5.8.2. Problem Setup

The initial mass fraction for both part parts is 0.3, Figure 5-54.

The definition of the constraints is displayed in Figure 5-55. The intrusion is calculated as the difference of two displacements and is constrained to be less than 0.003. The displacement results are extracted from the *nodout* database and named XDISP_N987 and XDISP_N1523 after the relevant node numbers. The mass fraction of part 3 is also used as a constraint with a lower bound of 0.1. The constraints therefore are:

MF_3 > 0.1 and

XDISP_N987 - XDISP_N1523 > 0.003

with the displacements being functions of the part 3 mass fraction, the part 5 mass fraction, and the topology at that iteration. The sum of the mass fractions is also defined for extraction for evaluation of the optimization results.

In the second approach to solve the problem the energy absorption of the design parts is constrained to be more than 800 units, with most of the energy absorbed by part 3. The internal energy of the design parts are extracted from the *matsum* database and named ENER_P3 and ENER_P5 after the relevant part IDs. The engine compartment is required to absorb 1.5 times the energy of the passenger compartment, for which, the mathematical expression of "ENER_RATIO = ENER_P3 / ENER_P5" is defined. Also, the energy absorption of the two parts is set to be more than 800, defined as "ENER_SUM = ENER_P3 + ENER_P5". The additional constraints therefore are:

ENER_P3 / ENER_P5 > 1.5 and

ENER_P3 + ENER_P5 > 800.0

with ENER_P3 and ENER_P5 being functions of the part 3 mass fraction, the part 5 mass fraction, and the topology at that iteration. The intrusion and mass fraction constraints from the previous approach are also active.

The multi-point methodology must be activated, Figure 5-56. – all advanced constraint definitions require this option. All simulations of an iteration are run simultaneously, Figure 5-57.

In all cases the mass of the structure (actually the sum of the mass fractions) is minimized, which is the default objective for the multi-point methodology.

A maximum of 50 iterations are allowed.

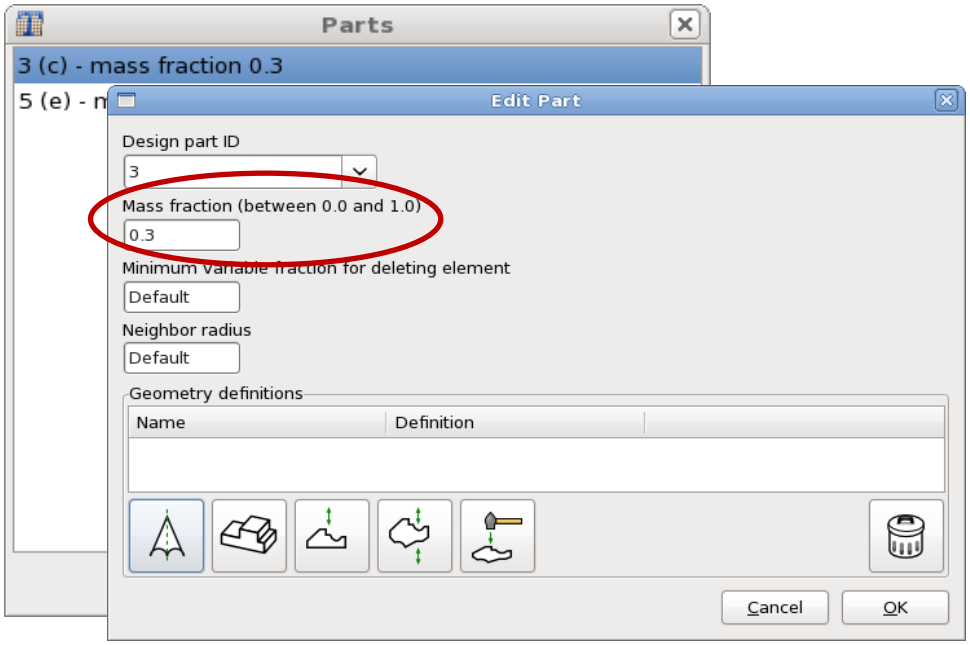

Figure 5-54: Design Parts definition with initial mass fraction 0.3

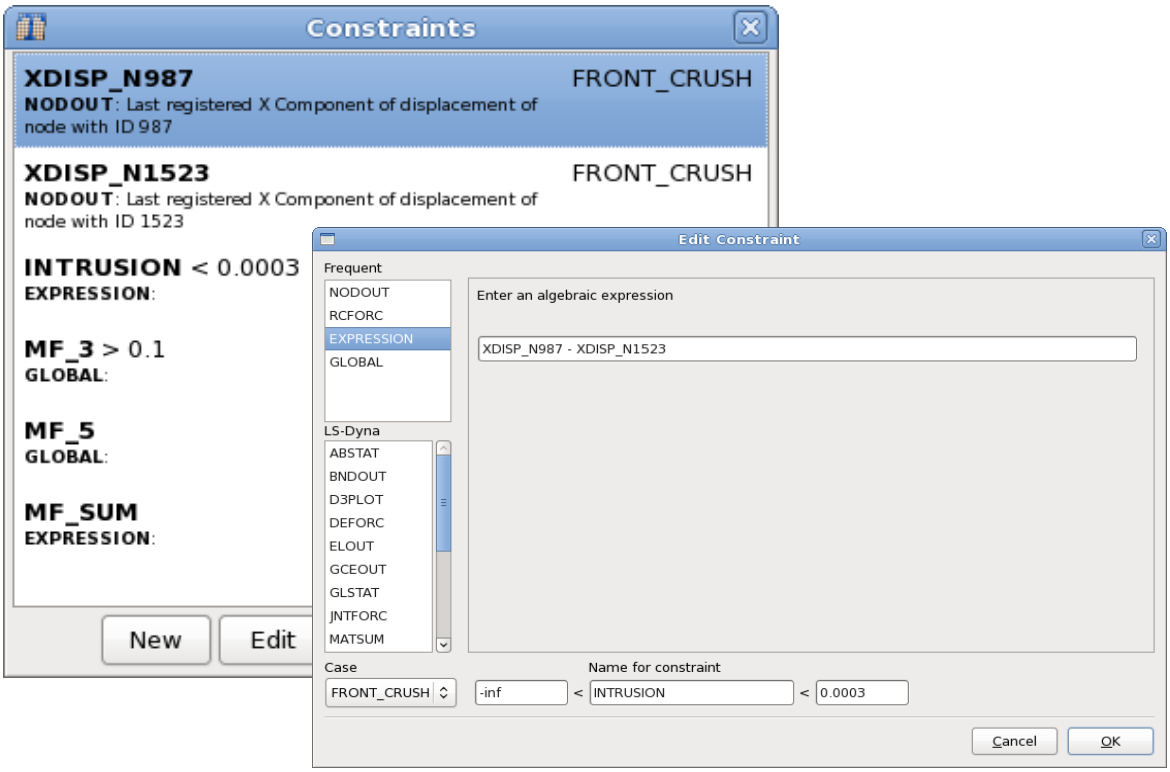

Figure 5-55: Constraints extracted from nodout database used in mathematical expression

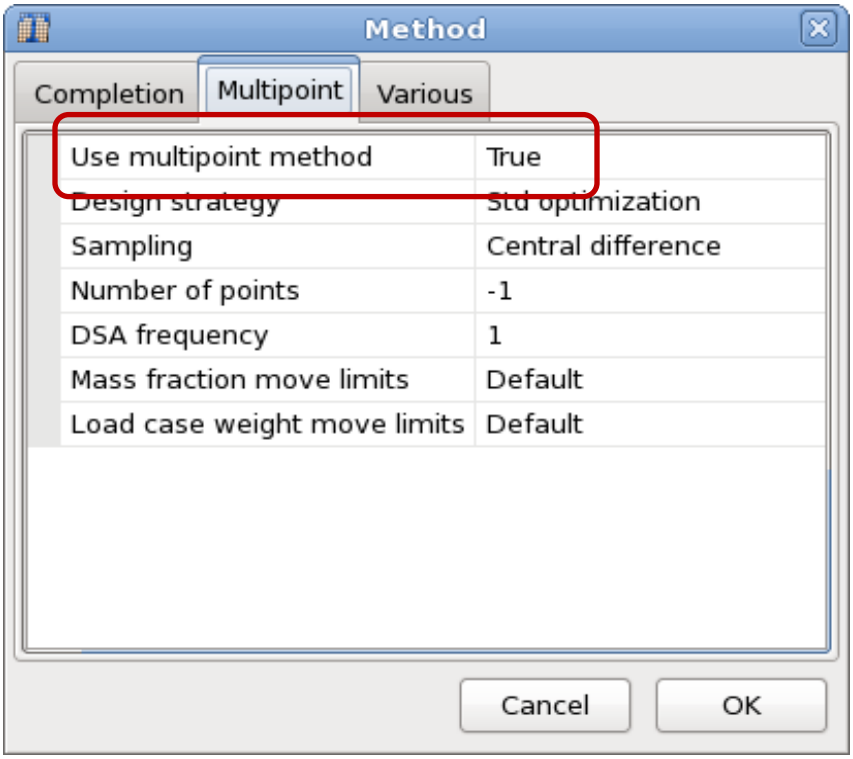

Figure 5-56: Activation of Multipoint methodology

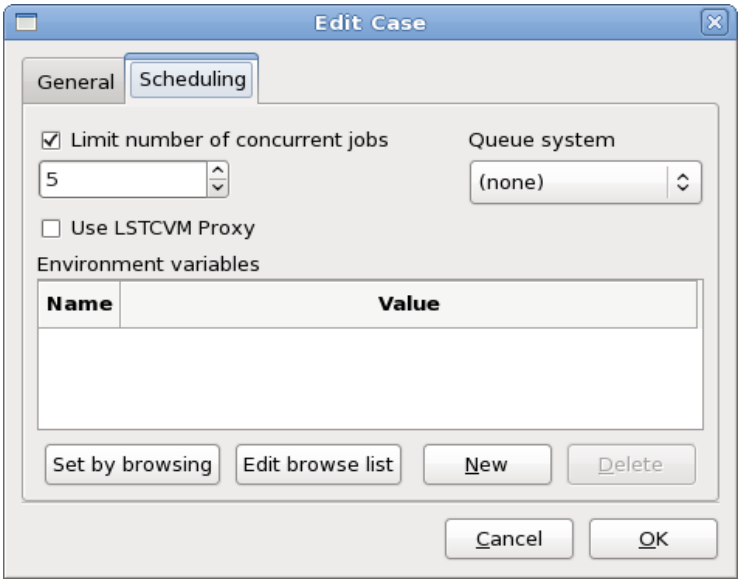

Figure 5-57: Concurrent jobs; Number of simulation jobs to be run in parallel

5.8.3. Results considering only the intrusion

The optimization converges after 32 iterations. The results are as shown in Figure 5-58 and Figure 5-59. In the figures it can be seen that part 3, the engine compartment, was made light to ensure the deformation occurs in the engine compartment. Also the side constraint on the

part 3 mass fraction is active – without this constraint, part 3 will be deleted to achieved a structure meeting the displacement constraint at the minimum mass.

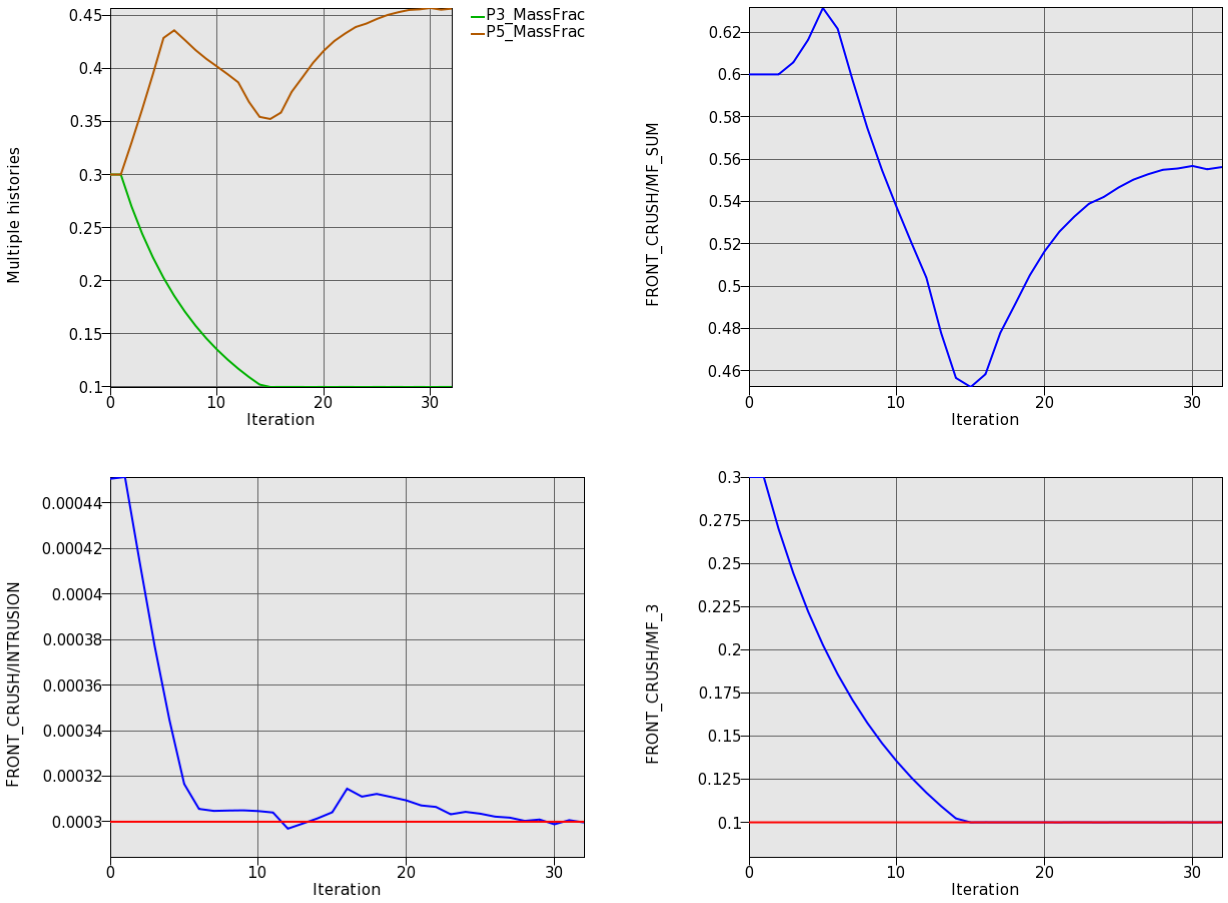

Figure 5-58: Convergence history for the multiple parts with intrusion constraint example

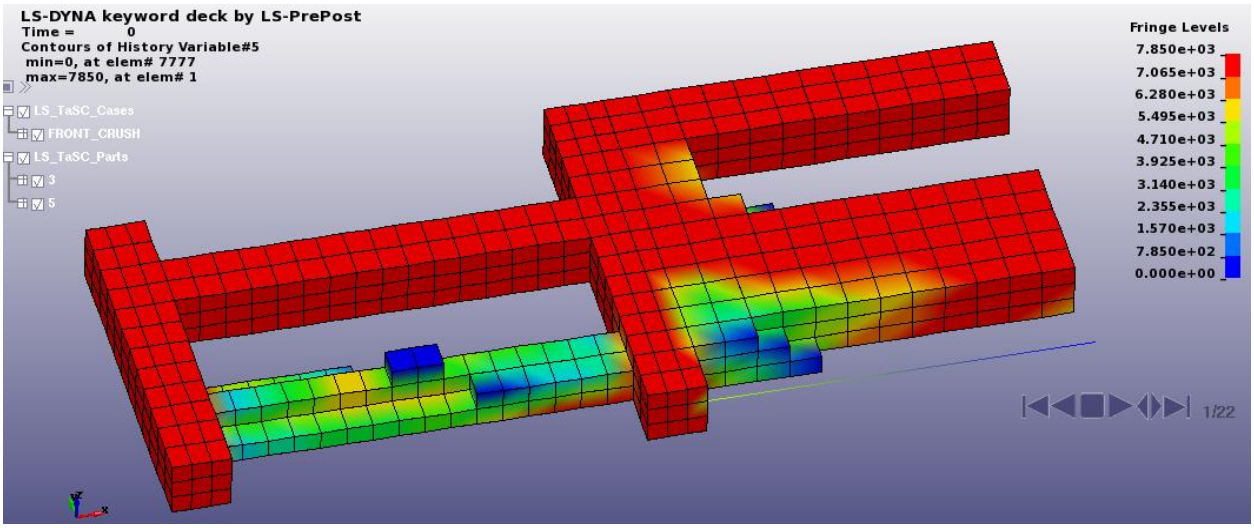

Figure 5-59: Final design for the multiple parts with intrusion constraint example; fringe component density

5.8.4. Results considering energy absorption of the parts

The optimization converges after 42 iterations. The results are as shown in Figure 5-60 and Figure 5-61. In the figures it can be seen that part 3, the engine compartment was made light to ensure that it absorbs most of the energy – thereby meeting the requirement that it absorbs 1.5 times more energy than the passenger compartment. Also, the two compartments together absorbed the required 800 units of energy. The intrusion constraint is not active.

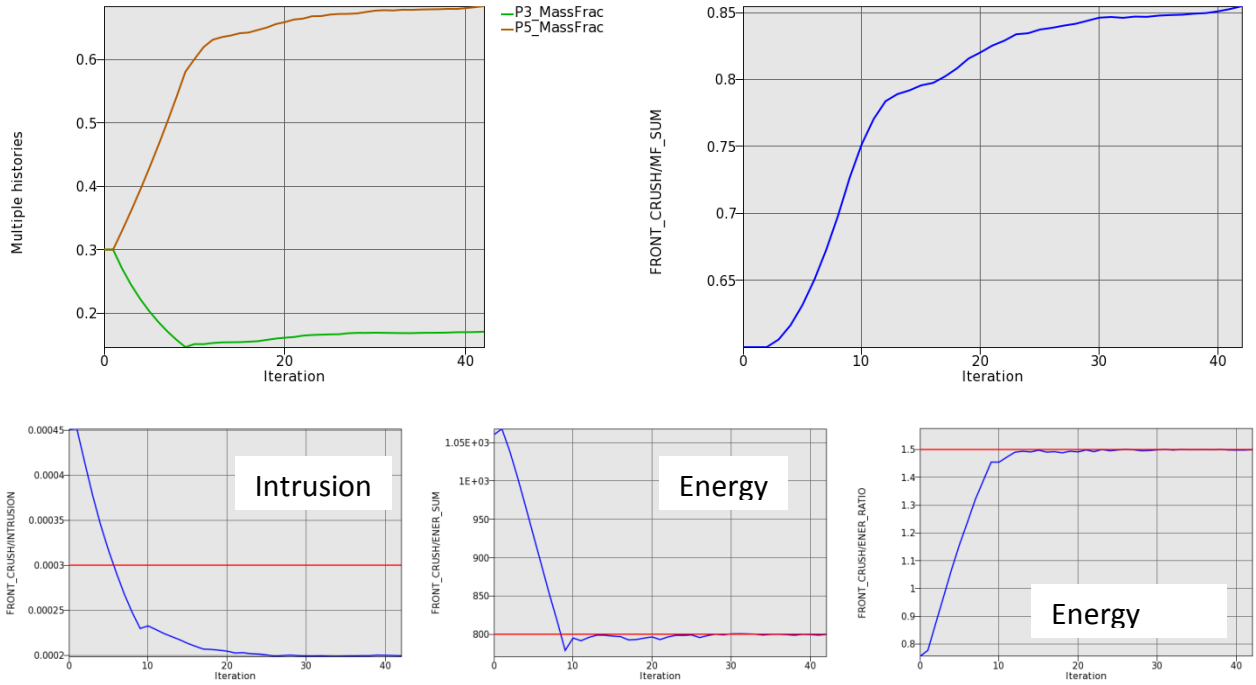

Figure 5-60: Convergence history for the multiple parts with energy constraints example

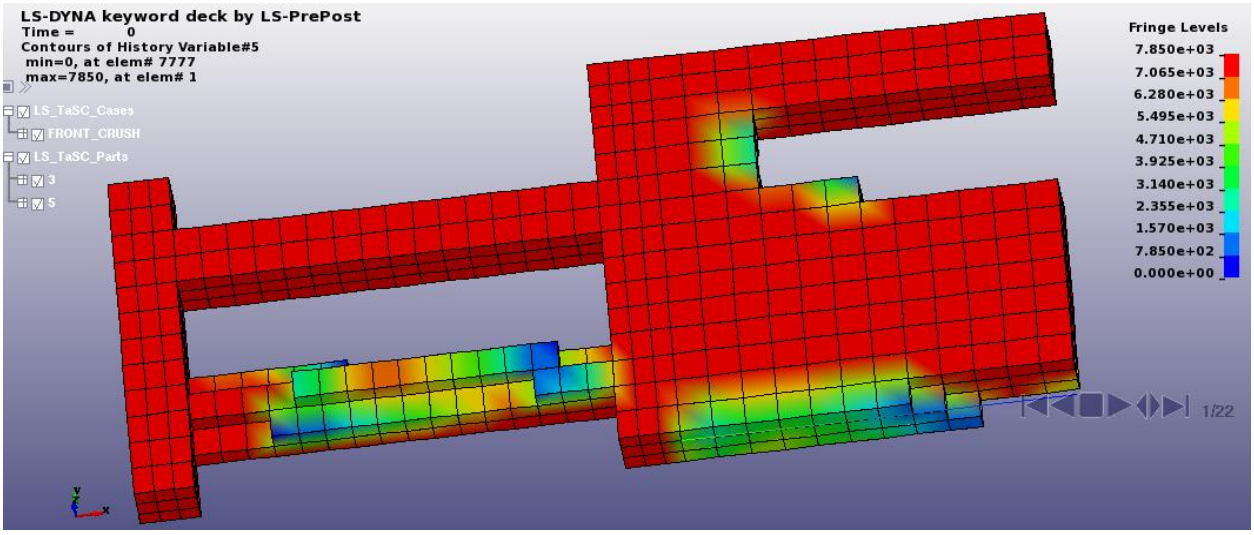

Figure 5-61: Final design for the multiple parts with energy constraints example, fringe component density

The design sensitivity information (the derivatives with respect to the part mass fractions) are displayed in Figure 5-62. The derivatives of the intrusion and energy ratio with respect to the mass fractions are reported. This design sensitivity information, used to compute the optimum design shown previously, helps to understand the structural behavior – e.g. how the structure should be changed to obtain a specific behavior. The values contain some noise, specifically relative the mass fraction of part 3, which has few elements, but the magnitude of the noise is

small relative to the value of the result. In some cases the results are constant over all iterations, but this is with respect to a different part in each case, so this observation cannot be generalized. All in all, the design sensitivity information seems useful and well behaved for this specific example.

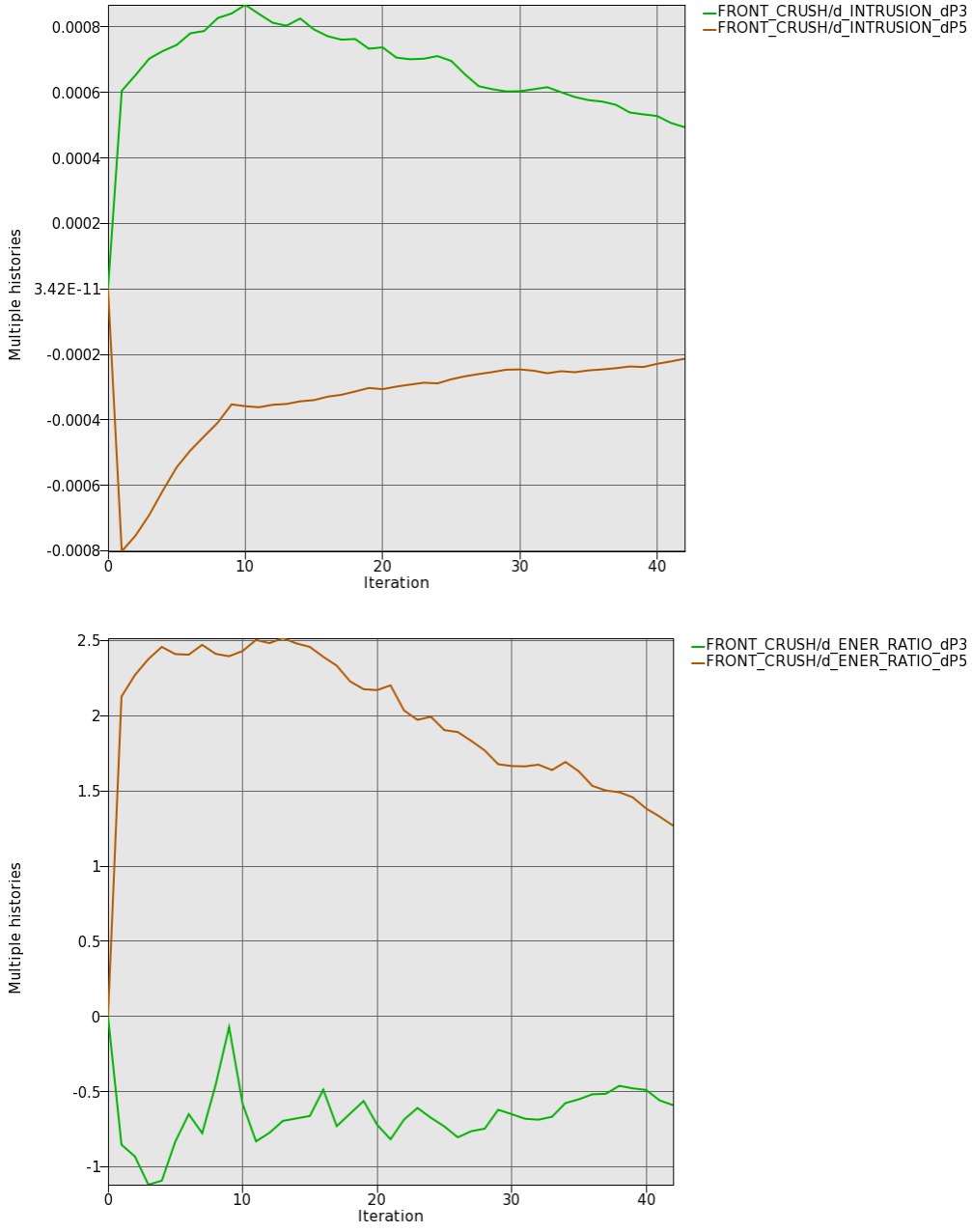

Figure 5-62: Design sensitivity information for the multiple part with energy constraints example.

5.9. Surface Design of a Beam

This example demonstrates:

- Free surface design for solids
- Extrusion and symmetry constraints for free surface design
- Smooth transition for free surface design

The related files are available in MANUAL/SURFACE/BEAM.

5.9.1. Problem Description

The geometry and loading conditions for the example are shown in Figure 5-63. The objective is to reduce stress concentrations using free surface design.

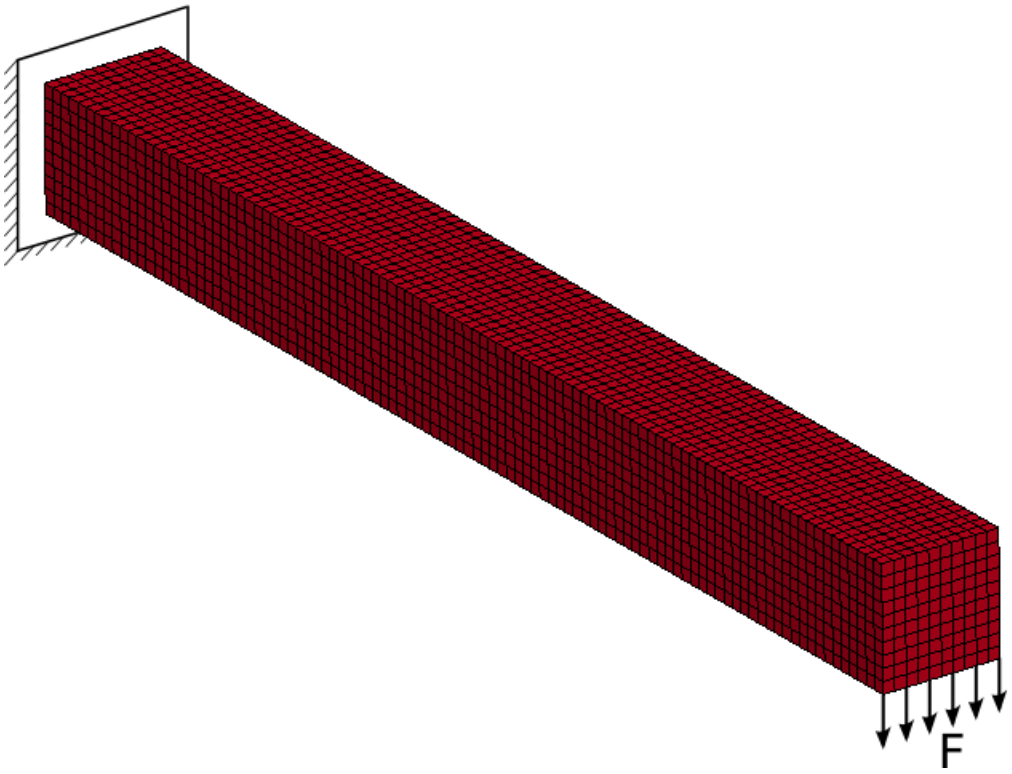

Figure 5-63: Beam model for free surface design

5.9.2. Problem Setup

To show various features of free surface design, four surfaces of the beam are optimized in the first example, in the second example, an extrusion and a symmetry constraint are defined, and in the third example, a smooth transition constraint is used.

The surface definition is displayed in Figure 5-64, Figure 5-65 and Figure 5-66, respectively. For the first two examples, the objective is to match the average stress, which is the default. The

smooth transition example uses the minimize volume objective, which matches the maximal stress. Note that for the example with symmetry and extrusion constraints, the neighbor radius was increased to 0.5 to avoid a sharp structure.

The convergence tolerance for this example is a 50% smoothing of the stress, Figure 5-67.

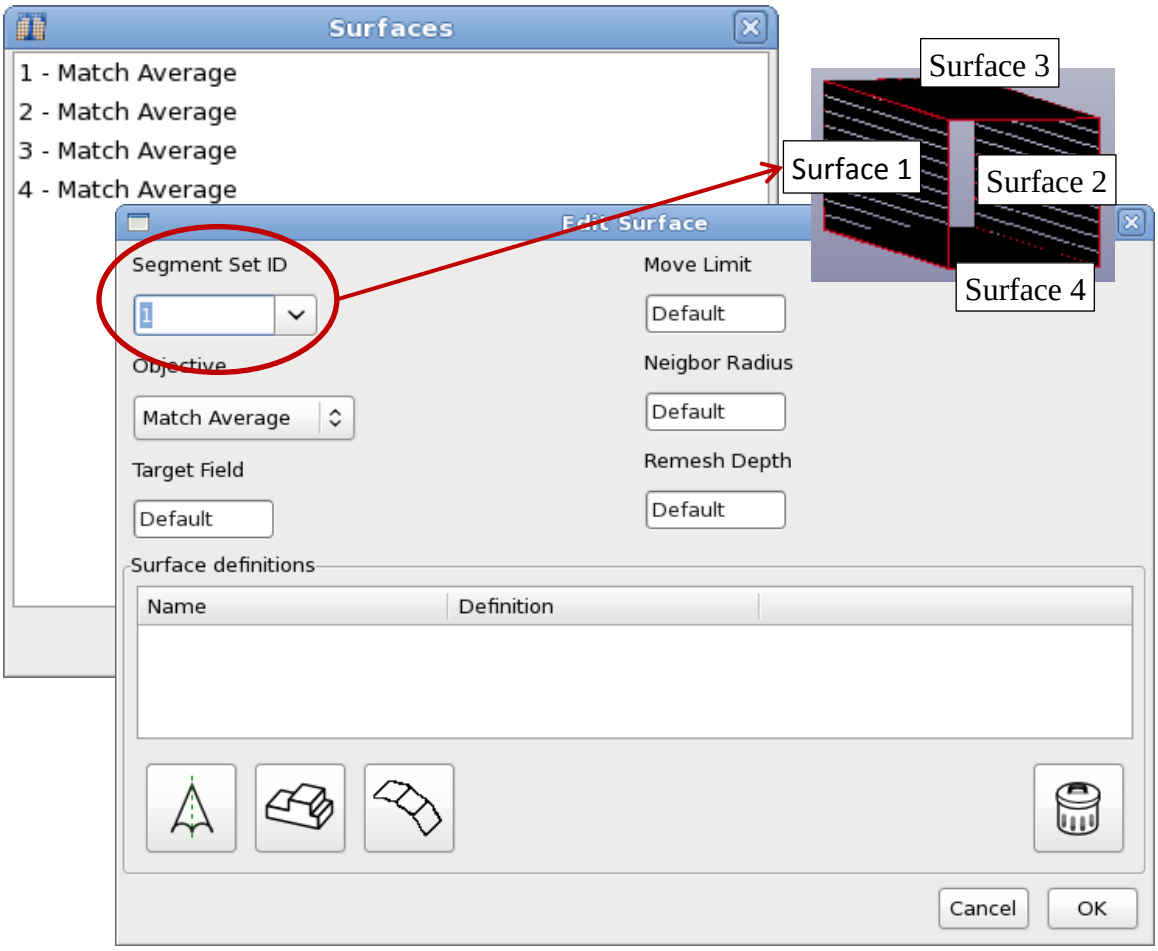

Figure 5-64: Definition of Surfaces; the objective is to match the average stress.

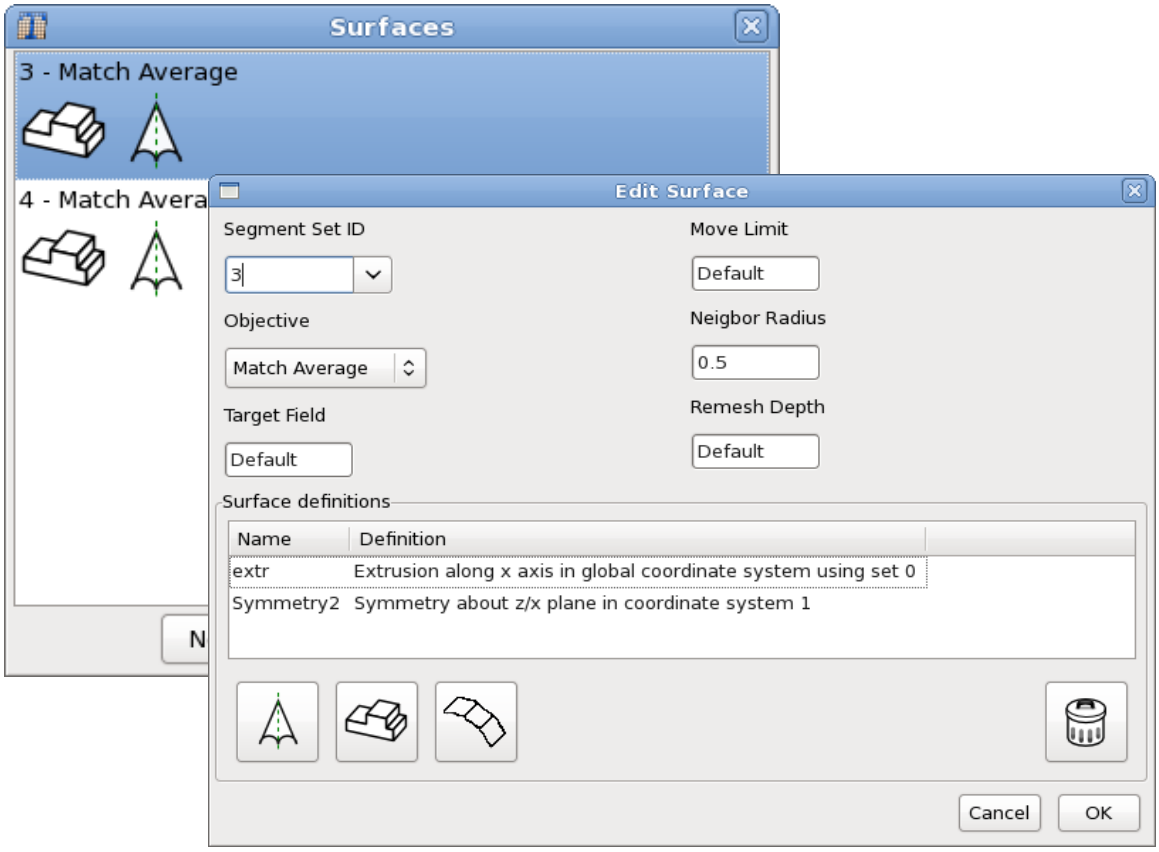

Figure 5-65: Definition of Surfaces with extrusion and symmetry constraint. To avoid a sharp geometry, the neighbor radius was increased to 0.5.

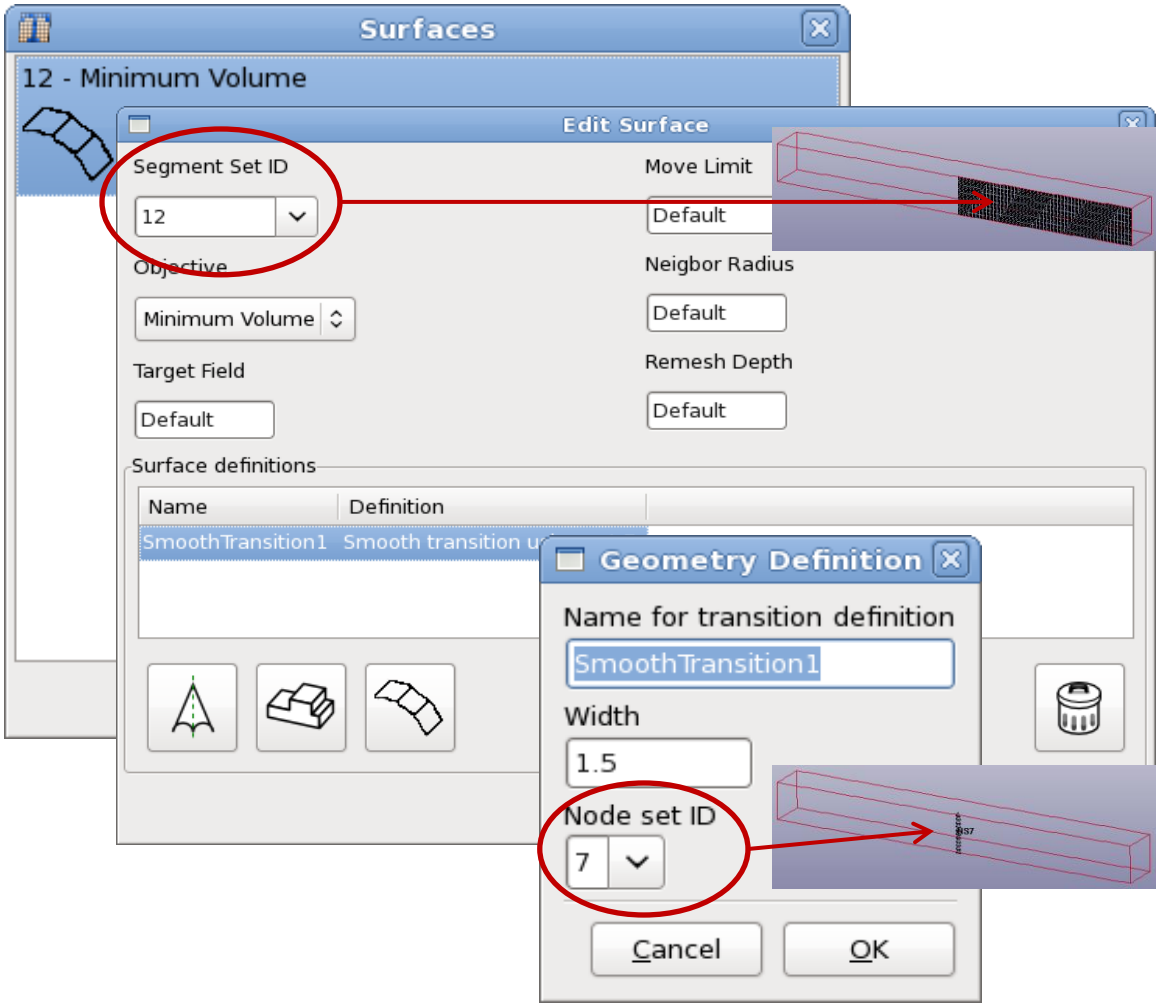

Figure 5-66: Surface with smooth transition definition. The objective is a minimum volume.

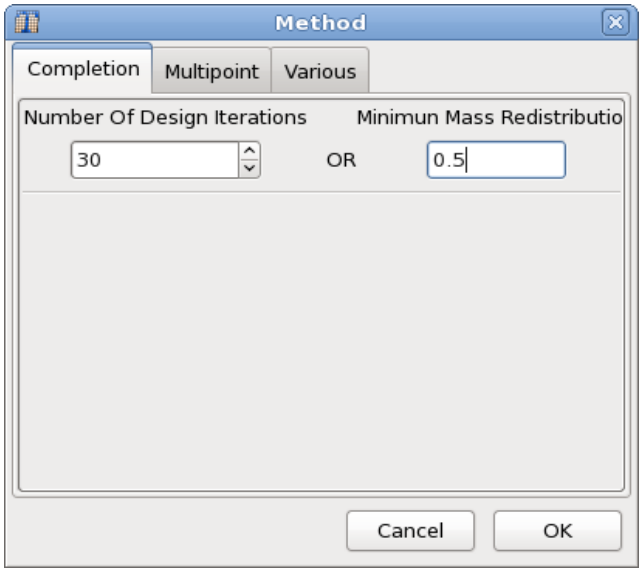

Figure 5-67: Termination criteria; the convergence tolerance is a 50% smoothing of the stress

5.9.3. Results with four surfaces

The project input data is saved to the file *all.lstasc* as provided in the examples distribution. All four sides of the beam were selected for shape design. The problem converged in 8 iterations. The initial and final design is displayed in Figure 5-68. Figure 5-69 shows the improvement of the stress smoothing.

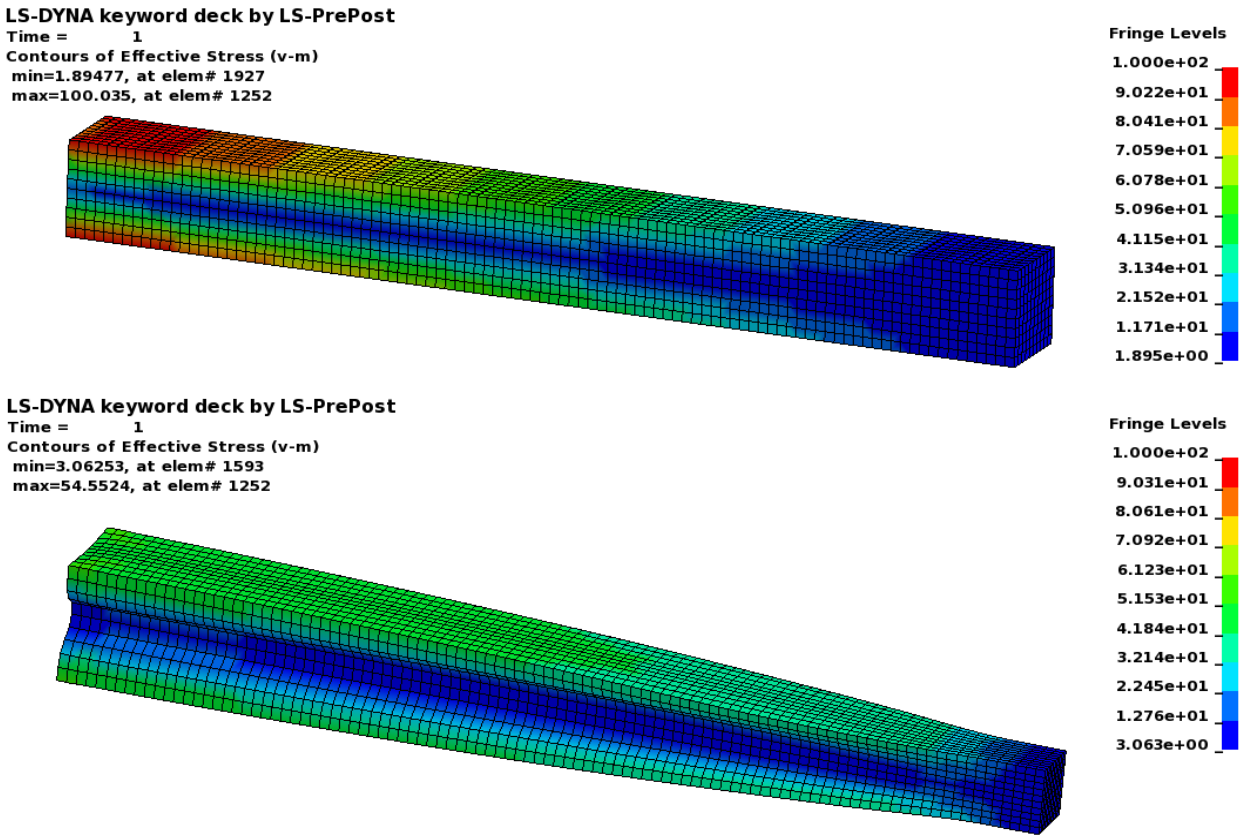

Figure 5-68: Initial and final design for four surfaces, Von Mises Stress fringed on the model

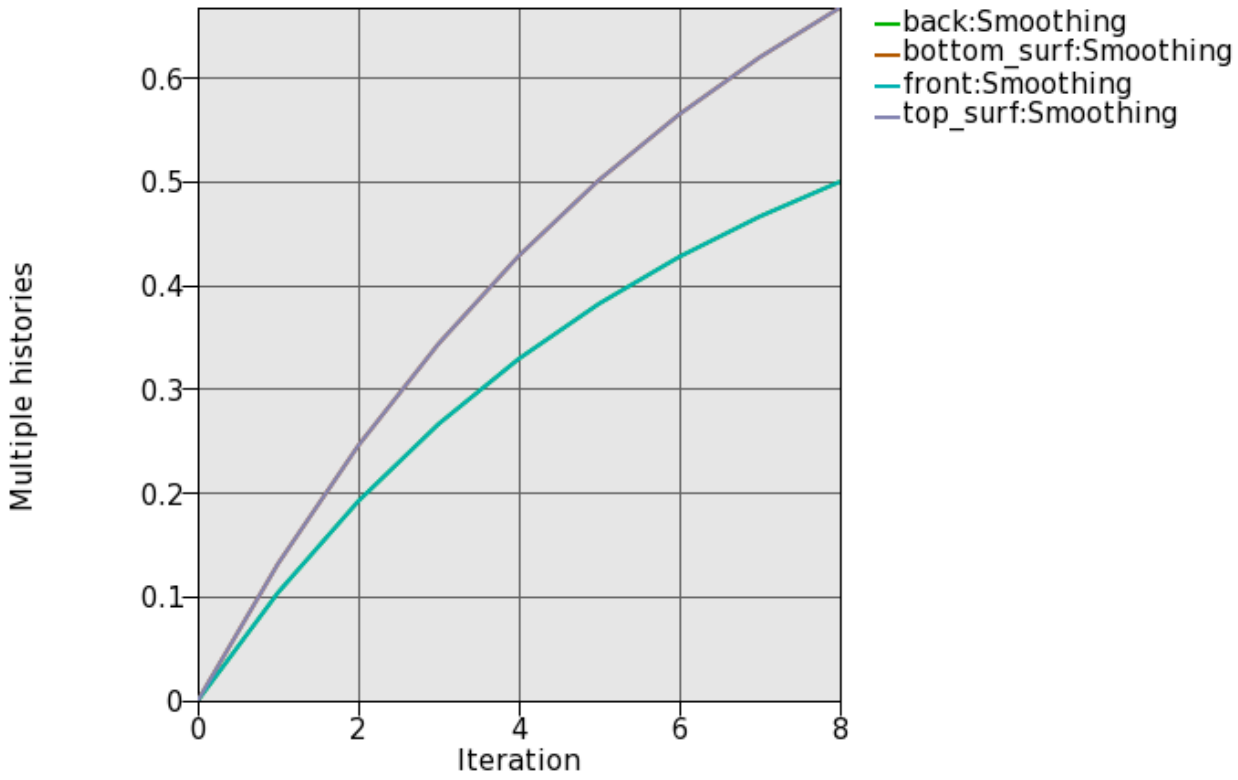

Figure 5-69: Convergence history; smoothing improvement of back/front and top/bottom

5.9.4. Results with extrusion and symmetry geometry definitions

The project input data is saved to the file *extr_symm.lstasc* as provided in the examples distribution. The front and back side of the beam were selected for shape design. The problem converged in 27 iterations. The initial and final design is shown in Figure 5-70. Note that for an extrusion such as this a complete smoothing of the stress is not possible, because the loading varies along the extrusion direction while the geometry does not. Figure 5-69 shows the improvement of the stress smoothing.

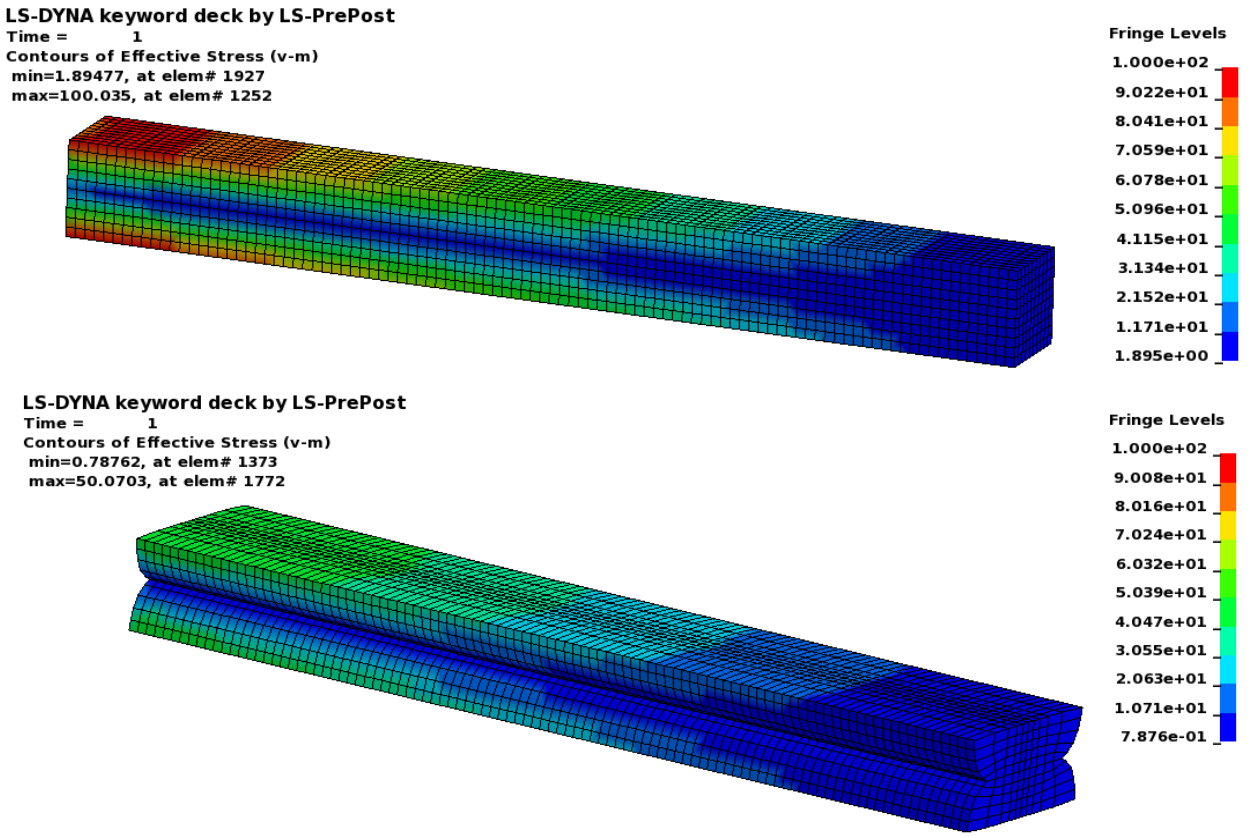

Figure 5-70: Initial and final design of beam with extrusion and symmetry geometry definitions with Von Mises stress fringed on model

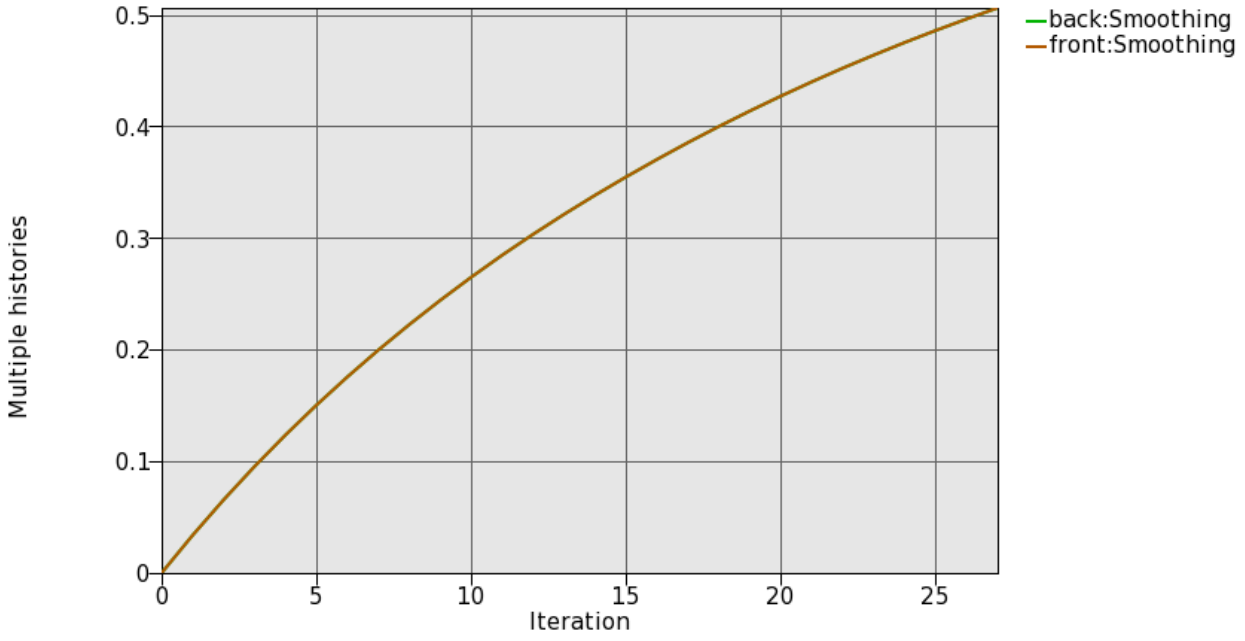

Figure 5-71: Convergence history of beam with extrusion and symmetry geometry definitions

5.9.5. Results with smooth transition geometry definition

The project input data is saved to the file *smooth_trans.lstasc* as provided in the examples distribution. The front half of the beam was selected for shape design. A node set was defined on the center edge and used to define the smooth transition, Figure 5-66. The objective was the minimum volume of the part. The initial and final design is as shown in Figure 5-72. The design without the smooth transition definition is shown in Figure 5-73 – the resulting poor mesh quality can be seen.

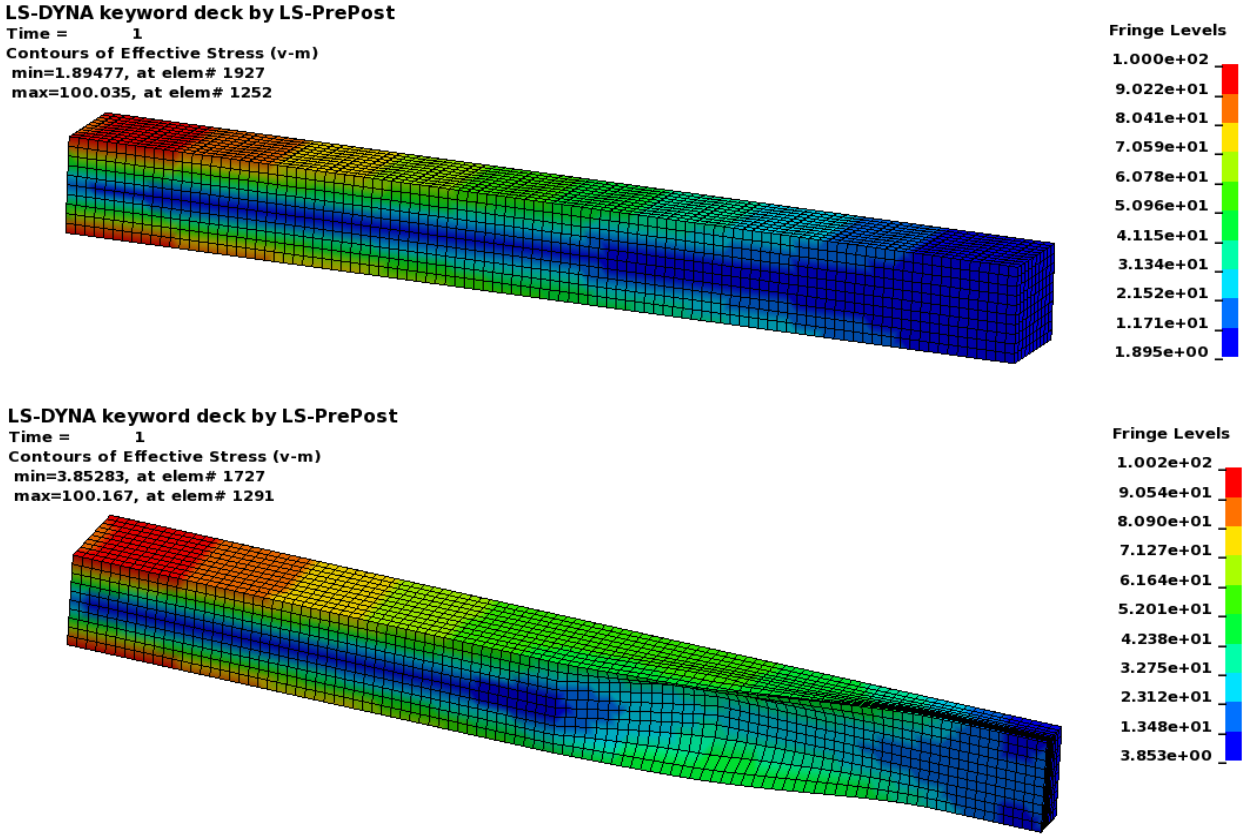

Figure 5-72: Initial and final design of beam with smooth transition geometry definition

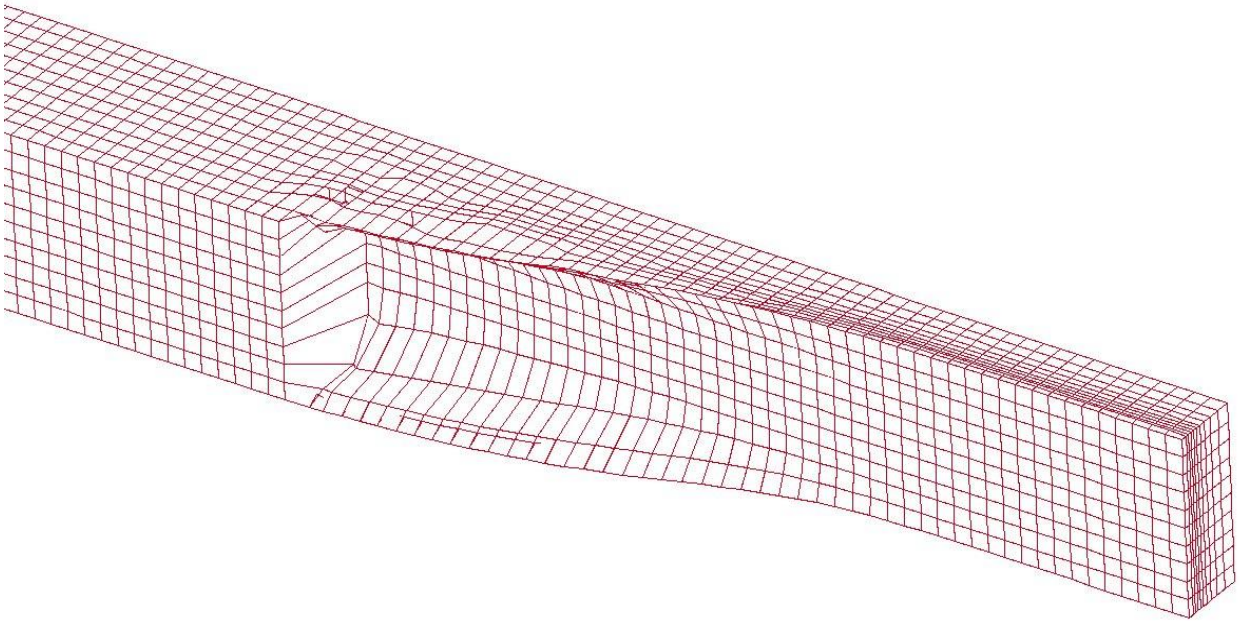

Figure 5-73: Design of beam without smooth transition geometry definition

6. Troubleshooting

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

6.1. 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.

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
```

6.2. 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.

6.3. 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 are not the same, 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.

6.4. 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.

- 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
- Increase minimum density fraction (default 0.05 for dynamic problems).

6.5. 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.

6.6. Convergence

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.

6.7. 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*. Do not use a symbolic link. You may need to investigate the latest version of LS-Prepost 2.4 and 3.1.

6.8. 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.

6.9. 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.

7. User Results

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

7.1. Background

For importing user results the user must compute a utility value for each element. For example, in LS-TaSC the energy density of each element is used as the element utilty.

In other codes the design is done considering some performance of the structure, for example a fundamental frequency, and the derivative of this performance with respect to the element variables are available. These derivatives must then be converted in element utility values. For example, if you maximize(F(x)) with F(x) the natural frequenc, then with say dF/dX1 = 50 and dF/dX2 = 20, then possible utility values are U1 = 0.5, U2 =0.2

7.2. 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:

6. LS-TaSC: LS-TaSC writes a file describing the variable values to the"*VariableValues.txt*" file.
7. USER: The user program reads the variable values from the "*VariableValues.txt*" file.
8. USER: The user program analyzes the structure
9. USER: The user program writes the results to a file
10. USER: The user results are converted to LS-TaSC utility values using a post processing script, and writes these values to the "*UserResults.txt*" file.
11. 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.

12. 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).
13. LS-TaSC: LS-TaSC read the "*UserResults.txt*" results from the file.

7.3. 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 performance (commonly called an objective or constraint) 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.

7.4. Details of the performance and element utility

A good choice of the performance (commonly called an objective or constraint) 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 however the utility of each element in the design domain. The performance is converted to this element utility values. So the element utility values can be:

- The IED for each element. This is the utility value normally used by LS-TaSC. This sums up to the internal work, related to the external work. Note that the value reported in the d3plot file for solids must be scaled with the design variable (the volume of material in the element) to obtain the actual IED for the material in that element. LS-TaSC scale this value value internally for solids, both for the d3plot file and for user results. The shell IED values in the d3plot file is correct seeing that the shell thickness is the design variable.
- A user-defined utility value for each element. For example, $U_{i,j}$, the utility of element i for performance P of load case j, can be computed using the derivatives as $U_{i,j} = A_j \frac{\partial P_j}{\partial x_i} + B_j$ with A and B constants selected considering experience regarding P.

7.5. 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 $ will be ignored.

7.6. 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 utility. Here is a sample "*UserResults.txt*" file:

1 4.88738e-06

2 5.87231e-06

…

31 0.00468277

32 0.0114259

The utility of the element, have the following properties:

14. U must be larger than 0.
15. The U value is used to rank the elements in terms of usefulness. A U =1.0 indicates that the element is more useful than an element with U=0.0. Similarly a U=5.5 is more useful than U=3.7.
16. If the U value is scaled, then the same scaling should be used in all iterations. Best to scale using the load case weight.
17. The results for solid elements should be relative to the volume of the element, and not the amount of material (design variable) in that element. For example, if a FE program reports an IED value of 30 for an element with x=0.1, then on a material basis the actual IED value should be reported as 30/0.1 = 300, but LS-TaSC expects a value of 30 (based on the element volume). LS-TaSC does this conversion internally for solid elements, bot d3plot results and user-defined results.

7.7. 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 the 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.

7.8. Solid/Void

It is best if solid/void schemes such as SIMP are not activated in the user program. These will be applied by LS-TaSC.

7.9. 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.

8. 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.

8.1. Theory manual

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

8.2. Scripting manual

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

8.3. Queueing system installation

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