The LS-TaSC™ Software

TOPOLOGY AND SHAPE COMPUTATIONS USING THE LS-DYNA® SOFTWARE

USER'S MANUAL

July 2013
Version 3.0

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

Support Addresses
Livermore Software Technology Corporation
7374 Las Positas Road
Livermore, California 94551
Tel: 925-449-2500 ♦ Fax: 925-449-2507
Email: sales@lstc.com
Website: www.lstc.com

Livermore Software Technology Corporation
1740 West Big Beaver Road
Suite 100
Troy, Michigan 48084
Tel: 248-649-4728 ♦ Fax: 248-649-6328

Disclaimer

LSTC reserves the right to modify the material contained within this manual without prior notice.

The information and examples included herein are for illustrative purposes only and are not intended to be exhaustive or all-inclusive. LSTC assumes no liability or responsibility whatsoever for any direct of indirect damages or inaccuracies of any type or nature that could be deemed to have resulted from the use of this manual.

18-Dec-13

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 China office. 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

Preface to Version 3 3
Preface to Version 2.1 4
Preface to Version 2 5
Preface to Version 1 6
Table of Contents 7
1. Introduction 10
1.1. Classification of Structural Optimization Techniques 10
1.1.1. Topology Optimization 10
1.1.2. Topometry Optimization 10
1.1.3. Size Optimization 10
1.1.4. Shape Optimization 10
1.2. Brief Overview 11
1.3. Topology Optimization Method in LS-TaSC 11
1.4. Finding Information 12
1.5. References 12
2. Topology Optimization 14
2.1. The Design Parts 14
2.1.1. Design of Solids 14
2.1.2. Design of Shells 14
2.1.3. Element types 15
2.1.4. Material data 15
2.2. Geometry and Manufacturing Definitions 15
2.3. Convergence 17
2.4. Design Variables 17
2.4.1. Mapping Elements to the Design Variables 17
2.4.2. Filtering of Results 17
2.4.3. Initialization, Deletion, and Regeneration of the Design Variables 18
2.5. LS-DYNA® Modeling Specifics 18
2.5.1. The Contact Definition 18
2.5.2. Part Definition 19
2.5.3. Part Set Definition 19
2.5.4. Element Set Definition 19
2.5.5. Unsupported Keywords 19
2.5.6. Disallowed Keywords 19
2.5.7. Automatic Keyword Edits by LS-TaSC 20
2.5.8. LS-DYNA® Simulation 20
2.6. Global Constraints 20
2.7. Dynamic Load Cases Weighing 21
3. Free Surface Design 22
3.1. The Design Surfaces 22
3.2. Geometry and manufacturing definitions 22
3.3. Convergence 23

3.4. Design Variables 23
3.5. Filtering of Results 23
3.6. LS-DYNA® Modeling Specifics 24
3.6.1. Surface Definition 24
3.6.2. Smooth Transition 24
3.6.3. Disallowed Keywords 24
3.7. Automatic mesh smoothing 24
4. Program Execution 25
4.1. Running the Program 25
4.2. Problem Definition 25
4.2.1. LS-DYNA® Simulation 25
4.3. Setting up the Problem 26
4.3.1. The Toplevel GUI 26
4.3.2. The Cases Panel 27
4.3.3. The Constraints Panel 28
4.3.4. The Parts Panel 30
4.3.5. The Surface Panel 33
4.3.6. Part and Surface Geometry 34
4.3.7. The Completion Panel 35
4.3.8. The Run Panel 36
4.3.9. Setting advanced options 36
4.4. Viewing Results 38
4.5. Databases and Files 40
4.6. Opening and Saving Projects 41
4.7. Restart 41
4.8. Script Commands 42
5. Example Problems 43
5.1. Fixed Beam with Central Load 43
5.1.1. Problem Description 43
5.1.2. Input 43
5.1.3. Output 44
a) Convergence History 44
b) Density Contours 45
5.2. Beam using geometry definitions 45
5.2.1. Input 46
5.2.2. Output 46
a) Extrusion and Casting 46
b) Extrusion and two-sided casting 47
5.3. Force-Displacement Constraints 47
5.3.1. Problem Definition 47
5.3.2. Input 47
5.3.3. Output 48
a) Convergence History 48
b) Density Contours 49
5.4. Linear Static Loading 49
5.4.1. Problem Definition 49

5.4.2. Input 50
5.4.3. Output 50
a) Convergence History 50
b) Density Contours 51
5.5. Shell Example 52
5.5.1. Problem Definition 52
5.5.2. Input 52
5.5.3. Output 52
a) Convergence History 52
b) Final Shell Thicknesses 53
5.6. Multiple Load Cases 53
5.6.1. Problem Definition 54
5.6.2. Input 54
5.6.3. Results with constant weights 54
5.6.4. Results with dynamic weighing 56
5.7. Surface Design of a Beam 58
5.7.1. Problem Definition 58
5.7.2. Results with four surfaces 58
5.7.3. Results with extrusion and symmetry geometry definitions 60
5.7.4. Results with smooth transition geometry definition 61
6. Troubleshooting 64
6.1. Executable failing or no output 64
6.2. Design Part 64
6.3. Extrusion Set 64
6.4. Negative Volumes 64
6.5. The LS-DYNA analysis fails if a smaller mass fraction is requested 64
6.6. Convergence 65
6.7. LS-PREPOST 65
6.8. Casting definitions 65
6.9. Mysterious Error when/after calling LS-DYNA and/or Errors involving the LSOPT Environment Variable 65
7. Other LS-TaSC MANUALS 67
7.1. Theory manual 67
7.2. Scripting manual 67
7.3. Queueing system installation 67

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:

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. Brief Overview

Topology optimization in structures has been studied since the 1970s resulting in many books and numerous papers. The books by Rozvany [1] and Bendsøe and Sigmund [2] provide a very comprehensive and contemporary survey of optimization techniques used in topology optimization. Most previous studies [3, 4] in topology optimization have focused on designing structures with static loading conditions but there is relatively little work on handling problems involving dynamic loads, like those observed in crashworthiness optimization [5]. The topology optimization in the context of crashworthiness is a very complex problem due to non-linear interactions among material non-linearities, geometry, and transient nature of boundary conditions.

The most efficient topology optimization methods use sensitivity information (optimality criterion based methods, Rozvany [1], Bendsøe and Kikuchi [6]) to drive the search for an optimum. Sensitivity calculations are computationally inexpensive for linear-static problems but not for the problems that involve non-linearities. To use the same set of topology optimization methods, one needs to explicitly calculate sensitivities which is practically infeasible due to very high computational cost involved with simulations. Thus the theory used to solve the linear-static load cases, though quite mature, is not practical for the crashworthiness problems and alternate methods need to be explored.
Previously different approaches have been adopted by authors to solve topology optimization with nonlinearities. Pedersen used the Method of Moving Asymptotes for crashworthiness optimization of two-dimension structures [7]. They used a quasi-static nonlinear FEA to account for geometric nonlinearities to handle large deformation and rotation of plastic beam elements. However, the method ignored the contact between elements arising due to nonlinear behavior of the structures. Soto [8, 9] presented a heuristics based method using a prescribed plastic strain or stress criterion to vary the density to achieve the desired stress or strains with a constraint on mass. However, this method could not be generalized to solid structures. Pedersen [10] used beam elements to handle topology in crashworthiness optimization. Forsberg and Nilsson [11] proposed two algorithms to get a uniform distribution of the internal energy density in the structure. In the first method, they deleted inefficient elements and in the second method they updated the thicknesses of the shell elements. This method also was limited to a small set of optimization problems. Shin et al. [12] proposed an equivalent static load method where they calculated an equivalent static load for the dynamic problem and then used the linear-static topology optimization techniques to find the optimal topology. The main difficulty in this method is the requirement to accurately compute the equivalent loads.

1.3. Topology Optimization Method in LS-TaSC

A heuristic topology optimization method developed at the University of Notre Dame, known as hybrid cellular automata [13], 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 3.0 methodology (the product name together with the version number).

1.4. 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.

1.5. References

1. GIN Rozvany, *Structural Design via Optimality Criteria*, Kluwer, London, 1989.
2. MP Bendsøe, O Sigmund, *Topology Optimization: Theory, Methods and Applications*, Springer-Verlag, Heidelberg, 2003.
3. HA Eschenaur, N Olhoff, Topology Optimization of Continuum Structures: A Review, *Applied Mechanics Review*, 54(4), 331-390, 2001.
4. GIN Rozvany, *Topology Optimization in Structural Mechanics*, Springer-Verlag, Vienna, 1997.
5. CA Soto, Applications of Structural Topology Optimization in the Automotive Industry: Past, Present, and Future, in HA Mang, FG Rammerstorfer, J Eberhardsteiner (eds), *Proceedings of the Fifth World Congress on Computational Mechanics*, Vienna, 2002.
6. MP Bendsoe, N Kikuchi, Generating Optimal Topologies in Optimal Design using a Homogenization Method, *Computer Methods in Applied Mechanics and Engineering*, 71(2), 197-224, 1988.
7. CBW Pedersen, Topology Optimization Design of Crushed 2d-Frames for Desired Energy Absorption, *Structural and Multidisciplinary Optimization*, 25, 368-282, 2003.
8. CA Soto, Structural topology optimization: from minimizing compliance to maximizing energy absorption, *International Journal of Vehicle Design*, 25(1/2), 142-163, 2001.
9. CA Soto, Structural Topology Optimization for Crashworthiness, *International Journal of Numerical Methods in Engineering*, 9(3), 277-283, 2004.

10. CBW Pedersen, Crashworthiness Design of Transient Frame Structures Using Topology Optimization, *Computer Methods in Applied Mechanics in Engineering*, 193, 653-678, 2004.
11. J Forsberg, L Nilsson, Topology Optimization in Crashworthiness Design, *Structural and Multidisciplinary Optimization*, 33, 1-12, 2007.
12. MK Shin, KJ Park, GJ Park, Optimization of Structures with Nonlinear Behavior Using Equivalent Loads", *Computer Methods in Applied Mechanics and Engineering*, 196, 1154-1167, 2007.
13. A Tovar, *Bone Remodeling as a Hybrid Cellular Automaton Optimization Process*, PhD Thesis, University of Notre Dame, 2004.

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 stress.

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. Elements shapes close to perfectly cubic are the best for the current neighbor selection 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. Elements shapes close to perfectly square or an equilateral triangle are the best for the current neighbor selection algorithm.

Tetrahedral and triangular elements cannot be extruded.

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.

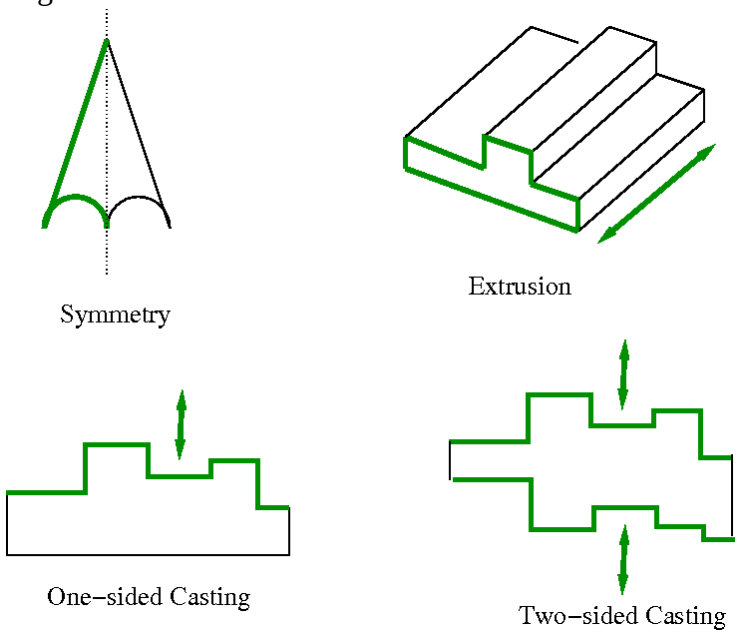

Figure 2-1: Geometry definitions

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.

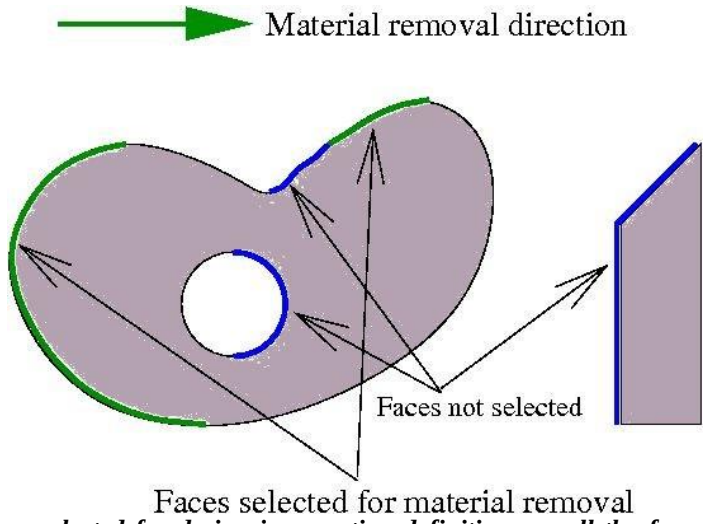

Figure 2-2: The faces selected for design in a casting definition are all the faces facing the material removal direction. The algorithm will not consider the faces shown in blue.

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 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

Structured grids are not always possible for industrial applications, and the results should be mesh independent. A radius based strategy is therefore 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.

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. Unsupported Keywords

The *INCLUDE keyword is not supported in the current version. It may be used, but LS-TaSC will not follow the link.

2.5.6. Disallowed Keywords

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

2.5.7. 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.

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.

2.5.8. 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 4.3.2.

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. Global Constraints

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 actually 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.

2.7. Dynamic Load Cases Weighing

It may happen that a single load case dominates the topology of the final design 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 good weighing 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.
The dynamic weighing is done by defining a desired relationship between the responses of all the load cases. The algorithm will scale the load case weights to achieve this relationship. Say we have constraint C_1 from the first load case and constraint C_2 from the second load case, then we write our desired behavior as $k_1 C_1 + offset_1 = k_2 C_2 + offset$ with C the constraint value, k a scale factor, and an offset added.
The final weights found are not suitable for restarting. They can be examined though for an indication of good values of the weights, but usually the final weights found using dynamic weighing are too large.

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.

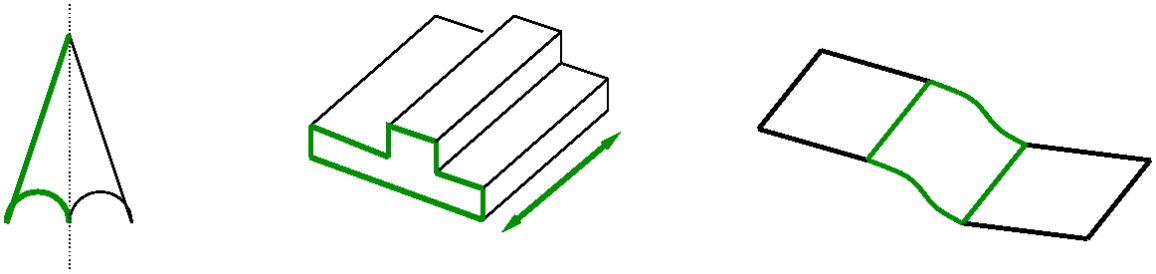

Figure 3-1: Geometry definitions

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.

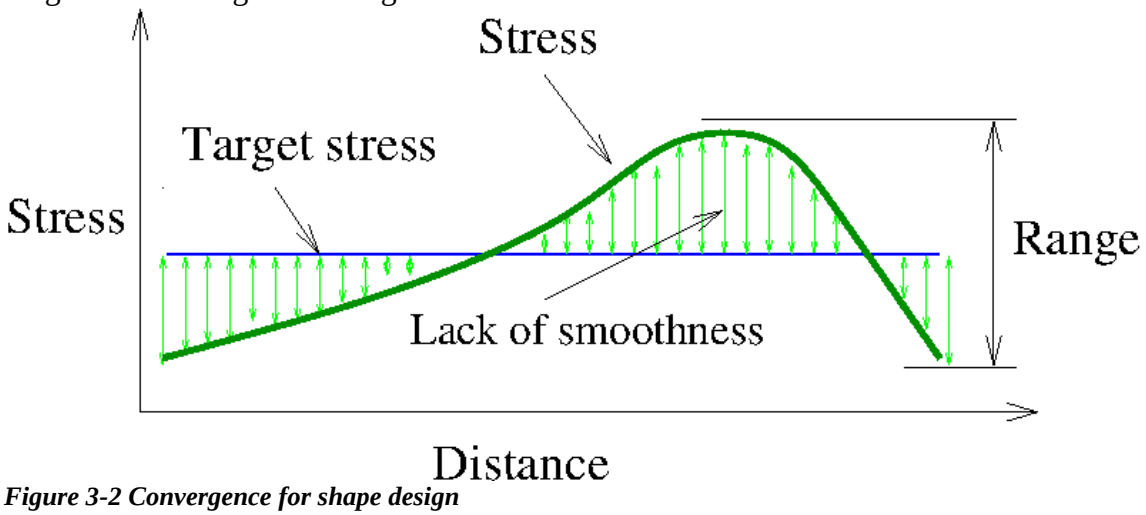

Figure 3-2 Convergence for shape design

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.

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.

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 *INCLUDE and *PARAMETER keywords are 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.

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 smooth 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 the automated design, and the evaluation of the results.

4.1. Running the Program

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

1. With the execution command
   ```
   $ lstasc myProject.lstasc
   ```
2. The *file open* dialogue, available from the File pulldown menu

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

4.2. 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.

4.2.1. 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 4.3.2.

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

4.3. 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.

4.3.1. 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.

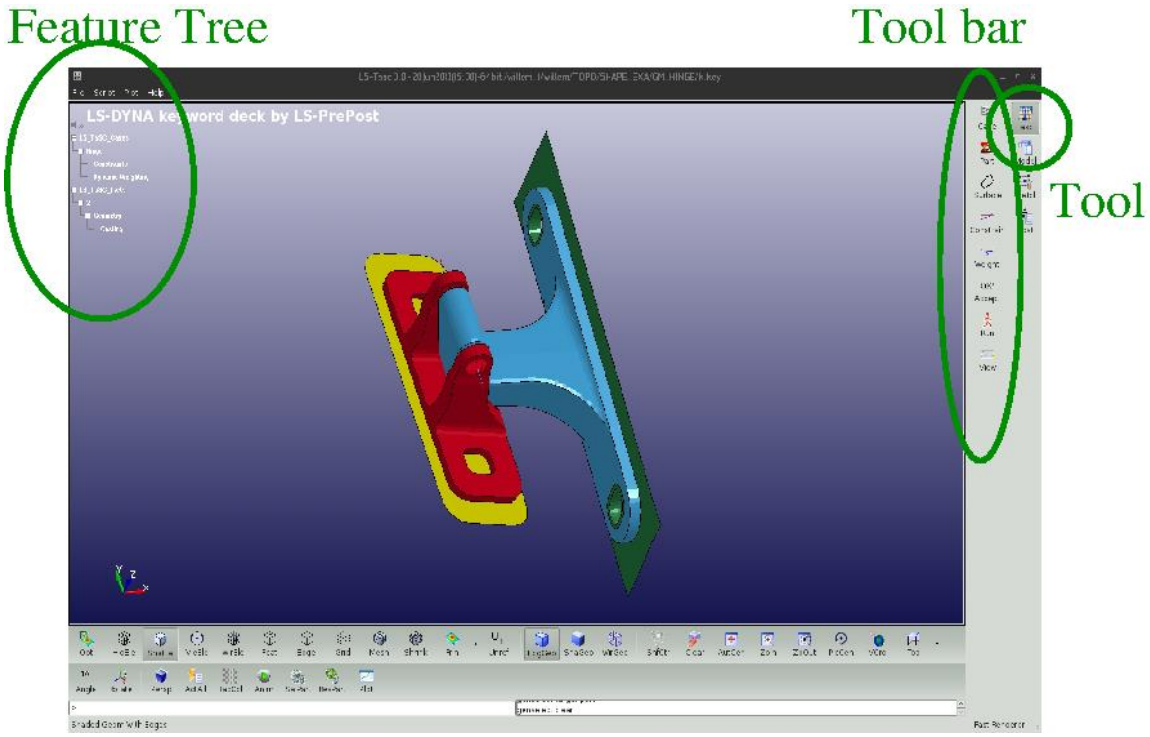

Figure 4-1: The toplevel GUI

4.3.2. 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.

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 jobs	This parameter indicates the number of processes to be run simultaneously. A value of zero indicates all processes would 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.

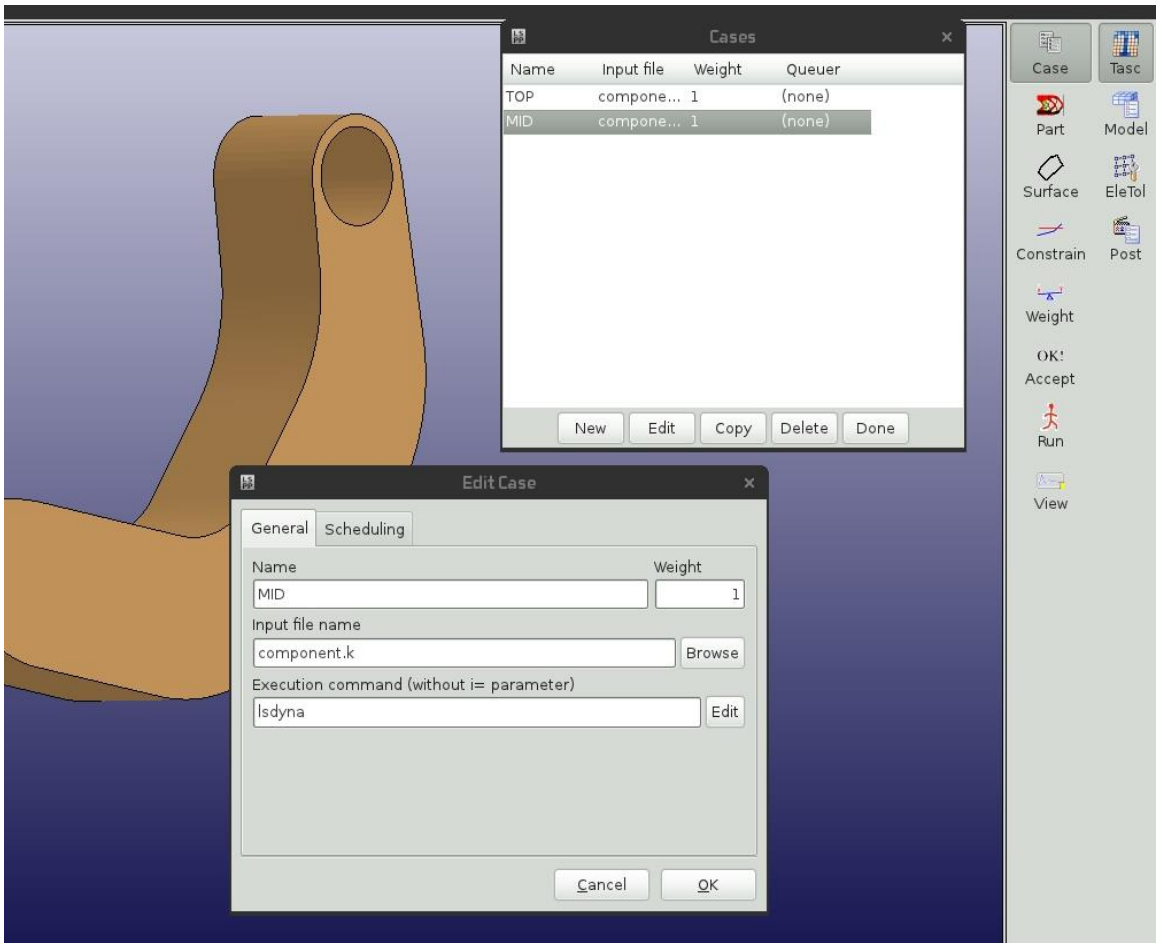

Figure 4-2: The cases panel.

4.3.3. The Constraints Panel

The constraint panel contains the global constraints on the structure. See the following table and Figure 4-2 for more details.

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`	One of NODOUT (stiffness), RCFORC (compliance), or USERDEFINED (see text).
`Lower and upper bound`	The weight associated with a case is defined here. This enables the user to specify non-uniform importance while running multiple cases.
`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.

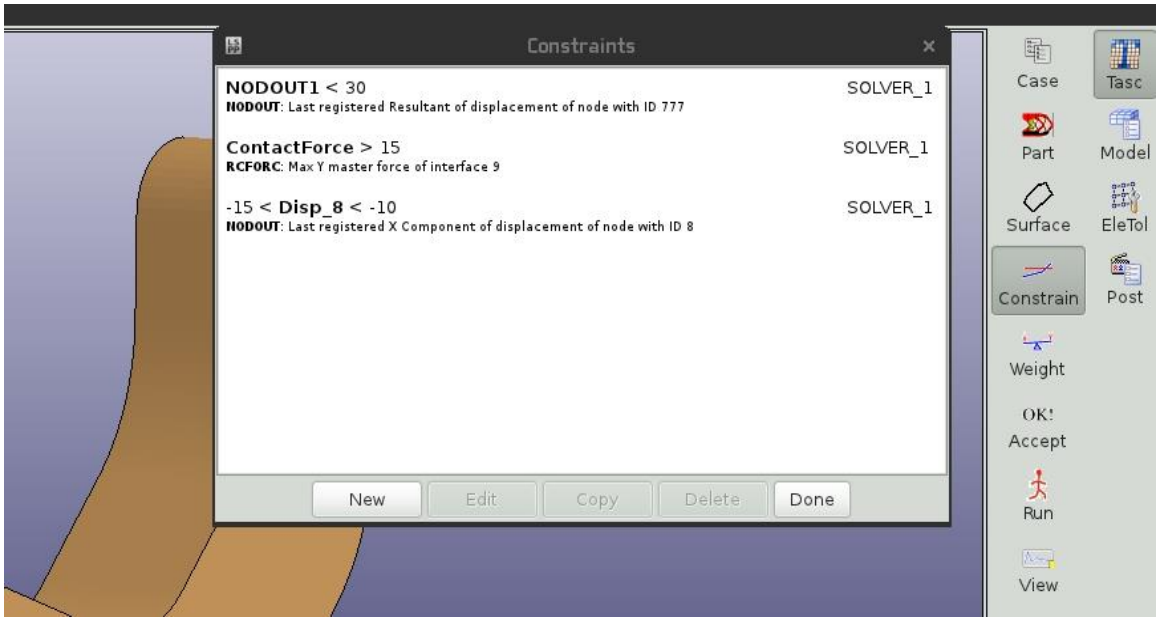

Figure 4-3: The constraints overview panel.

The USERDEFINED responses require a string to be specified. The only allowable commands are the *D3PlotResponse* and *BinoutResponse* commands as defined in the LS-OPT manual; for example, "*D3PlotResponse -pids 101 -res_type stress -cmp von_mises -select MAX -start_time 0.0000*". Easiest is to use LS-OPT to create these strings. You also need to specify whether an increase of weight of the structure will increase or decrease this response.

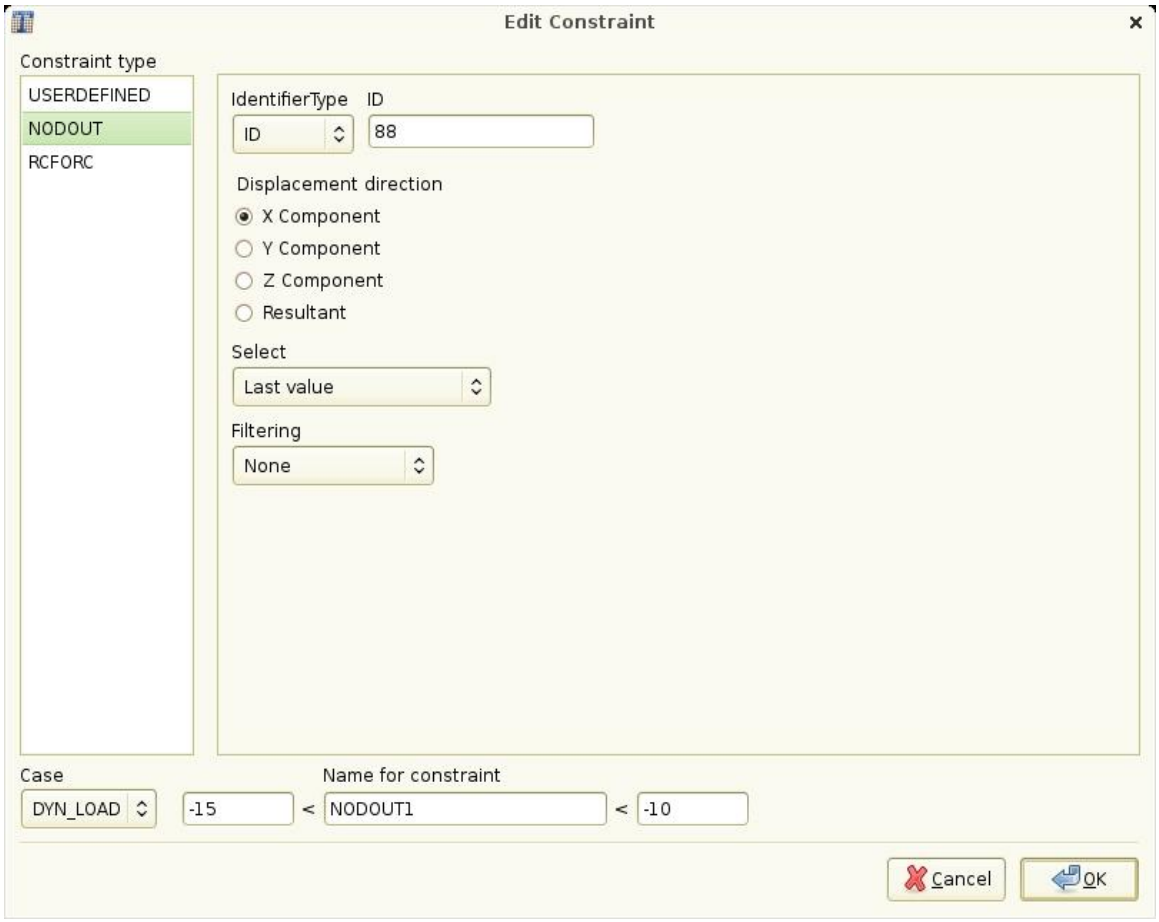

Figure 4-4: The constraints creation panel.

4.3.4. The Parts Panel

The part definition panel contains information about the parts to be designed, such as the geometry and mass fraction. See the following table, Figure 4-5 and Figure 4-6 for more details.

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.

`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. The default value depends on the average element size.
`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.

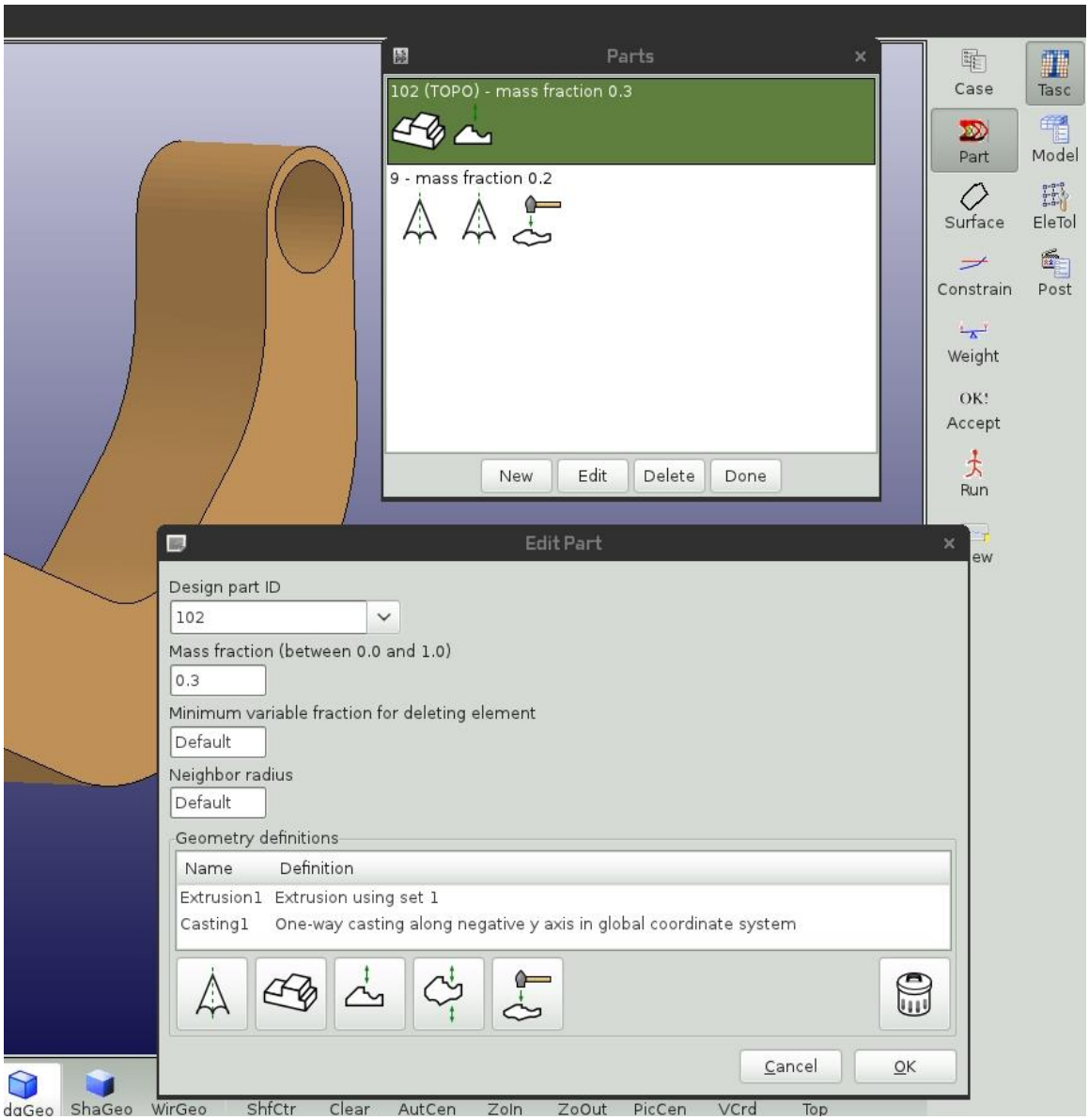

Figure 4-5: The parts panel.

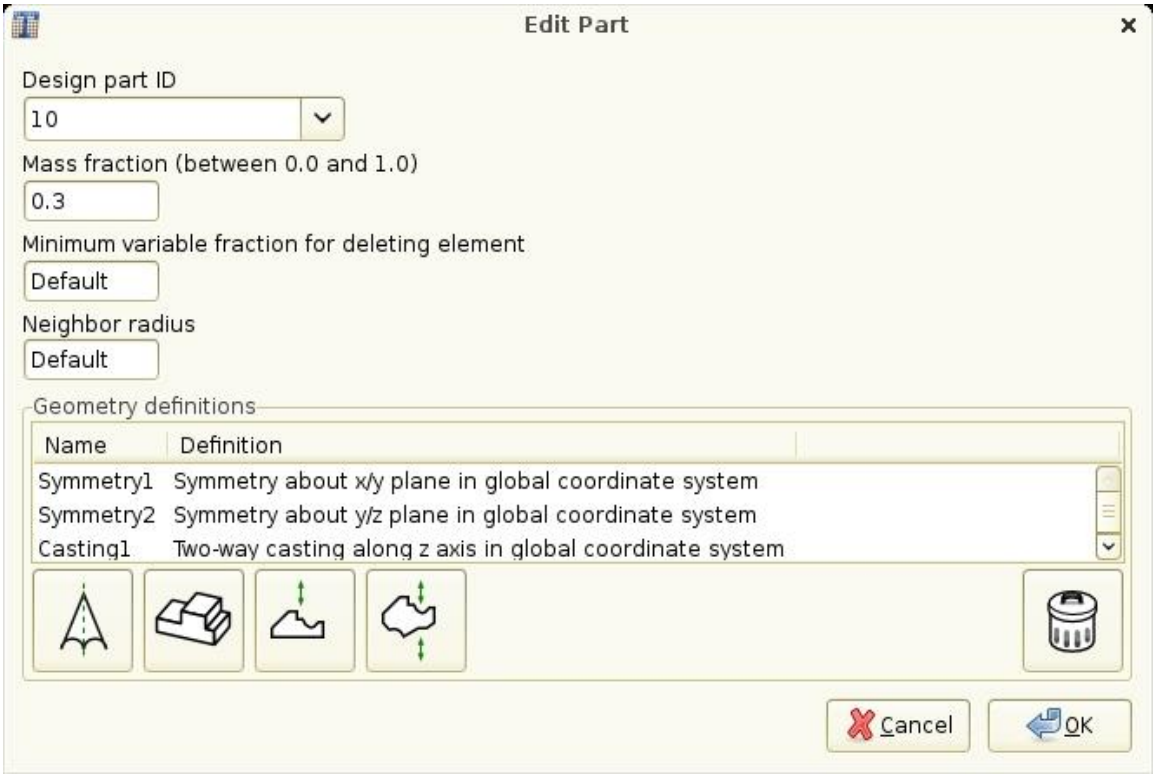

Figure 4-6: The panel to create part and geometry.

4.3.5. The Surface Panel

The shape definition panel contains information about the surfaces to be designed, such as the geometry. It is similar in function and layout to the parts panel shown in Figure 4-5.

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 value depends on the

	average element size.
`Move limit`	This is the maximum distance a node will be moved in an iteration.
`Remesh depth`	This the number of elements that should be considered in remeshing after a shape change was done.

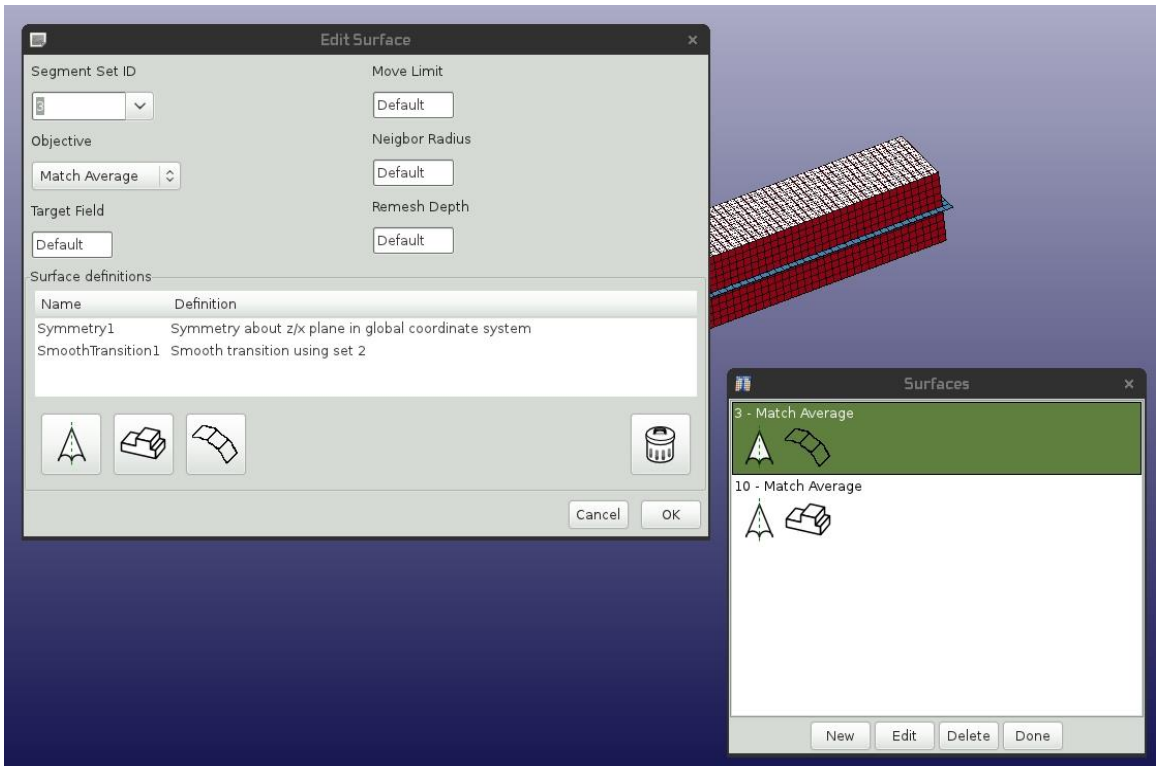

Figure 4-7: The surface panel.

4.3.6. Part and Surface Geometry

The geometric properties can be defined for every part and surface. See the following table and Figure 4-8 for more details.

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.
`Smooth transition Set ID`	To define a smooth transition, the user firstly creates a node set definition defining the edge. Allowable set definitions are

	*SET_NODE and *SET_NODE_LIST.
Symmetry Plane	Specify a symmetry plane to define symmetry.
Cast direction	A cast direction is required for a casting 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.

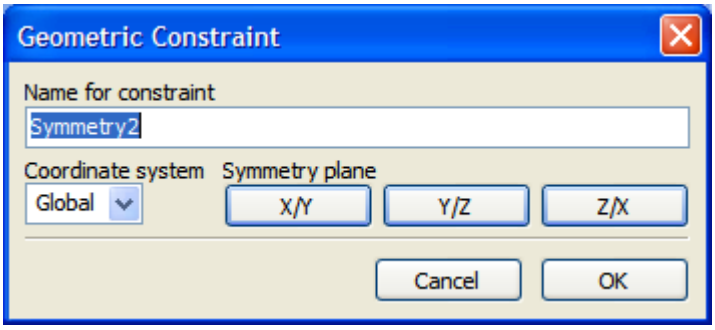

Figure 4-8: Creating a geometry constraint.

4.3.7. The Completion Panel

The completion panel specifies how the optimization problem will be solved. See the following table and Figure 4-9 for more details.

Completion data	
Number of design iterations	This is the maximum number of iterations allowed. The default value is 30.
Minimum mass redistribution	The minimum mass redistribution is the termination criterion used to stop the search when the topology has evolved sufficiently. This value is compared with the *Mass_Redistribution* history variable displayed in the view panel. The default value is 0.002.

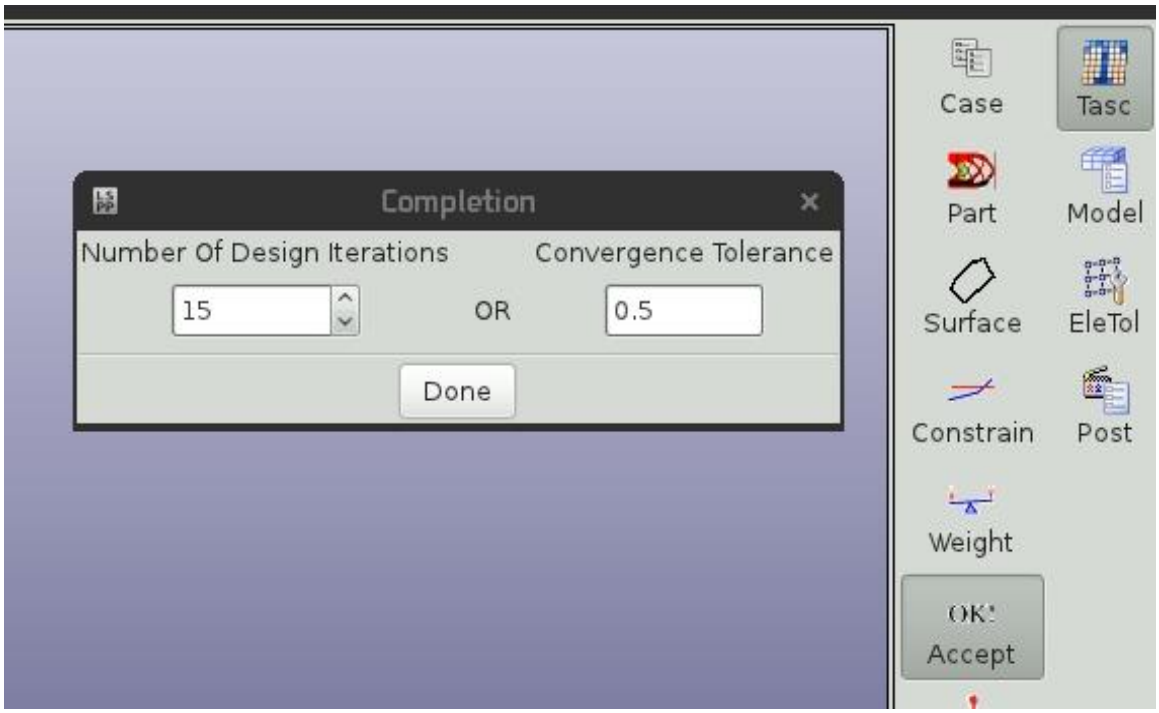

Figure 4-9: The completion panel.

4.3.8. The Run Panel

The control panel is used to submit the design problem. 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 4-11 for more details.

4.3.9. Setting advanced options

Advanced options can be set as shown in Figure 4-10. This is accessed through the *Settings* pulldown menu.

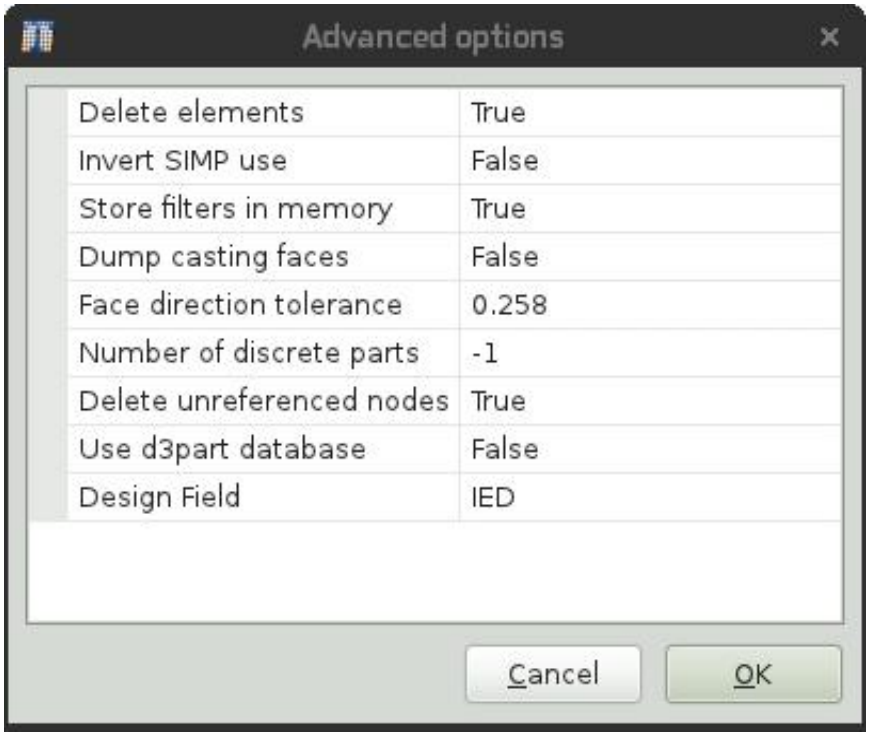

Figure 4-10 Options panel

The available options are described in the following table.

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 which can be viewed in LS-PrePost
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.
Design field	This is the criteria used to decide whether an element is utilized. One of Internal Energy Density or Von Mises.

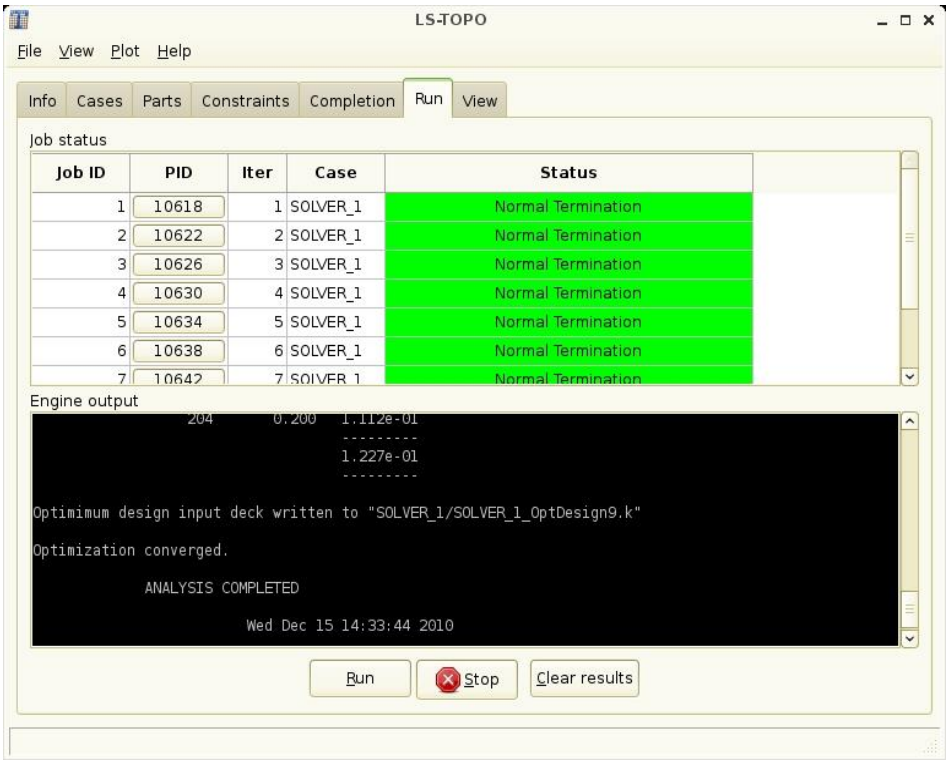

Figure 4-11: The run panel.

4.4. 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 4-12 and Figure 4-13 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.

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.
`P`*`123`*`_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).
`P`*`123`*`_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.

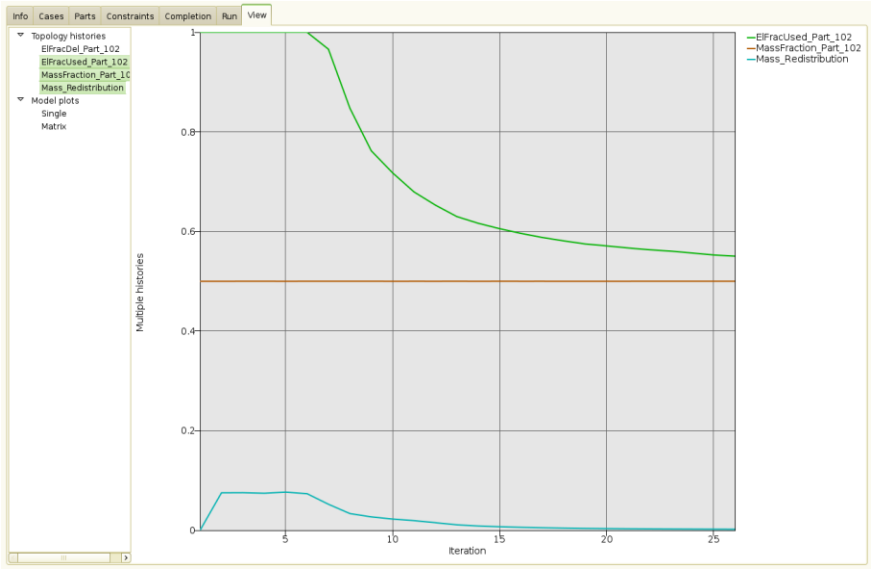

Figure 4-12: The view panel with histories.

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.

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.

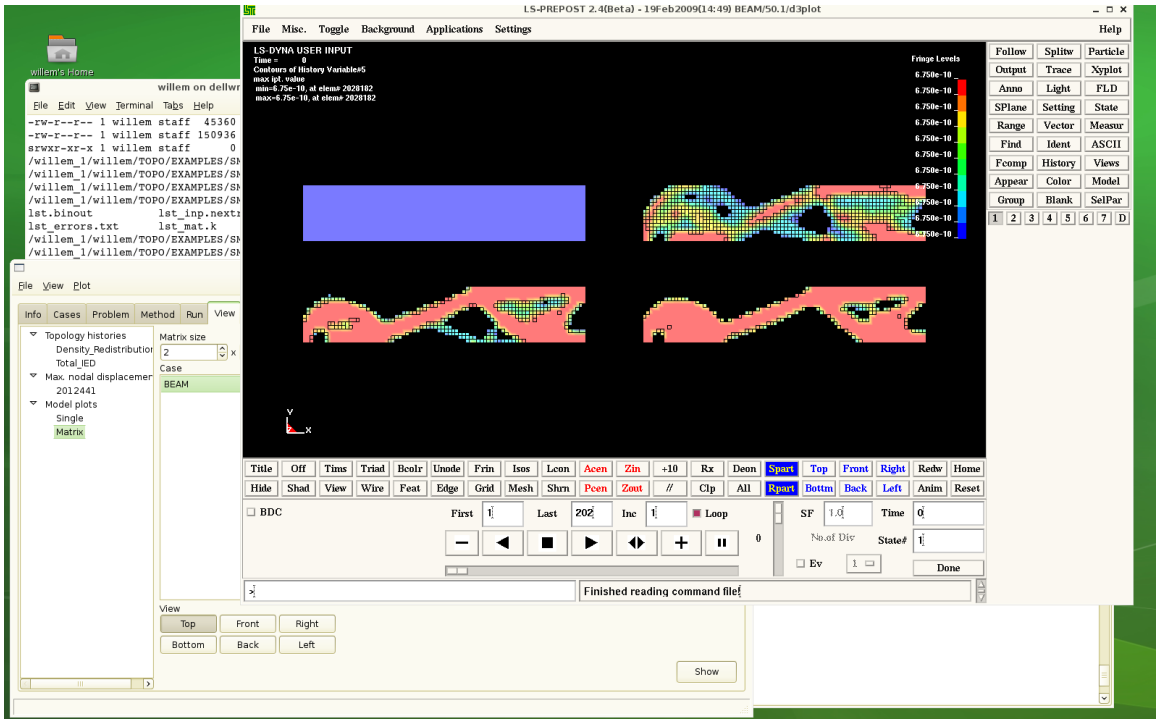

Figure 4-13: Viewing the model evolution in LS-PREPOST.

4.5. 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

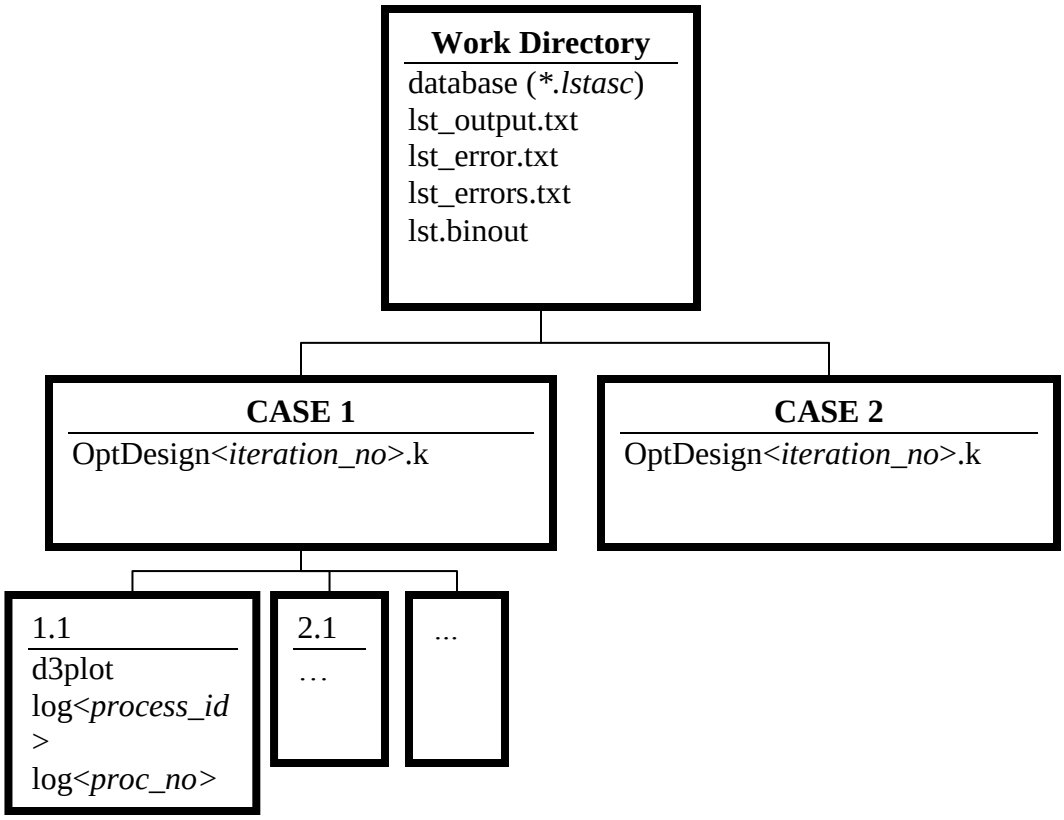

Figure 4-14 Directory structure

4.6. 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*.

4.7. 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.

4.8. Script Commands

The script commands issued to create the database can be viewed from the View pulldown menu. Use these commands as a template for scripts.

5. EXAMPLE PROBLEMS

The application of the topology code is demonstrated with the help of a few test examples below. The examples are supplied together with the software executables.

5.1. Fixed Beam with Central Load

This example demonstrates

1. how to define a problem,
2. how to add a case,
3. how to optimize the topology for a non-extrusion example, and the
4. analysis of output.

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. The material model used in this example is defined previously.

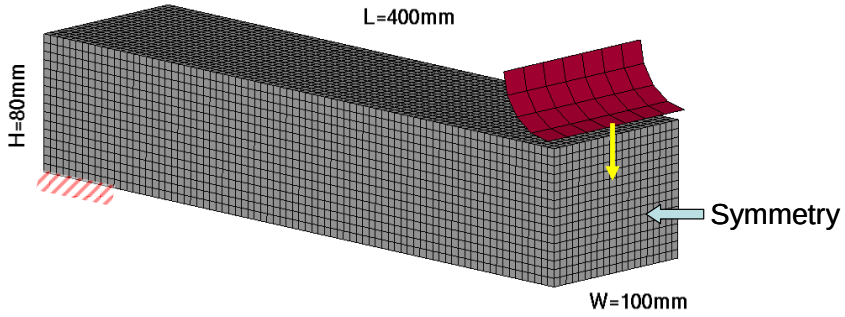

Figure 5-1: Geometry and loading condition of a single-load case example.

5.1.2. Input

The problem has a case named BEAM. The name of the DYNA input deck file is "Beam.dyn". Part 101 is the design part. A maximum of 100 iterations are used to find the optimal topology. The desired mass fraction is 0.25.

The project input data is saved to the file *lst_project.lstasc* as provided in the examples distribution. Additionally, scripts to recreate the database are also provided. The project database can be investigated using the scripts; use the script in example **Error! eference source not found.** to print the project data. The advanced user can conduct the

simulations using the LS-DYNA MPP version and hence using a script named "submit_pbs" for the PBS queuing system.

5.1.3. Output

The output of the code is written in the file named lst_output.txt. The error and warning messages are echoed in lst_error and lst_Warning files respectively. The typical output in the lst_output.txt is:

```
ls-dyna analysis time: 161s
it   1:   total IED: 9.933e+03   Mf: 0.250
ls-dyna analysis time: 177s
it   2:  total IED: 9.495e+03   Mf: 0.250  dX: 0.074627 (target: 0.001)
ls-dyna analysis time: 183s
it   3:  total IED: 8.983e+03   Mf: 0.250  dX: 0.077542 (target: 0.001)
ls-dyna analysis time: 187s
it   4:  total IED: 9.252e+03   Mf: 0.250  dX: 0.072176 (target: 0.001)
ls-dyna analysis time: 193s
it   5:  total IED: 9.156e+03   Mf: 0.250  dX: 0.063345 (target: 0.001)
ls-dyna analysis time: 193s
```

a) *Convergence History*

The convergence is quantified using the change in topology, characterized by the normalized density redistribution, and the total internal energy density as shown in Figure 5-2.

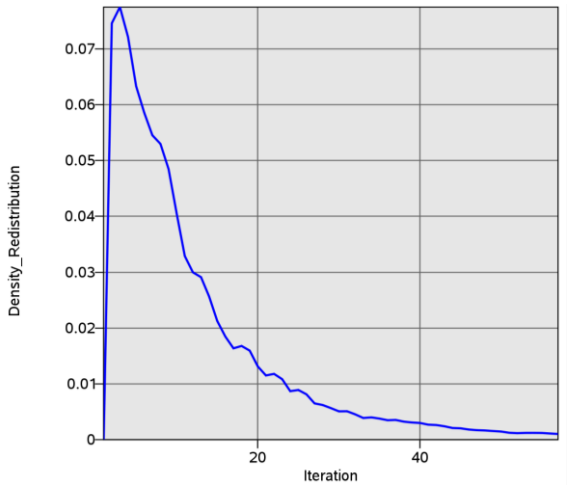

Figure 5-2: Convergence history of the mass redistribution.

The simulation converged after 57 iterations. It was observed that initially there were significant changes in the topology (upto 30 iterations). Afterwards, small changes were made in the topology. There was a drop in the total internal energy density during the early phase of the optimization but it increased during the later iterations. The final topology is visualized in LS-PREPOST.

b) ***Density Contours***

The initial and final topologies are shown in Figure 5-3, and the topologies at different iterations during the evolution process are shown in Figure 5-4.

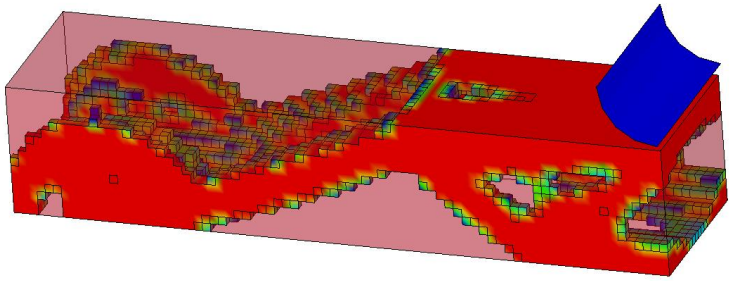

Figure 5-3: Initial and final density contours.

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.

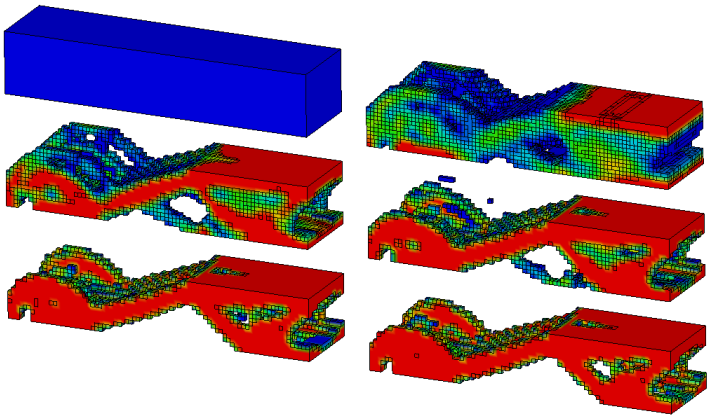

Figure 5-4: Evolution of the geometry shown using density contours.

Topologies at different stages of the evolution process show that the main features of the structure were evolved by iteration 20 (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.

5.2. Beam using geometry definitions

This example demonstrates how to set up a problem with geometry definitions.

The same fixed-beam with a central load example is analyzed with an extrusion and two 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.

5.2.1. Input

The main differences in this example compared to the non-extrusion example are:

- An extrusion definition is provided.
- A casting definition in Z direction is provided.

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, scripts to recreate the database are also provided. The project database can be investigated using the GUI or a script; use the script in example **Error! Reference ource not found.** to print the project data.

5.2.2. Output

a) ***Extrusion and Casting***

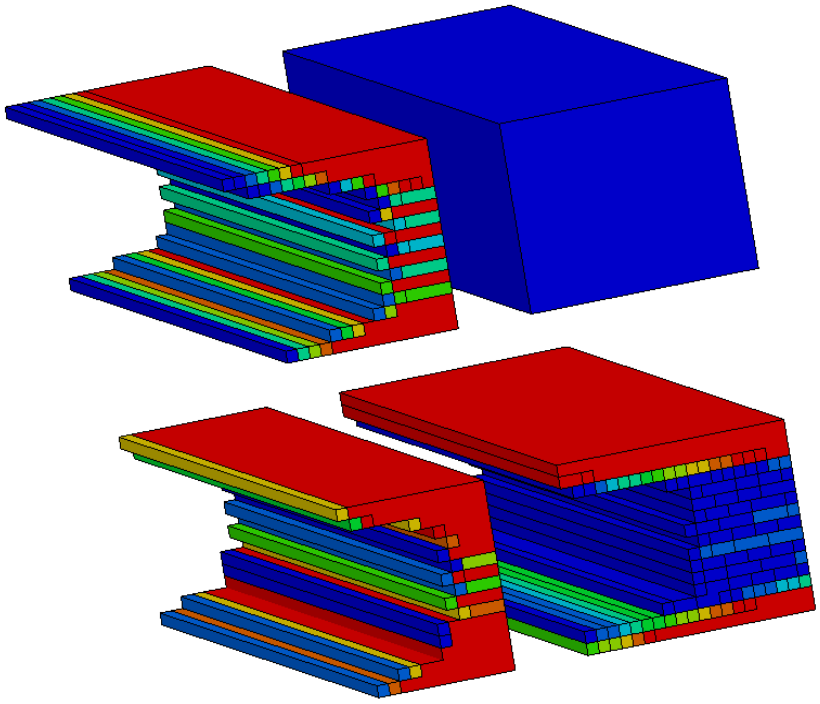

Figure 5-5: Evolution of the beam using extrusion and single-sided casting constraints

Different phases in the evolution are depicted in Figure 5-5. 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.

b) ***Extrusion and two-sided casting***

Different phases in the evolution are depicted in Figure 5-5. 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.

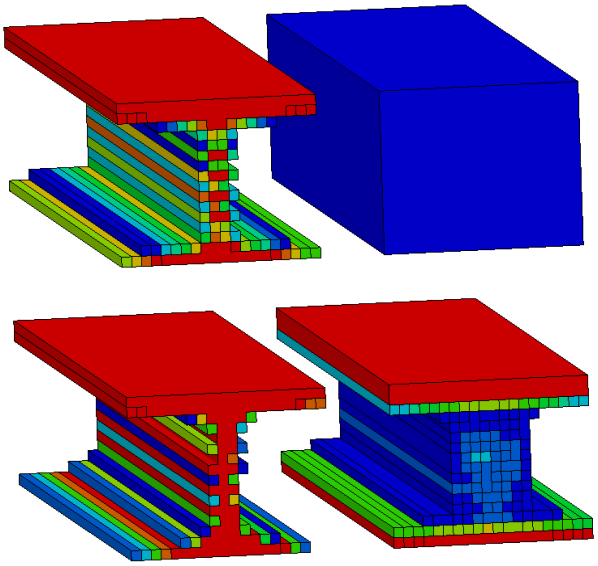

Figure 5-6: Evolution of the beam using extrusion and two-sided casting constraints.

5.3. Force-Displacement Constraints

The next example demonstrates a simulation with multiple constraints.

5.3.1. Problem Definition

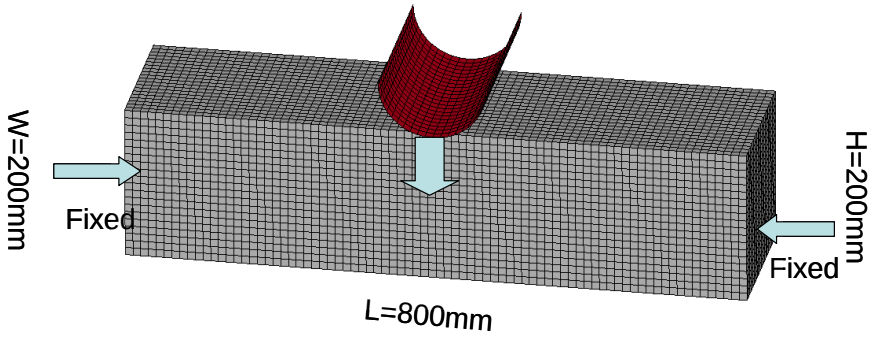

Figure 5-7: The geometry and loading conditions of the multiple constraints example.

The geometry and loading conditions for the example are shown in Figure 5-7. This is a fixed-fixed beam with a central load. The design part was meshed with $10mm^3$ elements.

5.3.2. Input

The center load was assigned at the location of the pole hitting the beam. The desired mass fraction for this example was 0.25. A maximum of 100 iterations were allowed. 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.

The project input data is saved to the file *lst_project.lstasc* as provided in the examples distribution. Additionally, scripts to recreate the database are also provided. The project database can be investigated using the scripts; use the script in example **Error! eference source not found.** to print the project data. The advanced user can conduct the simulations using the LS-DYNA MPP version and hence using a script named "submit_pbs" for the PBS queuing system.

5.3.3. Output

a) *Convergence History*

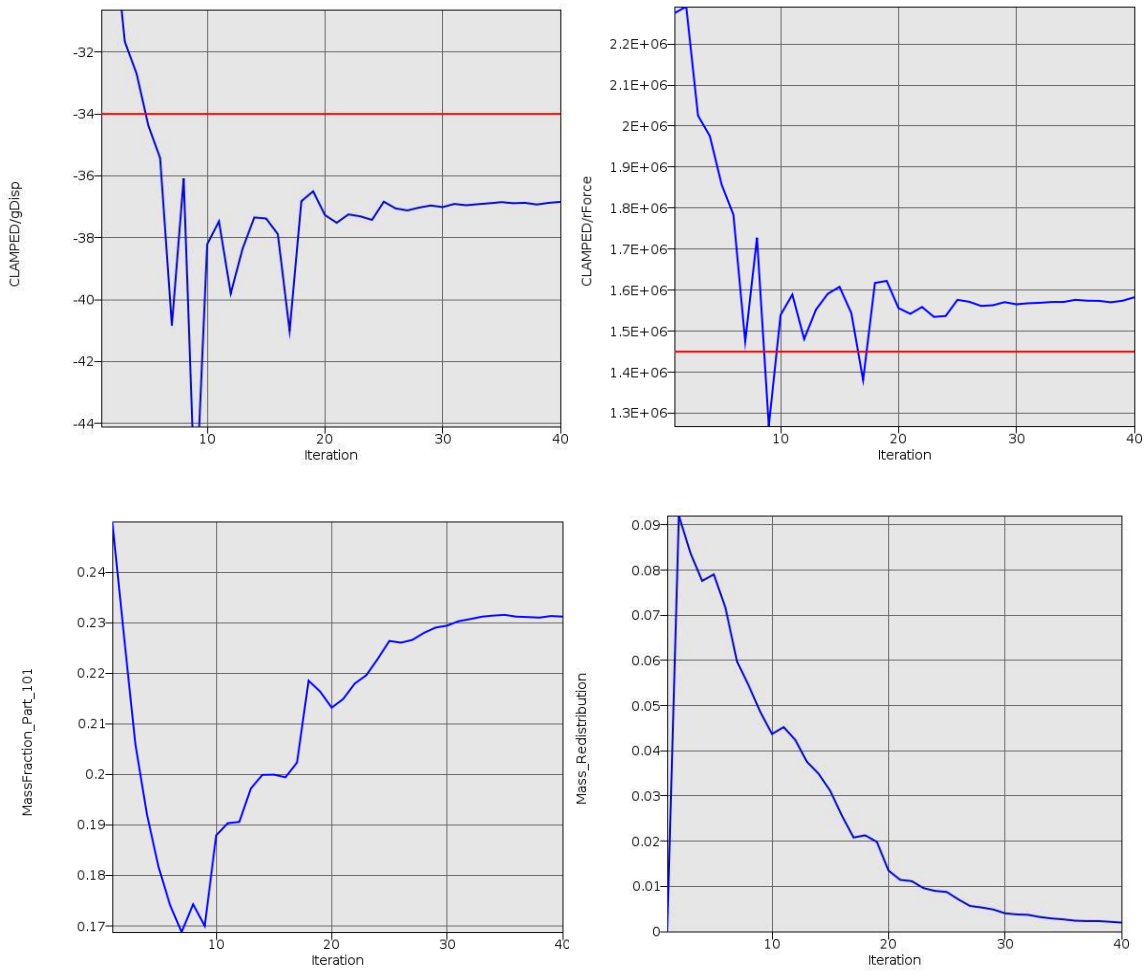

Figure 5-8: Convergence history for the example with multiple constraints.

The convergence history for the multiple-constraints example is shown in Figure 5-8. There were minimal changes in the geometry after 25 iterations and the simulation converged after 40 iterations. While there was largely monotonic reduction in the density redistribution, the constraints and IED 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.

b) ***Density Contours***

The evolution of the topology of the clamped beam with multiple constraints is shown in Figure 5-9. The final structure had many cavities and resembled an optimized truss-like structure. The main cavities in the structure were formulated by the 15th iteration and the structure was fully developed in a largely 0-1 type structure by the 30th iteration. Further redistribution of the material refined this structure between the 30th and the 40th iteration.

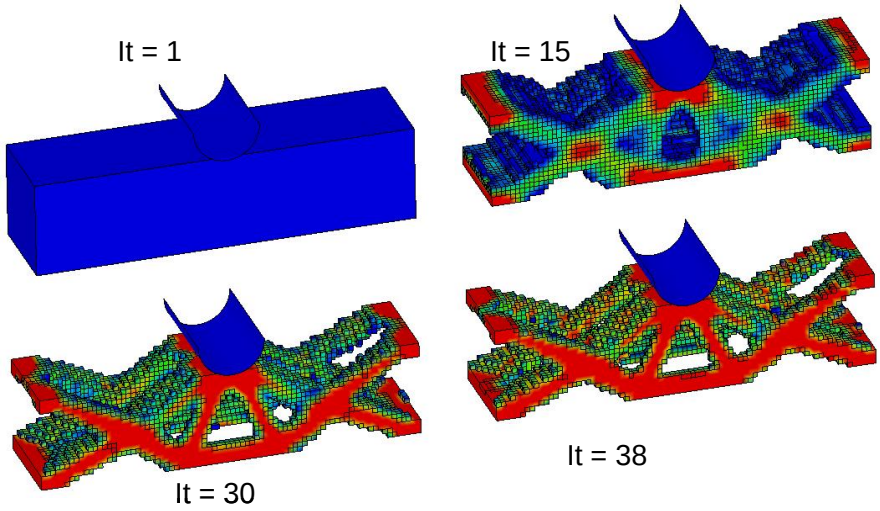

Figure 5-9: Evolution of the geometry for multiple-constrained clamped beam.

5.4. Linear Static Loading

The next example demonstrates the topology optimization of a statically loaded structure.

5.4.1. Problem Definition

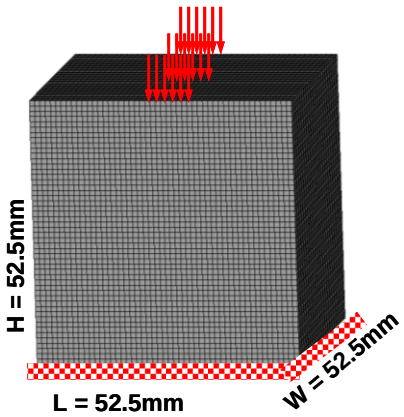

Figure 5-10: The geometry and loading conditions of a statically loaded structure.

The geometry and loading conditions for the example are shown in Figure 5-23. The design part was meshed with 1.05mm^3 elements such that there were approximately 125,000 elements.

5.4.2. Input

In this example, a unit load is applied in the center of the structure. The structure was fixed on the bottom. The problem has a case named *TopLoad*. The simulations are carried out using the double precision SMP version of LS-DYNA (*ls971_double*). The name of the DYNA input deck file is "*LinearStructure.dyn*". Part 102 is the design part. A maximum of 100 iterations are used to find the optimal topology and the desired mass fraction is 0.30.

The project input data is saved to the file *lst_project.lstasc* as provided in the examples distribution. Additionally, scripts to recreate the database are also provided. The project database can be investigated using the scripts; use the script in example **Error! eference source not found.** to print the project data.

5.4.3. Output

a) *Convergence History*

The convergence history for the statically loaded structure topology optimization example is shown in Figure 5-11. The simulation converged after 28 iterations, though only minor changes were noted after 20 iterations. As observed before, monotonic reduction in the change in topology was observed. The total internal energy of the structure also decreased with topology evolution.

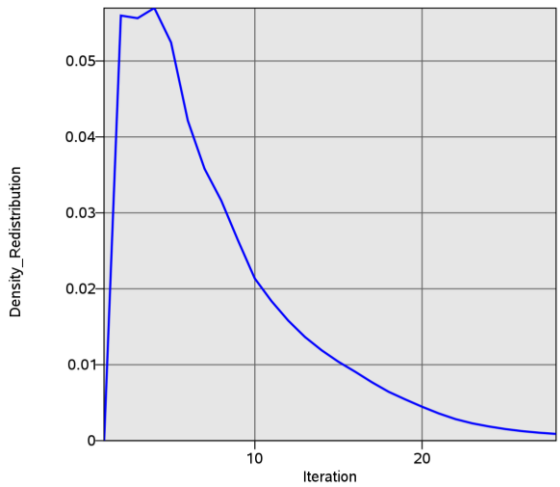

Figure 5-11: Convergence history for linear-static example.

b) ***Density Contours***

The initial and final structures are shown in Figure 5-12. 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.

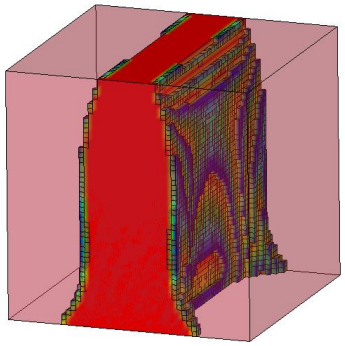

Figure 5-12: Initial and final density contours.

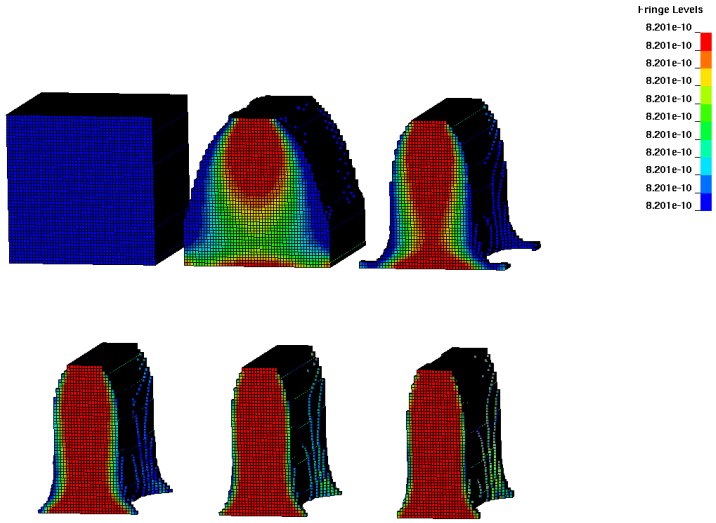

Figure 5-13: Evolution of the geometry for statically loaded structure.

The evolution of the topology under the static loading conditions is shown in Figure 5-13. While the final form of the structure was largely evolved by 17th 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 28 iterations.

5.5. Shell Example

This example shows how to work with shell structures.

5.5.1. Problem Definition

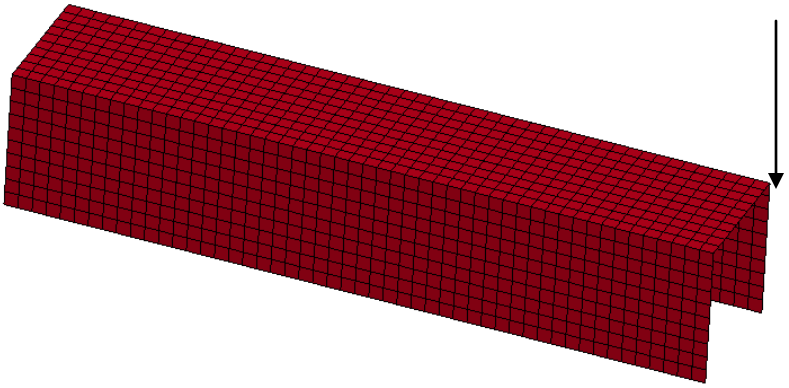

Figure 5-14: The geometry and loading conditions of the shell example. The left side is built-in, while a downward load is applied to the right, back corner.

The geometry and loading conditions for the example are shown in Figure 5-14.

5.5.2. Input

The project input data is saved to the file *Shell.lstasc* as provided in the examples distribution. Additionally, scripts to recreate the database are also provided. The project database can be investigated using the scripts; use the script in example **Error! eference source not found.** to print the project data.

5.5.3. Output

a) ***Convergence History***

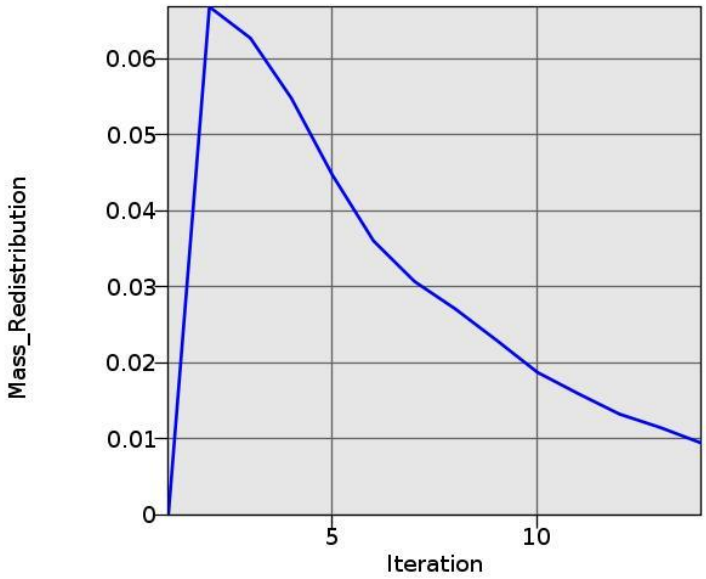

Figure 5-15: Convergence history for the shell example.

The convergence history for the shell example is shown in Figure 5-15. The simulation converged after 14 iterations. There was largely monotonic reduction in the density redistribution.

b) *Final Shell Thicknesses*

The final design is shown in Figure 5-16. The final structure had many cutouts and resembled an optimized truss-like structure.

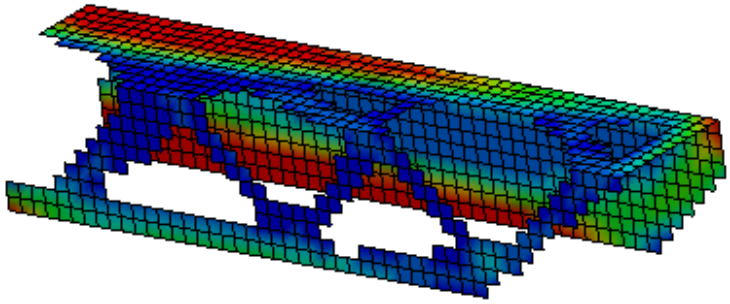

Figure 5-16: Final geometry and thicknesses for the shell problem.

5.6. Multiple Load Cases

This example demonstrates

1. multiple load cases,
2. dynamic weighing of load cases,
3. constraints, and a

4. symmetry geometry definition.

5.6.1. Problem Definition

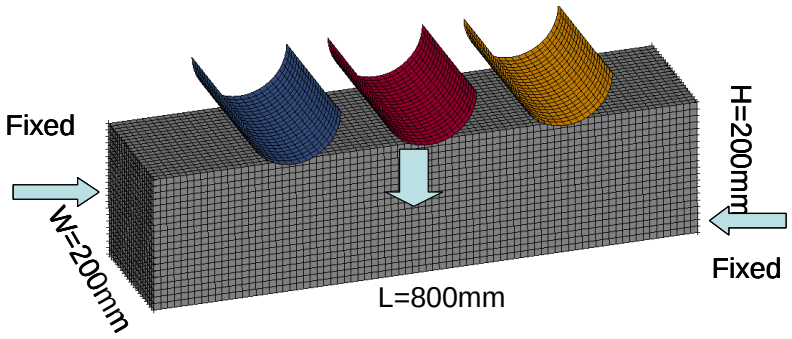

Figure 5-17: The geometry and loading conditions of the multiple load case example.

The geometry and loading conditions for the example are shown in Figure 5-17. 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.

5.6.2. Input

The problem is symmetric, so symmetry is defined and only two load cases are therefore used. The desired mass fraction for this example is 0.3. A maximum of 50 iterations are allowed. All simulations are run simultaneously.

The displacements for both load cases are constrained to be less than 110. The locations are the center of impact and the maximum value over time was selected.

The problem is analyzed using with and without dynamic scaling of results. For the use of the dynamic scaling, the two selected maximum displacements are required to be the same. With dynamic scaling, the all load cases are assigned a unit weight.

All of the details can be found in in the examples distribution in the MLC directory.

5.6.3. Results with constant weights

The results are as shown in Figure 5-18 to Figure 5-20. 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 5-18.

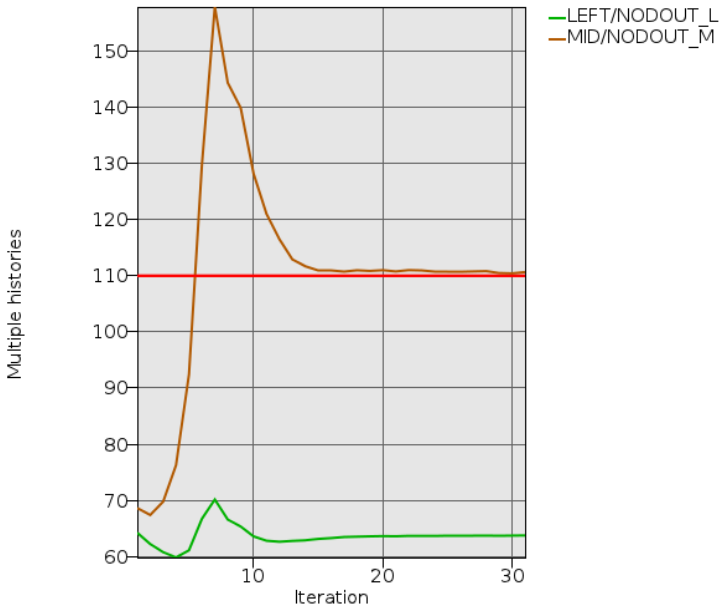

Figure 5-18 Constraint convergence history for multiple-load case example with constant weights.

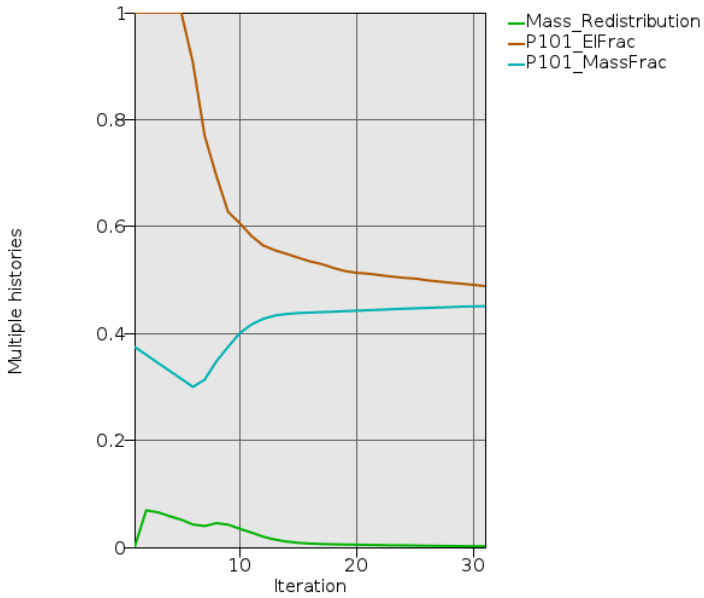

Figure 5-19 Various histories of the load case weight for multiple-load case example using with constant weights: mass redistribution, the fraction of elements kept, and the mass fraction.

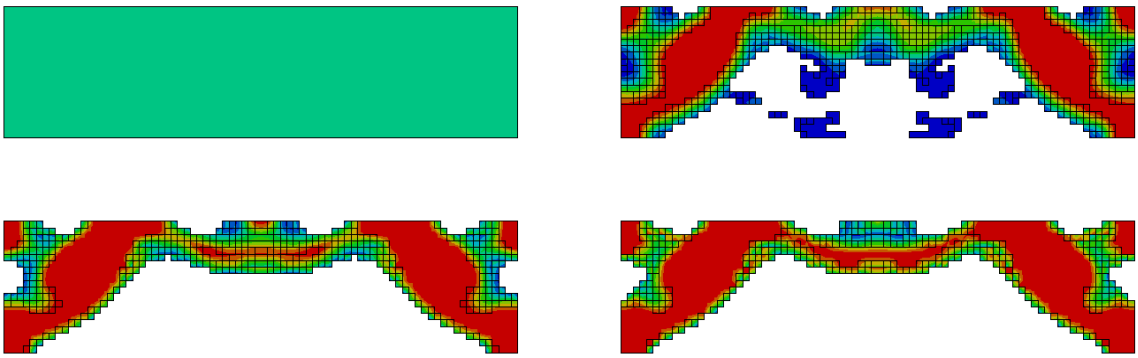

Figure 5-20 Evolution of the geometry for multiple-load case structure using constant weights.

5.6.4. Results with dynamic weighing

The convergence history for the multiple-load example is shown in Figure 5-21. The simulation converged after 46 iterations. Results are much improved by the dynamic weighing. The constraints are reasonably close to the bound as shown in Figure 5-21 due to the load case weighing computed also shown.

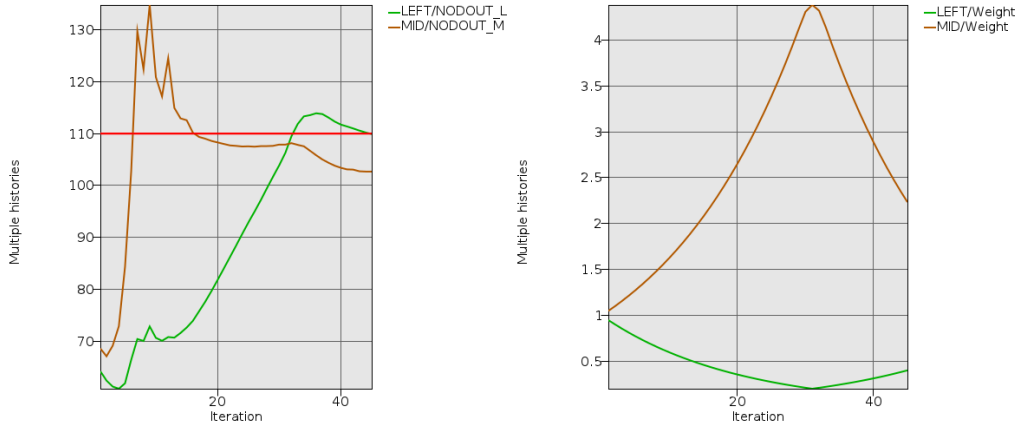

Figure 5-21: Constraint convergence history for multiple-load case example using dynamic weighing is shown on the left. Note the improvement with respect to not using dynamic weighing. The corresponding weight factors are shown on the right.

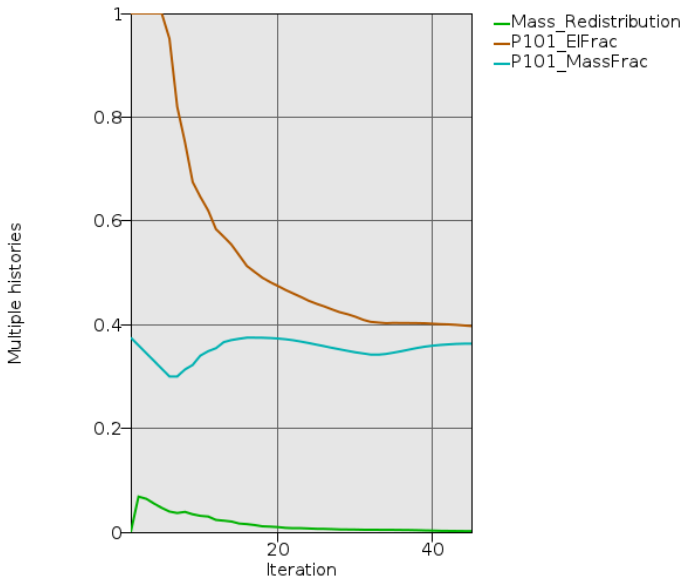

Figure 5-22 Various histories of the load case weight for multiple-load case example using dynamic weighing: mass redistribution, the fraction of elements kept, and the mass fraction.

The evolution of the topology under multiple loading conditions is shown in Figure 5-23. 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.

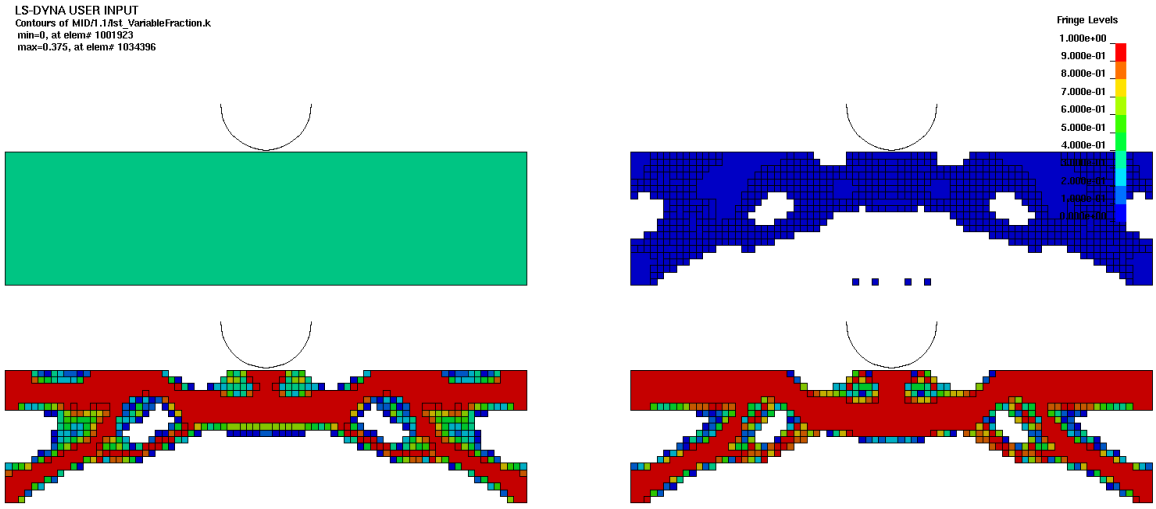

Figure 5-23: Evolution of the geometry for multiple-load case structure using dynamic scaling of the weights. The design is improved with respect to not using dynamic weighing by strengthening the portion of the structure carrying the center load.

5.7. Surface Design of a Beam

This example demonstrates:

1. Free surface design for solids
2. Geometry constraints for free surface design

5.7.1. Problem Definition

The geometry and loading conditions for the example are shown in Figure 5-24.

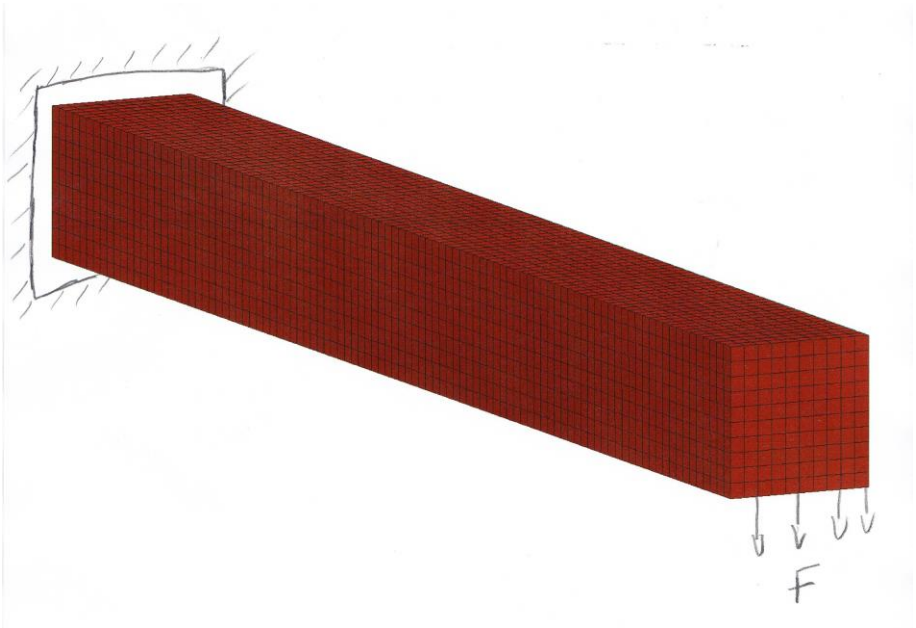

Figure 5-24 Beam model for free surface design

5.7.2. Results with four surfaces

All four sides of the beam were selected for shape design. The convergence tolerance for this example is a 50% smoothing. The problem converged in 8 iterations.

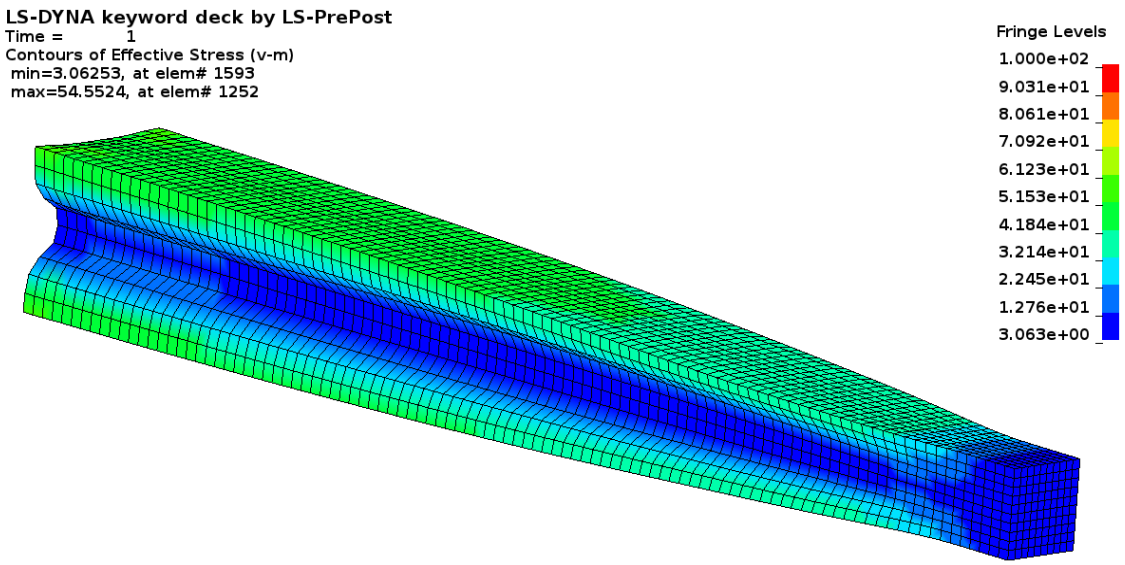

Figure 5-25 Final design for four surfaces

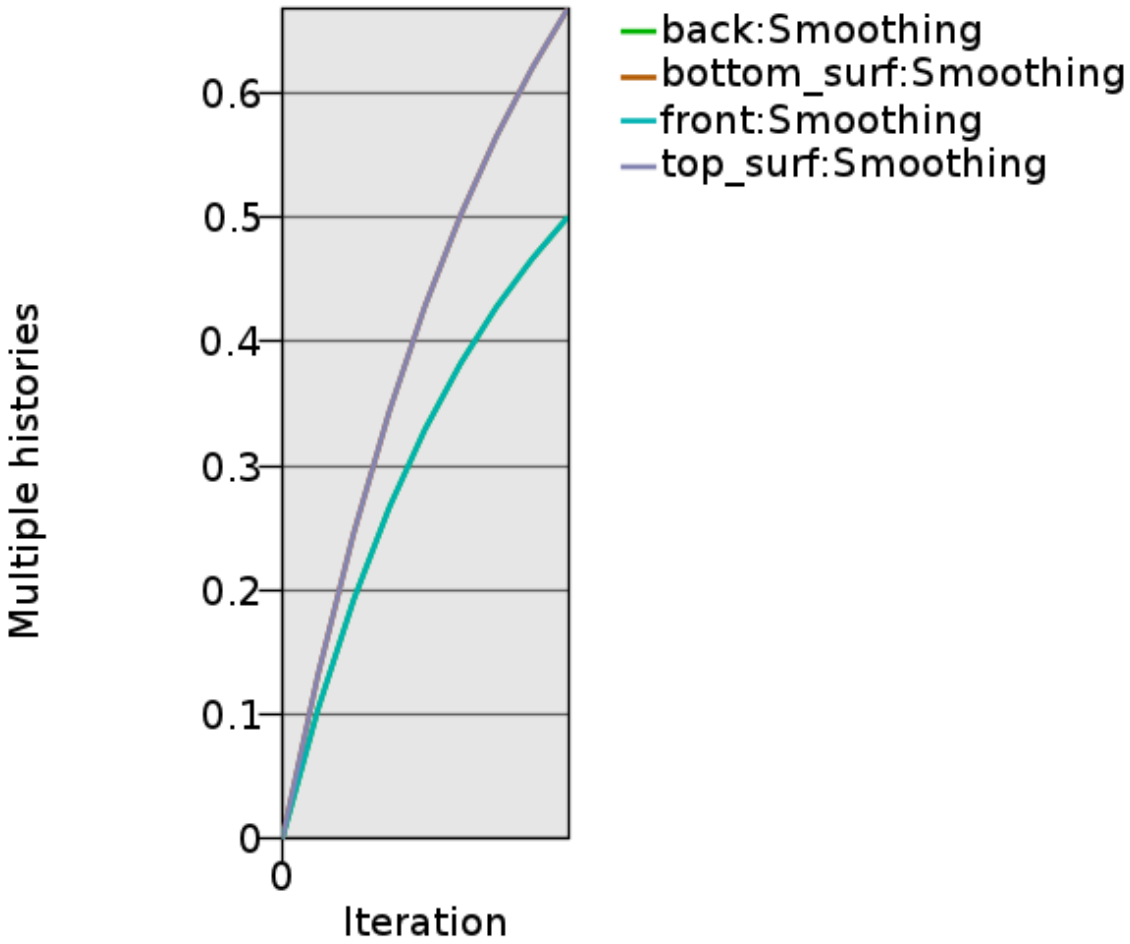

Figure 5-26 Convergence history

5.7.3. Results with extrusion and symmetry geometry definitions

The front and back side of the beam were selected for shape design. The convergence tolerance for this example is a 50% smoothing. The problem converged in 27 iterations. The final design is shown in Figure 5-27. 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.

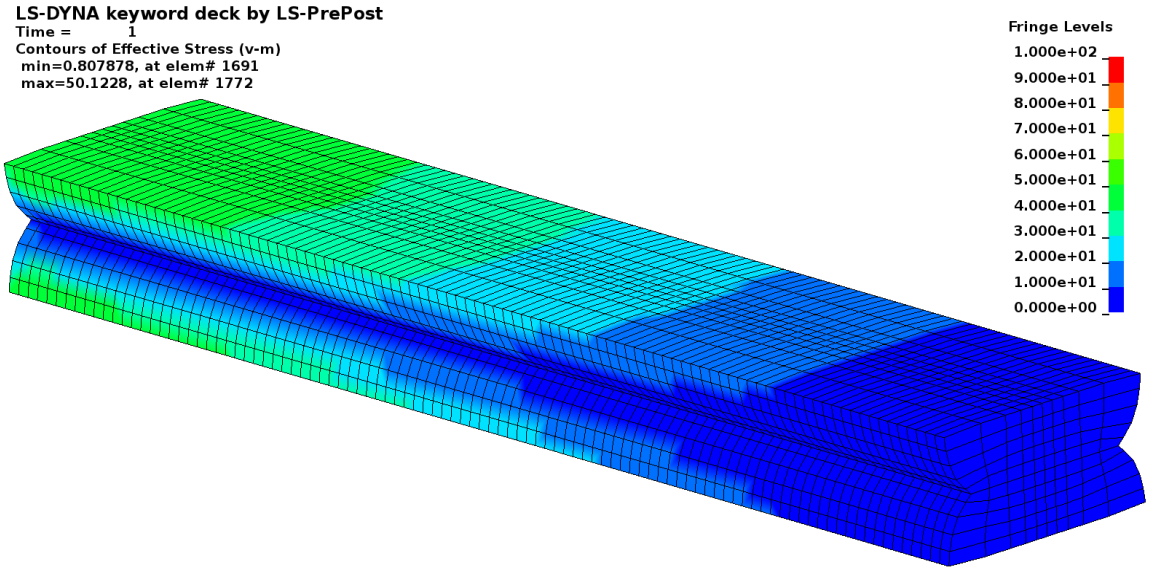

Figure 5-27 Final design of beam with extrusion and symmetry geometry definitions

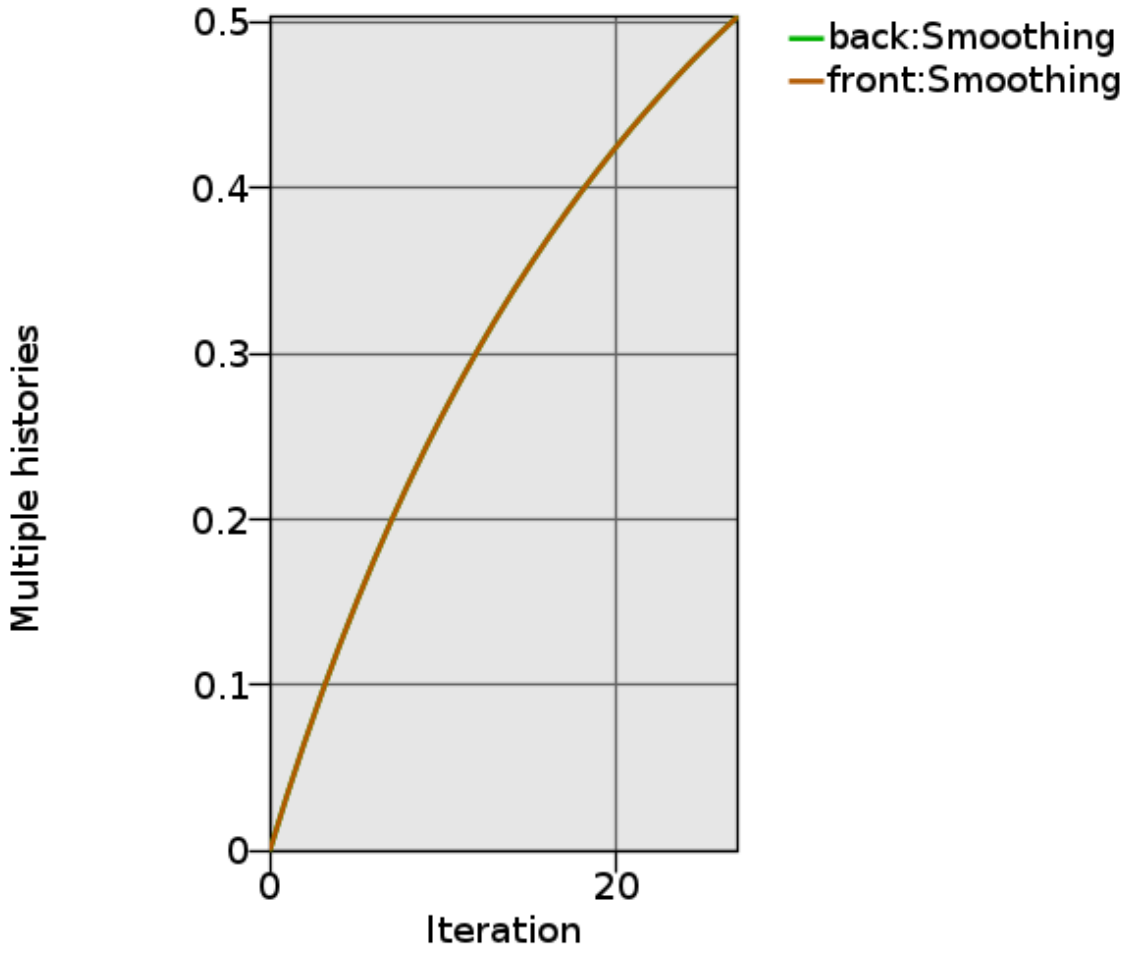

Figure 5-28 Convergence history of beam with extrusion and symmetry geometry definitions

5.7.4. Results with smooth transition geometry definition

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. Use LS-TaSC to investigate the problem definition visually. The objective was the minimum volume of the part. The convergence tolerance for this example is a 50% smoothing with a maximum of 30 iterations allowed. The final design is as shown in Figure 5-29. The design without the smooth transition is shown in Figure 5-31 – the resulting poor mesh quality can be seen.

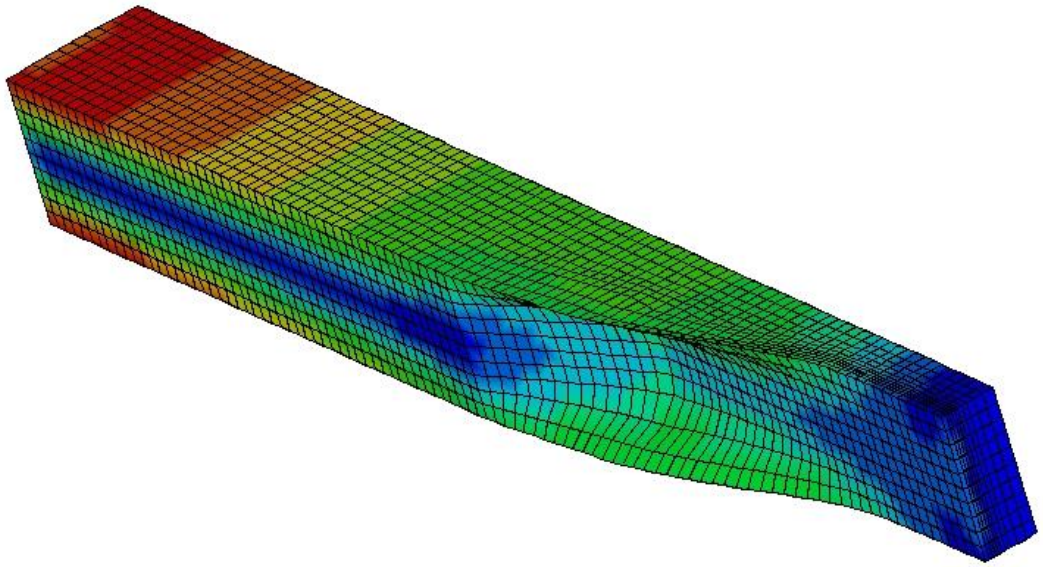

Figure 5-29 Final design of beam with smooth transition geometry definition

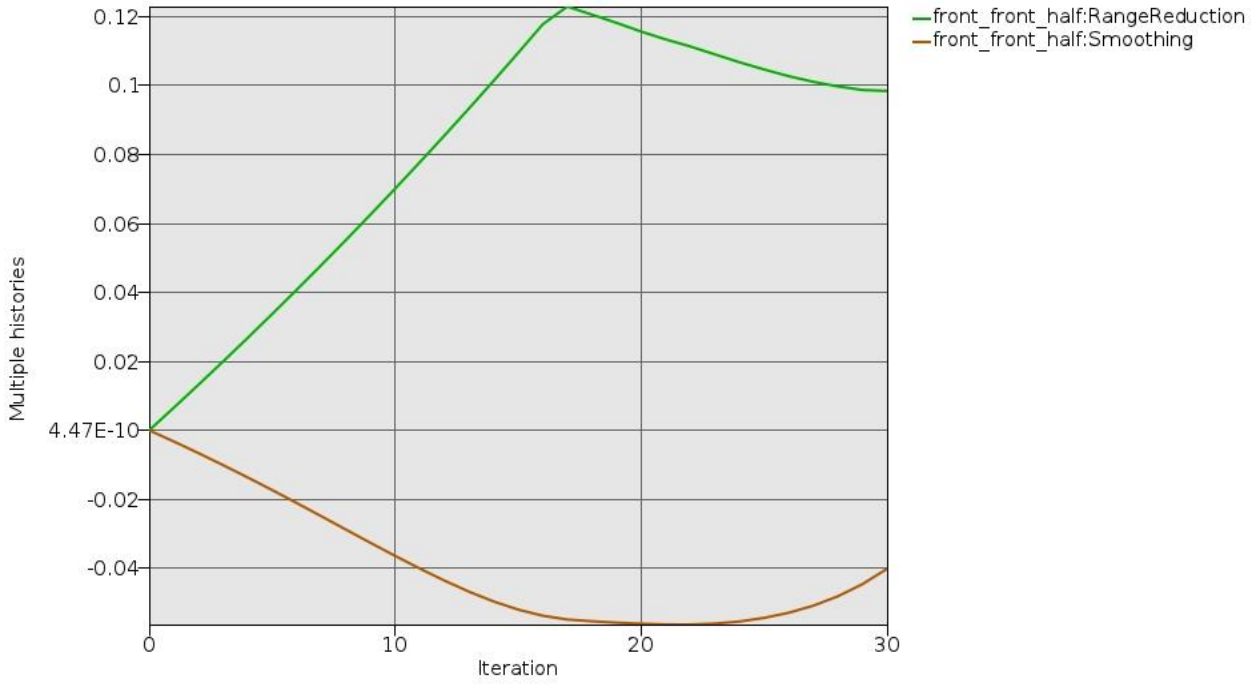

Figure 5-30 Convergence history of beam with smooth transition geometry definition

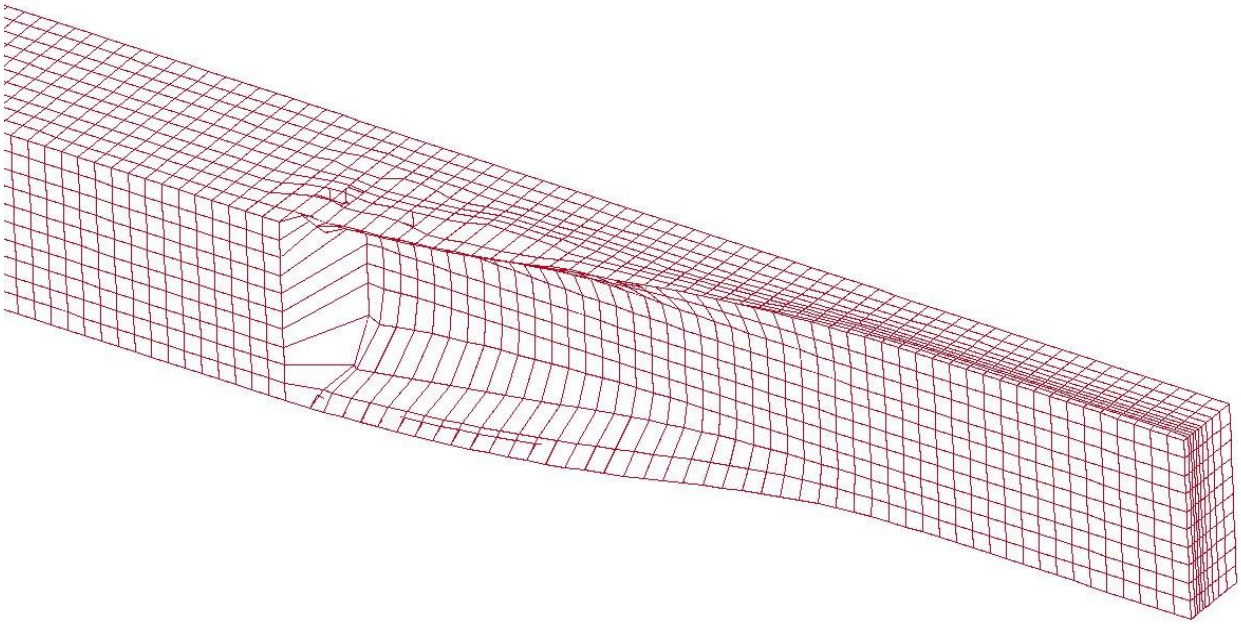

Figure 5-31Design of beam without smooth transition geometry definition

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.
1. 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.
2. Specify SOFT=2 option on the control card
3. 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. OTHER LS-TASC MANUALS

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

7.1. Theory manual

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

7.2. Scripting manual

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

7.3. Queueing system installation

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