User’s Manual

INTEGRATED COMPUTATIONAL MATERIALS ENGINEERING

USING LS-DYNA® AND LS-OPT®

ANIRBAN BASUDHAR, Ph.D.
USHNISH BASU, Ph.D.
NIELEN STANDER, Ph.D.
IMTIAZ GANDIKOTA, M.S.

March 2017

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
LS-OPT support site: www.lsoptsupport.com
LS-OPT Users Group:
http://groups.google.com/group/lsopt_user_group

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

2/22/2017

TABLE OF CONTENTS

1. Introduction

Ever-tightening regulations on fuel economy, and the likely future regulation of carbon emissions, demand persistent innovation in vehicle design to reduce vehicle mass. Classical methods for computational mass reduction include sizing, shape and topology optimization. One of the few remaining options for weight reduction can be found in materials engineering and material design optimization. Apart from considering different types of existing materials, by adding material diversity and composite materials, an appealing option in automotive design is to engineer new steel alloys with desired properties. The new steels can then be used to reduce the plate thickness while retaining sufficient strength and ductility required for durability and safety.

During the period 2013-16, a project to develop computational material models for advanced high strength steel was executed under the auspices of the United States Automotive Materials Partnership (USAMP) funded by the US Department of Energy. Under this program, new Third Generation Advanced High Strength Steel i.e. 3GAHSS were being designed, tested and integrated with the remaining design variables of a benchmark vehicle Finite Element model. The original objectives of the project were to integrate atomistic, microstructural, forming and performance models to create an integrated computational materials engineering (ICME) toolkit for 3GAHSS.

The mechanical properties of Advanced High Strength Steels (AHSS) are controlled by many factors, including phase composition and distribution in the overall microstructure, volume fraction, size and morphology of phase constituents as well as stability of the metastable retained austenite phase. The complex phase transformation and deformation mechanisms in these steels make the well-established traditional techniques obsolete, and a multi-scale microstructure-based modeling approach following the ICME strategy was therefore chosen in this project.

Multi-scale modeling as a major area of research and development is an outgrowth of the Comprehensive Test Ban Treaty of 1996 which banned surface testing of nuclear devices [2]. This had the effect that experimental work was reduced from large scale tests to multiscale experiments to provide material models with validation at different length scales. In the subsequent years industry realized that multi-scale modeling and simulation-based design were transferable to the design optimization of any structural system.

Horstemeyer [2] lists a number of advantages of the use of multiscale modeling. Among these are: the reduction of product development time by alleviating costly trial-and-error iterations as well as the reduction of product costs through innovations in material, product and process designs. Multi-scale modeling can reduce the number of costly large scale experiments and can increase product quality by providing more accurate predictions. Research tends to be focussed on each particular length scale, which enhances accuracy in the long term.

This User's Manual serves as a guide to software enhancements of LS-DYNA® and LS-OPT® conducted for this project. It mainly focuses on new capabilities to integrate material identification using material models of different length scales as well as the integration of material identification with product optimization.

1.1. The ICME model

The ICME model developed in this project consists of two material models with different length scales – a microscopic Crystal Plasticity (CP) model [3] and a macroscopic State Variable (SV) model [4]. These material models were implemented as user-defined materials in LS-DYNA® , and were integrated with each other as well as with design optimization using LS-OPT®. The overall ICME process, summarized in Figure 1-1, consists of multiple steps with various interdependencies. For instance, a forming analysis depends on the calibrated parameters of the two material models. These steps are, therefore, sequential. Additionally, some of the steps (e.g. material calibration), involve not just a single simulation but a nested optimization. Such steps of the overall process flow, which cannot be solved using a single analysis and may involve several substeps themselves, are referred to as multilevel steps. In addition, the vehicle design consists of different disciplines or load cases. Thus, the entire ICME model is a multilevel as well as a multidisciplinary process.

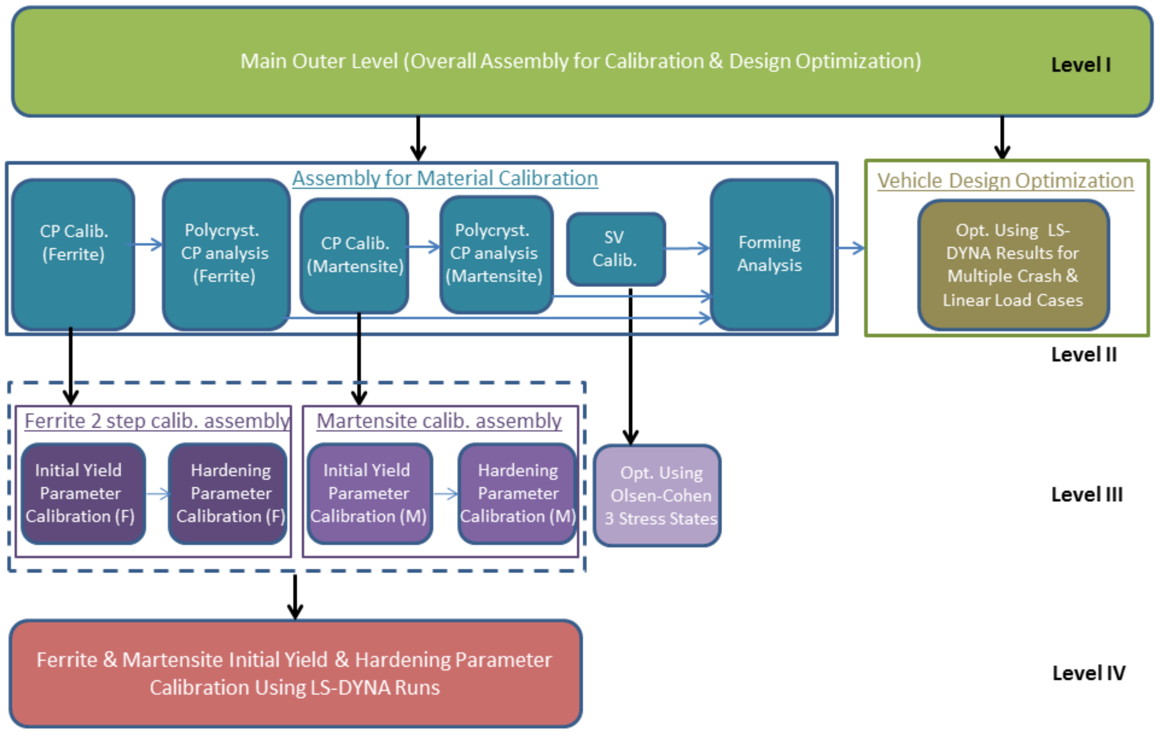
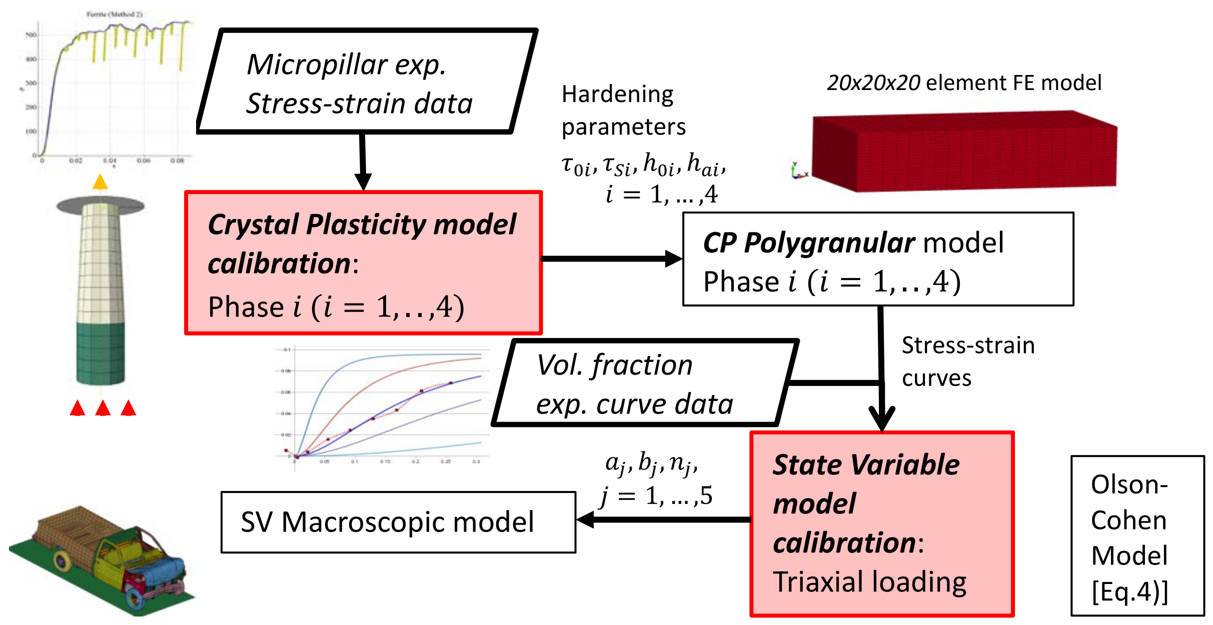

Main Outer Level (Overall Assembly for Calibration & Design Optimization)
Level I
Assembly for Material Calibration
CP Calib. (Ferrite)
Polycryst. CP analysis (Ferrite)
CP Calib. (Martensite)
Polycryst. CP analysis (Martensite)
SV Calib.
Forming Analysis
Vehicle Design Optimization
Opt. Using LS-DYNA Results for Multiple Crash & Linear Load Cases
Level II
Ferrite 2 step calib. assembly
Initial Yield Parameter Calibration (F)
Hardening Parameter Calibration (F)
Martensite calib. assembly
Initial Yield Parameter Calibration (M)
Hardening Parameter Calibration (M)
Opt. Using Olsen-Cohen 3 Stress States
Level III
Ferrite & Martensite Initial Yield & Hardening Parameter Calibration Using LS-DYNA Runs
Level IV

Figure 1-1: Multilevel structure of Integrated Computational Materials Engineering (ICME)



Micropillar exp. Stress-strain data
Hardening parameters
$\tau_{0i}, \tau_{Si}, h_{0i}, h_{ai},$ $i = 1, \ldots, 4$
20x20x20 element FE model
Crystal Plasticity model calibration: Phase i ($i = 1,..,4$)
CP Polygranular model Phase i ($i = 1,..,4$)
Stress-strain curves
Vol. fraction exp. curve data
$a_j, b_j, n_j,$ $j = 1, \ldots, 5$
SV Macroscopic model
State Variable model calibration: Triaxial loading
Olson-Cohen Model [Eq.4)]

Figure 1-2: Process flow of the integrated multi-scale calibration of Crystal Plasticity and State Variable models

LS-OPT® is a standalone simulation-based optimization algorithm which can be used to represent and optimize any general process with an arbitrary number of steps that may be sequential, parallel or nested. It is, therefore, well suited for ICME. LS-OPT® can be interfaced with several commercial Finite Element tools and pre- and post-processors, but has a dedicated interface to LS-DYNA®. This interface allows extraction of most results available in the LS-DYNA output database, as well as the substitution of parameter values in the LS-DYNA input files. LS-OPT includes capabilities for reliability-based analysis and optimization as well as special features for material calibration (mainly curve matching).

The LS-OPT part of the manual is an extract of the main LS-OPT User's Manual [1], but focuses mainly on the features required to execute the goals of the project. The main LS-OPT User's Manual [1] is frequently referenced in this ICME manual. The main new features introduced under the ICME program are:

- The construction of a multi-level capability for conducting an integrated calibration and product design optimization. This is required to drive the combination of multiple stages and levels as shown in Figure 1-1 and Figure 1-2. As can be seen, the multi-stage processes consist of (i) Material identification → vehicle optimization, (ii) CP calibration → Polycrystal analysis → SV calibration → Forming analysis and (iii) CP Yield → CP Hardening parameter calibration. These are executed at different levels using a multi-level structure.
- The ability to transfer variables between different levels (transfer variables) and different stages (response variables) of this multi-level structure. This feature includes the ability to evaluate expressions within the multi-stage analysis chain. Response variables are required so that scalar or vector responses generated in a parent stage can be substituted in the solver input files of a child stage.
- Graphical tools to navigate a complex ICME structure.

- o Interfaces for extracting optimization results (LS-OPT stage type).
- o Additional interfaces for running a third party FE analysis program. This was used at the beginning of the project, before the availability of the CP and SV models in LS-DYNA.

Since LS-DYNA is the main Finite Element analysis program employed in this project, the two material models central to the project, namely State Variable and Crystal Plasticity, were implemented as user material models in the program. The original source code developed by Pacific Northwest National Laboratory and Michigan State University/Ohio State University respectively was used for this purpose. The CP and SV material components form part of a multi-scale material model coordinated by LS-OPT.

The Crystal Plasticity model is used to model single crystal micro-pillars. The force-displacement curve obtained from a micro-pillar analysis and test data is used to calibrate the CP parameters. To bridge the two material scales, a polycrystal analysis, in which the crystals have random orientations, is then assembled using these parameters. The stress-strain output of the polycrystal analysis provides the input to the State Variable model which is used for the Finite Element analysis of structures in LS-DYNA. The examples in this manual include a sheet-metal forming problem (T-component) as well as a multi-disciplinary analysis (vehicle side panel). The CP model only applies to explicit dynamic analysis whereas the SV model also applies to implicit analysis as well as one-step analysis. The last two methods are used to accelerate the stamping analysis. See Figure 1-2.

All the steps starting with the CP calibration and ending with the output results of a vehicle analysis can be conducted in a single execution of the LS-OPT program. During this process the progress of each level can be monitored.

Examples are included to demonstrate the features and capabilities required to set up an ICME model. The first example is a simple two-level system whereas the second, main example uses the FE models generated and provided by the participants of the project, namely micro-pillars, the T-component stamping model as well as all the vehicle models.

The multi-level structure can also be applied to direct reliability-based optimization as well as tolerance optimization [5].

1.2. How to read this manual

This manual focuses on the main features (material models, multi-level optimization and parameter identification) of LS-OPT required to set up the ICME model. Therefore, not all the capabilities are discussed in depth, and hence *this manual should be read in conjunction with the LS-DYNA and LS-OPT User's Manuals* for Version 5.2 or beyond [1]. The full LS-OPT User's manual can be downloaded at http://ftp.lstc.com/user/ls-opt/5.2.1/lsopt_52_manual.pdf . The LS-DYNA User's Manual [6] can be downloaded at http://ftp.lstc.com/user/manuals . Download information can be obtained from support@lstc.com .

This manual functions as a hypertext document such that links in the manual body (to other sections as well as external reference material) can be used for cross-referencing (**Ctrl-click**) and will take the reader to the relevant item such as Section 9, Reference [5]. **Alt+Left Arrow** returns to the original reference point.

References to the *LS-OPT User's Manual* are shown in italics in the form e.g. *Chapter 3: Graphical User Interface* of Reference [1].

It is also recommended that this manual be read in conjunction with the final report: **Integrated Computational Materials Engineering Approach to Development of Lightweight 3GAHSS Vehicle Assembly (ICME 3GAHSS)**.

This manual is organized as follows. This chapter presents an overview of the ICME model. Figure 1-1 and Figure 1-2 present a summary of the integrated model, which consists of a process with several substeps, e.g. micro-level material calibration, macro-level calibration, component validation, design optimization etc. LS-OPT provides the ability to assemble and optimize a general process consisting of an arbitrary number of substeps of different types. Therefore, it is perfectly suited to represent the ICME model. Chapters 2 to 6, provide a general introduction to the software and its usage. Chapter 2 provides an overview the LS-OPT software to introduce a beginner user to the information about installing the software, setting up a process flow using it and running the software. The component entities of LS-OPT used in setting up a process are explained in Chapters 3 to 5. These components are used to represent the various steps in the ICME model, such as CP and SV model analysis, calibration, vehicle analysis, the design variables, responses etc. Chapter 6 presents the information about running and monitoring a process using LS-OPT, which in the context of ICME can calibrate the multiscale material models, and perform integrated design and material optimization. Chapter 7 presents two specific applications of LS-OPT pertinent to ICME – material parameter identication and multilevel optimization. Chapter 8 presents the two material models with different length scales (CP and SV), followed by multilevel optimization examples in Chapter 9. Finally, the detailed steps of the ICME setup, which is a multilevel process, are presented in Chapter 10 using an example.

1.3. REFERENCES

[1] Stander, N., Roux, W.J., Basudhar, A., Eggleston, T., Craig, K.-J. *LS-OPT Version 5.2 User’s Manual*, December 2015. http://ftp.lstc.com/user/ls-opt/5.2.1/lsopt_52_manual.pdf

[2] Horstemeyer, M., Multiscale Modeling: A Review, *Practical Aspects of Computational Chemistry, ed. J. Leszczynski and M.K. Shukla, Springer Science & Business Media*, pp. 87-135, 2009

[3] Zamiri, A. R. and Pourboghrat, F., “A novel yield function for single crystals based on combined constraints optimization”, *International Journal of Plasticity*, 26:731–746, 2010.

[4] Olson, G. B. and Cohen, M., “Kinetics of strain-induced martensitic nucleation”, *Metallurgical Transactions A*, 6A:791–795, 1975.

[5] Basudhar, A, Stander, N, Gandikota, I, Svedin, Å, Witowski, K: “Design Tolerance Optimization using LS-OPT”, 13th LS-DYNA Forum, Bamberg, Germany, 2014

[6] Hallquist, J.O. *LS-DYNA User's Manual, R8.0, 2015.*

2. Getting Started

This chapter presents a basic introduction to the LS-OPT software – installation of the software, and setting up and running a simple optimization using it. The ICME model setup consists of a more advanced multilevel process (Section 7.2) starting from the basic entities and features introduced in this chapter that are further explained in Chapters 3 to 6.

2.1. Installation of LS-OPT

The user is advised to consult Reference [1] for detailed information on the installation of LS-OPT.

Table 2-1 describes the LS-OPT execution commands.

Table 2-1: LS-OPT execution commands

Command	Description
lsoptui *command_file_name*	Execute the graphical user interface
lsopt *command_file_name*	LS-OPT batch execution
viewer *command_file_name*	Execute the graphical postprocessor (also accessible from main GUI)

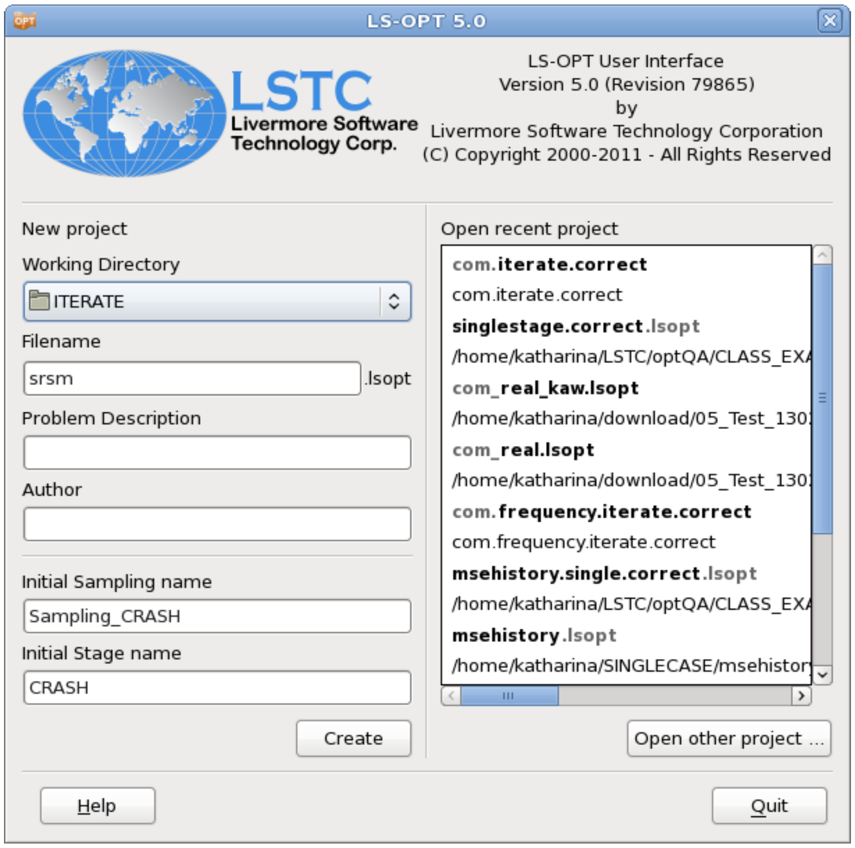
2.2. Setup of a simple optimization problem

2.2.1. Working directory

Create a working directory for keeping the main command file, input files and other command files as well as the LS-OPT program output. Make sure there are no blanks in the path names.

2.2.2. Startup

Open the graphical user interface of LS-OPT as described in *Chapter 3: Graphical User Interface* of Reference [1] and enter the required specifications to generate an LS-OPT project file to start from, Figure 2-1. Selecting *Create* will open up the main LS-OPT GUI window, Figure 2-2.



LS-OPT 5.0

LS-OPT User Interface
Version 5.0 (Revision 79865)
by
Livermore Software Technology Corporation
(C) Copyright 2000-2011 - All Rights Reserved

New project

Working Directory: ITERATE

Filename: srsm .lsopt

Problem Description:

Author:

Initial Sampling name: Sampling_CRASH

Initial Stage name: CRASH

Create

Open recent project

com.iterate.correct
com.iterate.correct
singlestage.correct.lsopt
/home/katharina/LSTC/optQA/CLASS_EXA
com_real_kaw.lsopt
/home/katharina/download/05_Test_130
com_real.lsopt
/home/katharina/download/05_Test_130
com.frequency.iterate.correct
com.frequency.iterate.correct
msehistory.single.correct.lsopt
/home/katharina/LSTC/optQA/CLASS_EXA
msehistory.lsopt
/home/katharina/SINGLECASE/msehistory

Open other project ...

Help

Quit

Figure 2-1: LS-OPT Startup dialog. Select the working directory, enter a name for the LS-OPT project file and a name for the initial sampling and initial stage to generate a new project.

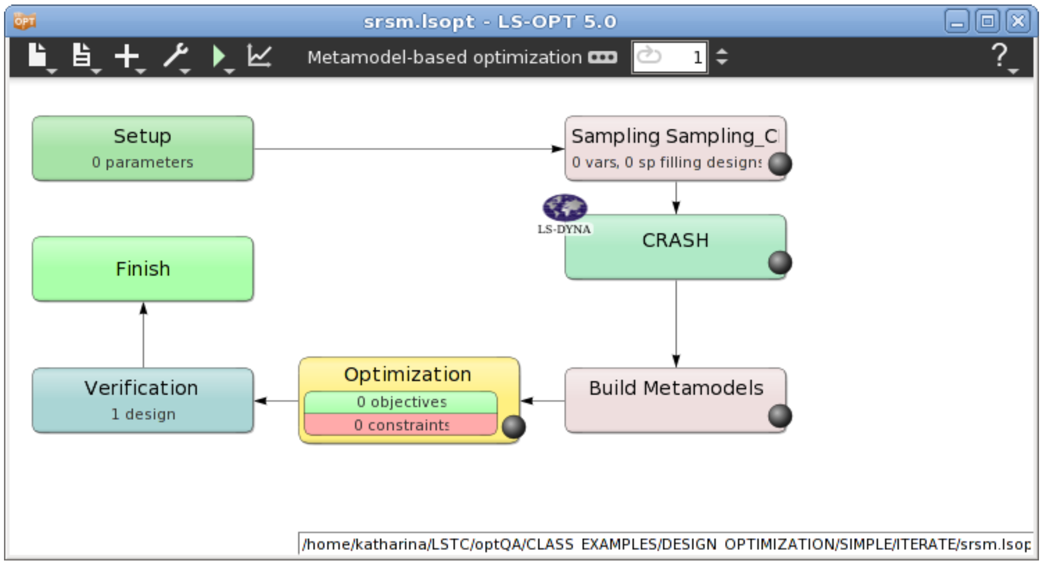
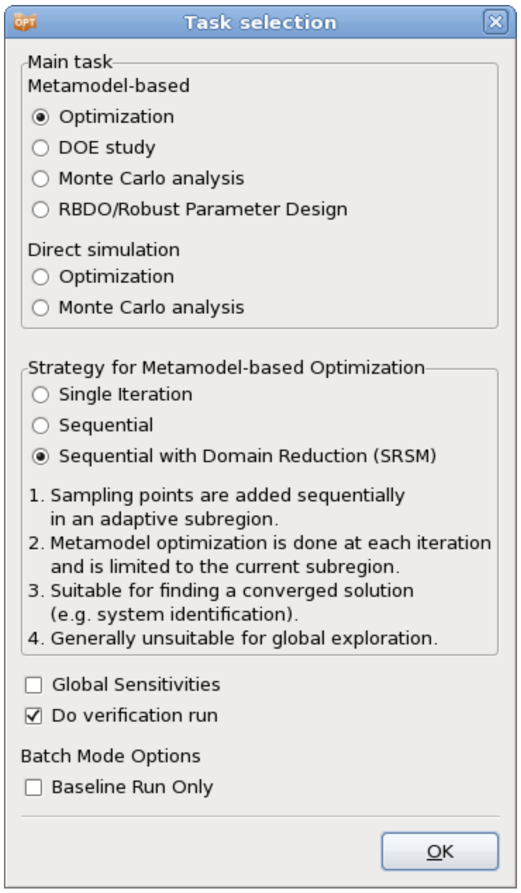


Figure 2-2: The main LS-OPT GUI window visualizes the optimization process flow. Selecting a box opens the respective dialog. The stage box (CRASH) can be moved freely using the left mouse button.

2.2.3. Task

Open the **Task** dialog by selecting the corresponding icon from the control bar (▪▪▪). Select the task to run, Figure 2-3, e.g. *Metamodel-based Optimization* with Strategy: *Sequential with Domain Reduction, (Chapter 4: Task Dialog – Selecting a Task and Strategy [1])*. The main GUI displays the process flow of the selected task.



Main task

Metamodel-based

- [✓ Optimization]
- DOE study
- Monte Carlo analysis
- RBDO/Robust Parameter Design

Direct simulation

- Optimization
- Monte Carlo analysis

Strategy for Metamodel-based Optimization

- Single Iteration
- Sequential
- [✓ Sequential with Domain Reduction (SRSM)]

1. Sampling points are added sequentially in an adaptive subregion.
2. Metamodel optimization is done at each iteration and is limited to the current subregion.
3. Suitable for finding a converged solution (e.g. system identification).
4. Generally unsuitable for global exploration.

- Global Sensitivities
- [✓ Do verification run]

Batch Mode Options

- Baseline Run Only

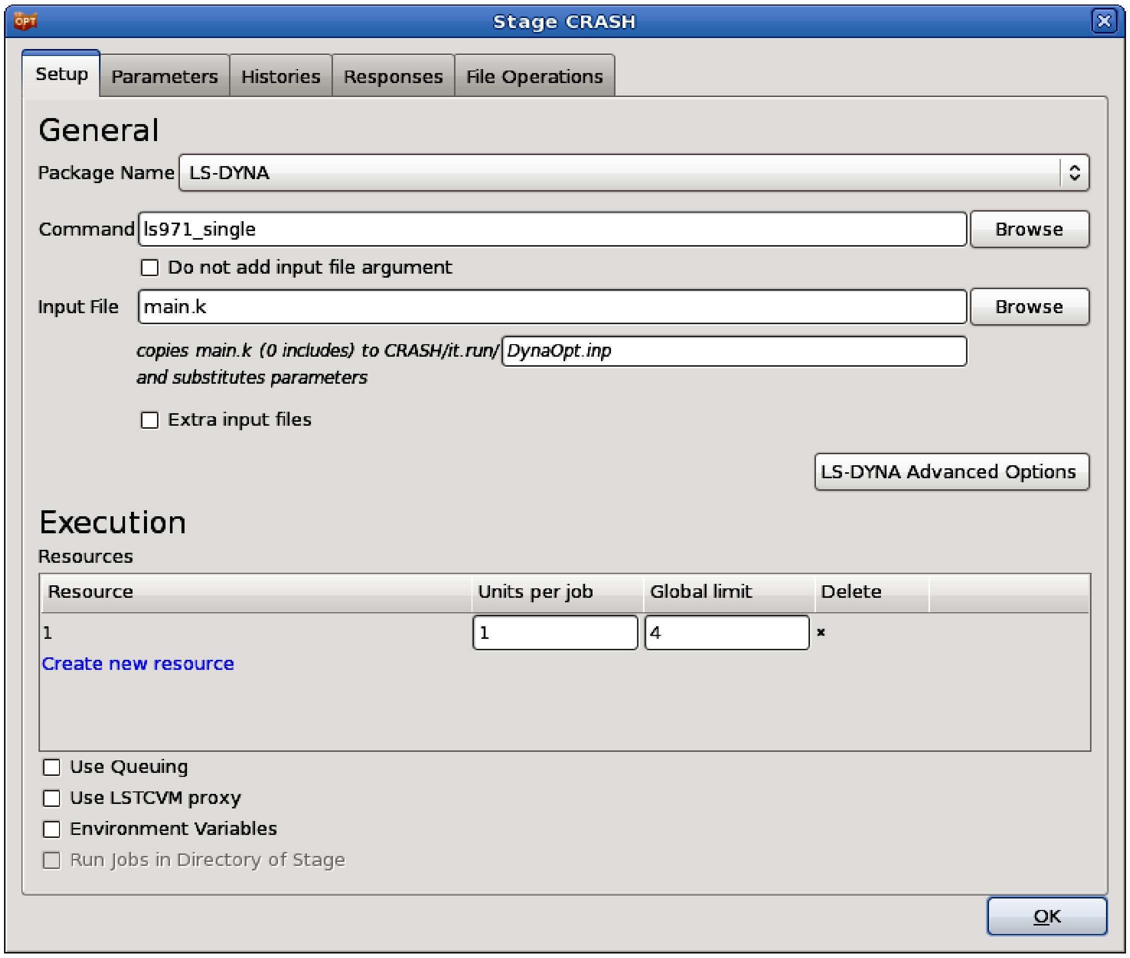
Figure 2-3: Task dialog. Select the main task and strategy

2.2.4. Stage

Set up the process chain. In the simplest case, a single **Stage** is required to interface with a solver, e.g. LS-DYNA. Select the already available **Stage** box, Figure 2-4. Select the solver *Package Name*, the solver *Command* and the parameterized *Input File*, Chapter 3. In more complex cases further stages can be added, e.g. for a pre-processor or post-processor.

Then switch to the **Parameters** tab to check the parameters found in the solver input file, Figure 2-5.

Next, switch to the **Responses** and **Histories** panel, Figure 2-6, to define results to be extracted from the solver output database (e.g. to be used as objectives or constraints in the optimization phase), Chapter 5.



Stage CRASH

Setup | Parameters | Histories | Responses | File Operations

General

Package Name: LS-DYNA

Command: ls971_single

☐ Do not add input file argument

Input File: main.k

copies main.k (0 includes) to CRASH/it.run/ *DynaOpt.inp*
and substitutes parameters

☐ Extra input files

LS-DYNA Advanced Options

Execution

Resources

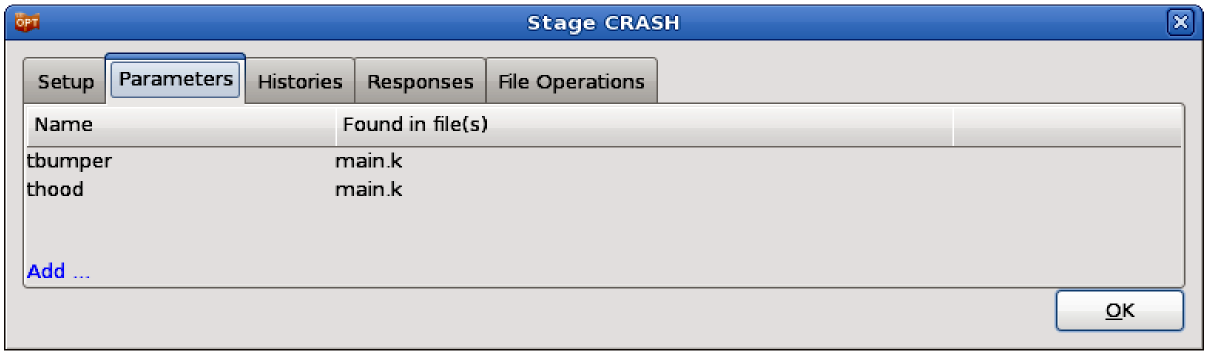
Resource	Units per job	Global limit	Delete
1	1	4	×

Create new resource

☐ Use Queuing
☐ Use LSTCVM proxy
☐ Environment Variables
☐ Run Jobs in Directory of Stage

OK

Figure 2-4: Stage dialog - Setup. Select the solver package name, the command and the solver input file



Stage CRASH

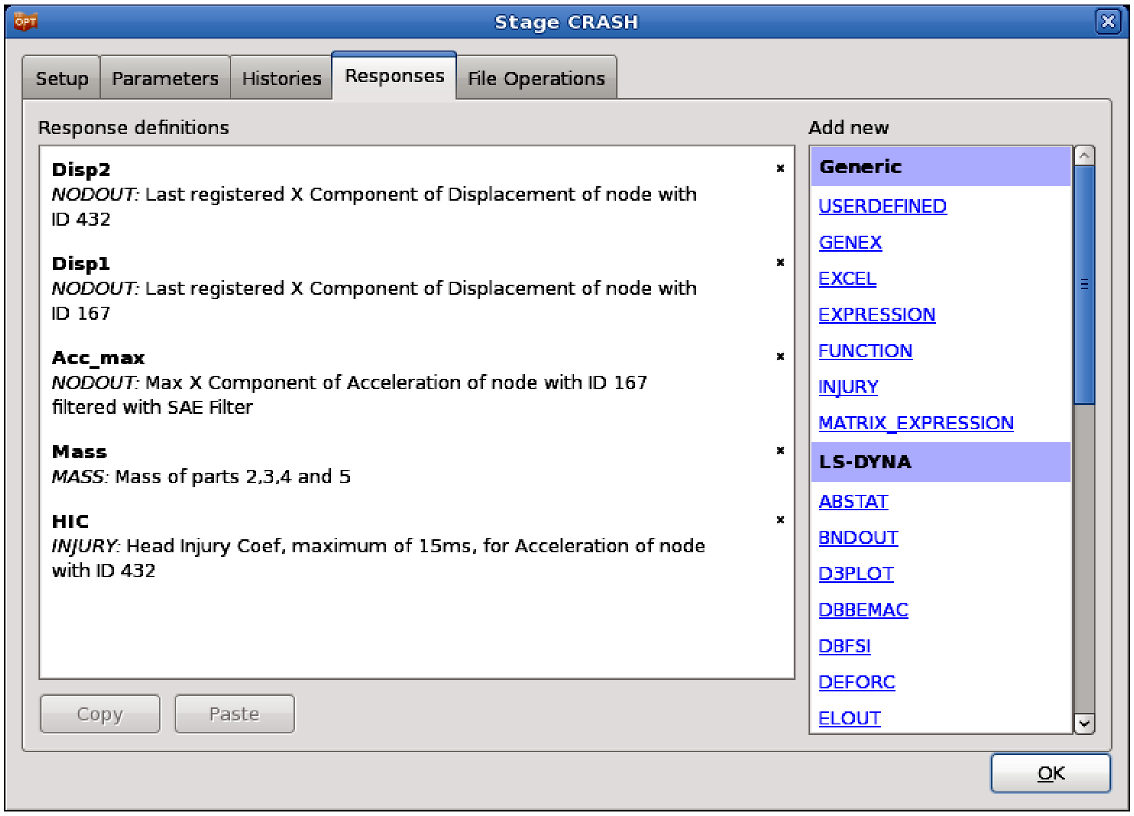
Setup | Parameters | Histories | Responses | File Operations

Name	Found in file(s)
tbumper	main.k
thood	main.k

Add ...

OK

Figure 2-5: Stage dialog – Parameters. Displays the parameters found in the input file specified in Setup



Disp2
NODOUT: Last registered X Component of Displacement of node with ID 432

Disp1
NODOUT: Last registered X Component of Displacement of node with ID 167

Acc_max
NODOUT: Max X Component of Acceleration of node with ID 167 filtered with SAE Filter

Mass
MASS: Mass of parts 2,3,4 and 5

HIC
INJURY: Head Injury Coef, maximum of 15ms, for Acceleration of node with ID 432

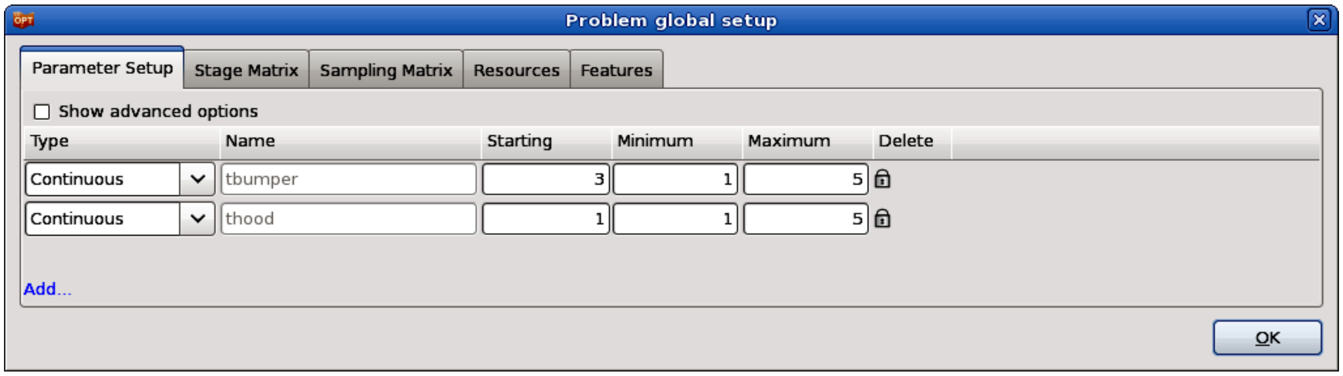
Figure 2-6: Stage dialog - Responses page. Select a response type from the list on the right to add a new response definition.

2.2.5. Setup

Select the **Setup** box at the top left of the main GUI, Chapter 4. All parameters that are defined in stage input files should automatically be available as constants, Figure 2-7. Select the desired variable **Types**. In most cases *Continuous* variables are used.

Then enter the requested values, e.g. the *Starting* value and *Minimum* and *Maximum* values to define the design space for a continuous variable.

Now follow the arrows to the next box in the process flow to define the respective settings.



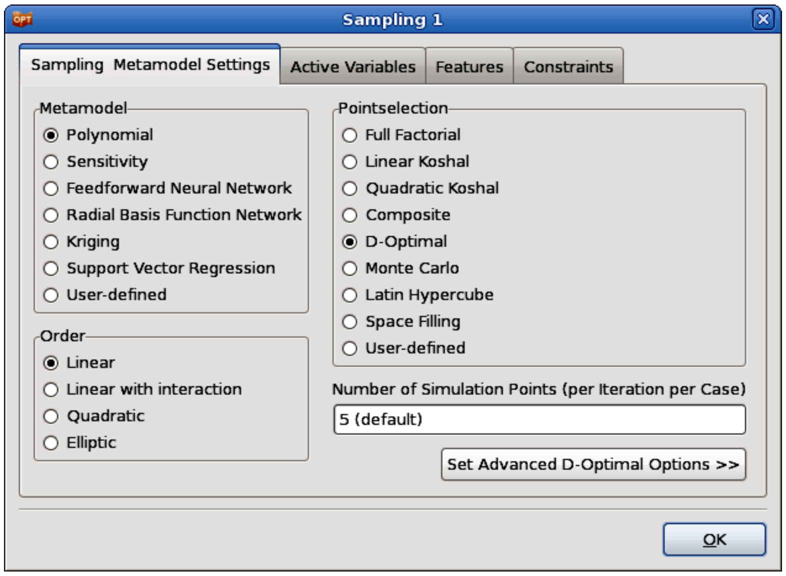
Type	Name	Starting	Minimum	Maximum	Delete
Continuous	tbumper	3	1	5	
Continuous	thood	1	1	5	

Figure 2-7: Parameter Setup dialog. Define the parameter type and required values.

2.2.6. Sampling and Metamodels

Select the **Sampling** box, *(Chapter 9: Sampling and Metamodel Dialog [1])*. Select the *Metamodel* and *Point Selection* types, or just use the default values, Figure 2-8.

The **Build Metamodels** box is coupled to the same dialog as the **Sampling** box. It is displayed at the end of the process to correctly represent the optimization process. Hence the Build Metamodels box can be skipped.



Sampling 1

Sampling Metamodel Settings | Active Variables | Features | Constraints

Metamodel
- [✓ Polynomial]
- Sensitivity
- Feedforward Neural Network
- Radial Basis Function Network
- Kriging
- Support Vector Regression
- User-defined

Order
- [✓ Linear]
- Linear with interaction
- Quadratic
- Elliptic

Pointselection
- Full Factorial
- Linear Koshal
- Quadratic Koshal
- Composite
- [✓ D-Optimal]
- Monte Carlo
- Latin Hypercube
- Space Filling
- User-defined

Number of Simulation Points (per Iteration per Case): 5 (default)

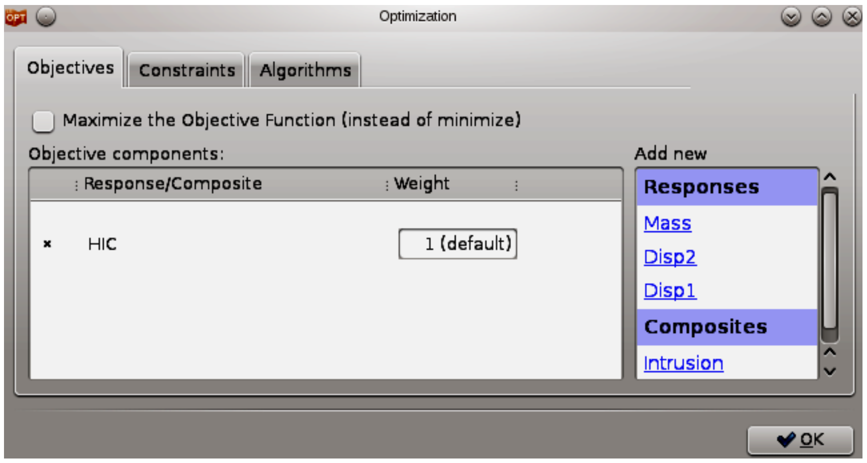
Set Advanced D-Optimal Options >>

OK

Figure 2-8: Sampling dialog. Select the metamodel type and point selection scheme.

2.2.7. Optimization

Select the **Optimization** box, *(Chapter 11 – Optimization Dialog – Objectives, Constraints and Algorithms [1])*. From the previously defined *Responses*, select the objectives, Figure 2-9.



Optimization

Objectives | Constraints | Algorithms

Maximize the Objective Function (instead of minimize)

Objective components:

Response/Composite	Weight
HIC	1 (default)

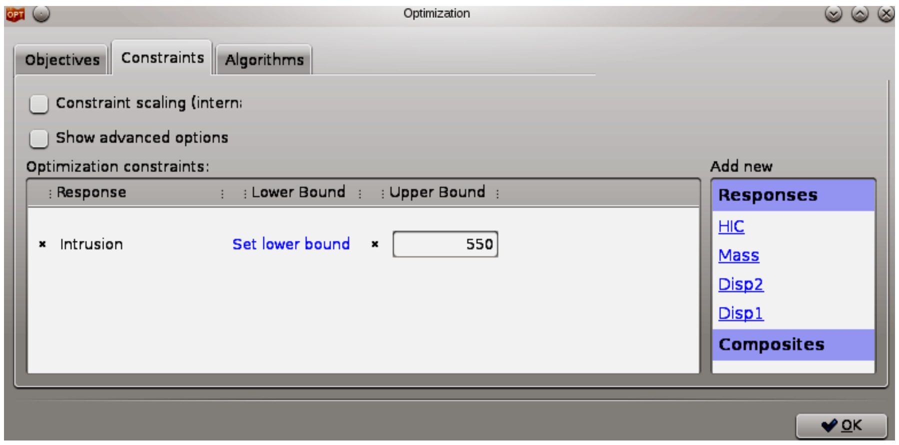
Add new

Responses
- Mass
- Disp2
- Disp1

Composites
- Intrusion

OK

Figure 2-9: Optimization - Objectives. Select the objective components from the list on the right.



Constraint scaling (interna

Show advanced options

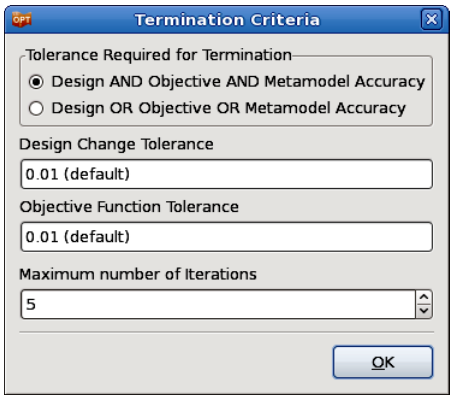
Optimization constraints:

Response	Lower Bound	Upper Bound
Intrusion	Set lower bound	550

Figure 2-10: Optimization - Constraints. Select constraints from the list on the right. Specify lower and upper bounds as required.

2.2.8. Termination criteria

Select the Termination criteria box, *(Chapter 12 – Termination Criteria [1])*. Specify the *Maximum number of Iterations*, e.g. 5 iterations. Use the default values for the other options.



Tolerance Required for Termination
[✓ Design AND Objective AND Metamodel Accuracy]
Design OR Objective OR Metamodel Accuracy

Design Change Tolerance: 0.01 (default)

Objective Function Tolerance: 0.01 (default)

Maximum number of Iterations: 5

Figure 2-11: Termination Criteria dialog. Specify the maximum number of iterations

2.2.9. Run

After setting up the optimization problem, run the task using the options from the control bar **Run** menu ( ), *(Section 3.3 – Graphical User Interface [1])*.

It is recommended to first run a *Baseline Run* to check if the stage process chain works correctly and the results are extracted as expected. Then run the full task using the *Normal Run* option.

2.2.10. Viewer

Use the *Viewer* [1] to evaluate the results by selecting from the the main GUI window control bar. The Viewer provides features to display metamodels and plot simulation results and optimization progress.

3. Stage Dialog − Defining the Solver

This chapter describes how to interface LS-OPT with simulation packages, parametric preprocessors or postprocessors. Standard interfaces as well as interfaces for user-defined executables are discussed.

The main entity discussed here is the *Stage* dialog which allows the user to define a step in the simulation process. Graphical tools are provided to define a process by inserting multiple stages.

3.1. Introduction

Since an executable program is considered to be a key part of the stage definition it is often simply referred to as the ***solver***. Therefore, in addition to its normal meaning as a program to, for instance, solve a physics problem, it can also refer to a pre- or postprocessor or any other executable program or script that is essential to the execution or management of a step within a simulation process. Several types of stages are used in the context of ICME (Figure 1-1 and Figure 1-2), such as LS-DYNA for solving the physics of vehicle impact, forming, micropillar compression etc., LS-OPT stage for nested optimization (e.g. SV and CP material calibration), ANSA for shape parametrization, and user-defined stage (e.g. analytical solver for SV Olson-Cohen model).

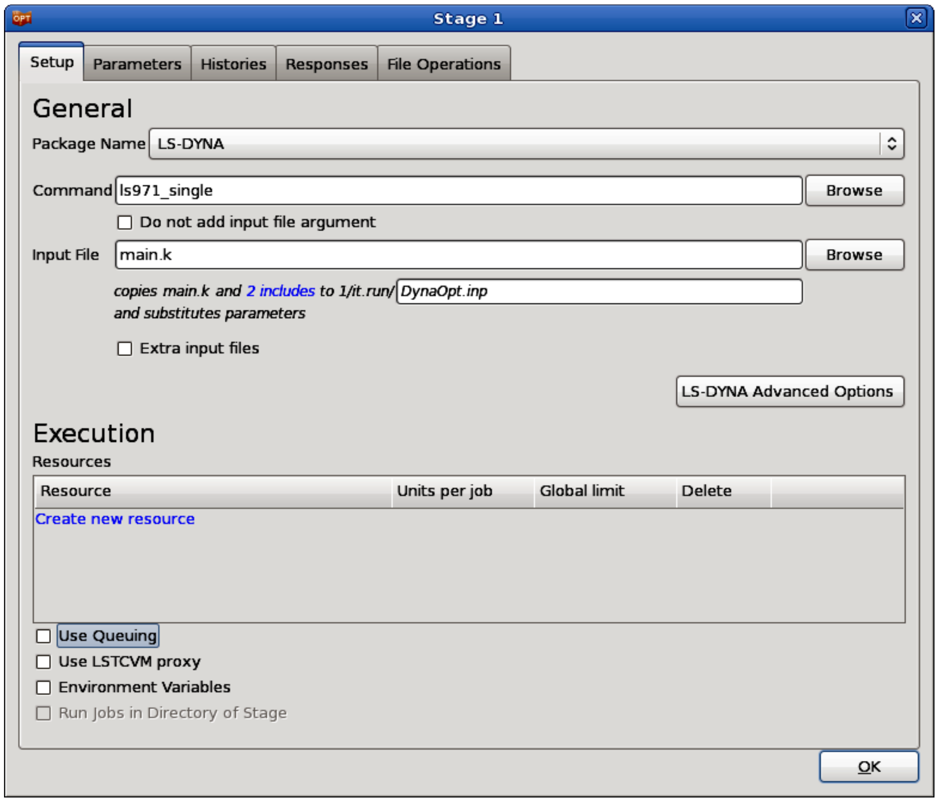
A stage not only executes the solver command to run a simulation, but it also extracts the necessary responses and histories associated with that simulation, so that these may be used in an optimization. For a simple optimization, a response is usually a physical attribute of the design, such as mass, displacement, injury criterion etc. In the context of ICME, a response can also be an optimized quantity, such as a calibrated material parameter. Different types of responses and histories are explained in further detail in Chapter 5. This chapter focuses mainly on the input parametrization and execution of different types of solvers.

3.2. General Setup

Figure 3-1 shows the general setup dialog for a Stage in the process. The options are described in Table 3-1.

Table 3-1: Stage dialog Setup options: General options

Option	Description	Reference
Package Name	The following software package identifiers are available:	
	LS-DYNA	Section 3.3.1
	LS-OPT	Section 3.3.3
	LS-PREPOST	Section 3.3.2
	ANSA	*Section 5.3.6 [1]*
	Excel	Section *5.3.10 [1]*
	Matlab	Section *5.3.12 [1]*
	META Post	*Section 5.3.8 [1]*
	User-Defined	Section 3.3.4
	User-Defined Postprocessor	Section 3.3.6
Command	Command to execute the solver.	Section 3.2.1
Input File	Parameterized input file for the preprocessor or solver. The specification of an input file is not required for a user-defined solver. The parameterization of the input file is explained in Section 3.2.3.	Section 3.2.2
Extra input files	A list of extra input files can be provided. The files are copied to the run directories from any user-defined source directory. Parameter values are substituted by default, but parsing can be omitted. LS-DYNA Include files do not have to be specified as they are automatically and recursively searched by LS-OPT when given the name of the main input file. This feature is also supported for certain packages under the user-defined solver type (see 3.3.4).	Section 3.2.2



Stage 1

Setup | Parameters | Histories | Responses | File Operations

General

Package Name: LS-DYNA

Command: ls971_single [Browse]

☐ Do not add input file argument

Input File: main.k [Browse]

copies main.k and 2 includes to 1/it.run/ *DynaOpt.inp*
and substitutes parameters

☐ Extra input files

[LS-DYNA Advanced Options]

Execution

Resources

Resource	Units per job	Global limit	Delete
Create new resource			

☐ Use Queuing
☐ Use LSTCVM proxy
☐ Environment Variables
☐ Run Jobs in Directory of Stage

[OK]

Figure 3-1: Stage dialog Setup panel

3.2.1. Command

The command to execute the solver must be specified. The command depends on the solver type and can be an executable program or a script. Since a standard input deck name (also called the base file name) is automatically appended during run-time the solver input file name argument should be omitted by default. See respective package interface sections for details. In the case of the standard solvers, the appropriate syntax is automatically used (e.g. `i=DynaOpt.inp` for LS-DYNA). The execution command may include any number of additional arguments.

Remarks:

1. The command must be specified in one of the following formats:
   - *Browse*. If browsing the project directory or a directory relative to the project directory, LS-OPT automatically prepends the project directory environment ${LSPROJHOME} to the execution command.
   - *Absolute path*, e.g. "/origin/users/john/crash/runmpp"
   - If the executable is located in a directory which is in the execution path, the command can be specified using only the name of the respective executable, e.g. "ls971_single"
2. *Linux:* Do not specify the command `nohup` before the solver command and do not specify the UNIX background mode symbol &. These are automatically taken into account.

3. *Linux:* The command name must not be an alias.
4. *Windows:* A path to a program or file cannot contain any blanks or - (dash) symbols.

3.2.2. Input Files

LS-OPT handles two main types of solver input files, namely

1. the main input file and
2. extra input files.

LS-OPT converts the input template to an input deck for the preprocessor or solver by replacing the original parameter values (or labels) with new values determined by the sampling procedure. The specification of an input file is not required for a user-defined solver.

For LS-DYNA and most of the preprocessor interfaces, LS-OPT automatically searches for include files specified in the main input file, see Table 3-2. Include files can be specified recursively, i.e. there can be include file specifications in include files. The user-defined stage type also supports these features, but only for certain solver types (see 3.3.4).

Input files are copied to the run directories, parsed to substitute parameter values and renamed. Each stage type has its own standard input file name, e.g. for LS-DYNA, the file is renamed to `DynaOpt.inp`. For remote runs, input files are automatically transmitted to a computer cluster.

A record of the specified input files and parameters is displayed in the GUI but can also be checked in the `lsopt_input` file.

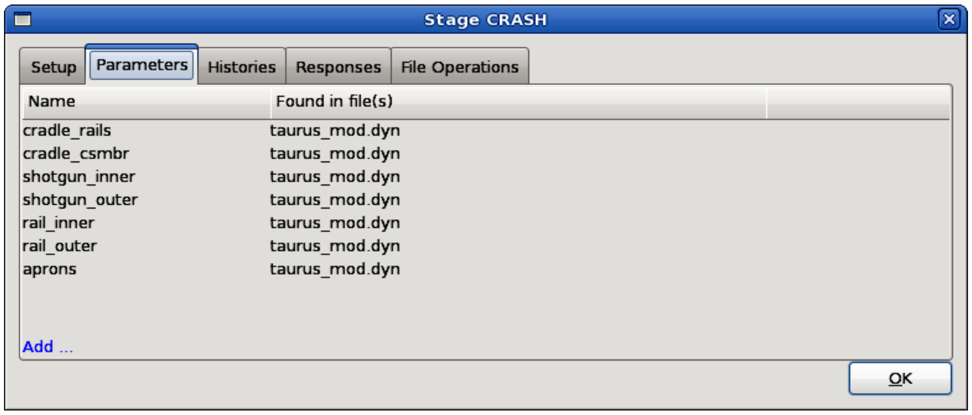
3.2.3. Parameterization of Input Files

For all stage types, input files can be parameterized using the User-defined parameter format, *(Section 5.2.5 – the User-defined parameter format [1])*. For the packages listed in Table 3-2, LS-OPT supports native parameters, see the respective package interface section for details. Native parameter types are also supported for certain solvers specified under user-defined solver types (see Section 3.3.4).

LS-OPTui will automatically recognize the native and User-defined parameters for the formats indicated in the table and list them on the **Parameters** panel, Figure 3-2. Parameters found in input files are also displayed as ‘Constants’ in the **Setup** dialog ‘Parameter Setup’ panel. The user can then change these constants to variables or dependents. The parameter names cannot be changed in the GUI so, if desired, must be changed in the original input file(s). A lock icon adjacent to the variable name indicates that the parameter names were imported from the input or include files.

Table 3-2: Parameters and include files

Package	Native parameters recognized in input file	User-defined Parameter Format recognized (see Section 3.2.3)	Include files recognized in input file	Reference
LS-DYNA®	Yes	Yes	Yes	Section 3.3.1
LS-PREPOST®	Yes	Yes	Yes	Section 3.3.2
ANSA[1]	Yes	Yes	Yes	*Section 5.3.6 [1]*
Matlab	Yes	Yes	No	Section *5.3.12 [1]*
LS-OPT	Yes	No	No	Section 3.3.3
Excel	N/A	No	No	Section *5.3.10 [1]*
User-defined	N/A	Yes	No	Section 3.3.4



Name	Found in file(s)
cradle_rails	taurus_mod.dyn
cradle_csmbr	taurus_mod.dyn
shotgun_inner	taurus_mod.dyn
shotgun_outer	taurus_mod.dyn
rail_inner	taurus_mod.dyn
rail_outer	taurus_mod.dyn
aprons	taurus_mod.dyn

Figure 3-2: Parameter panel: list of parameters found in stage input files

The 'include' files are also scanned recursively wherever this feature is available, making it nonessential to define extra files. Include files which are specified with a path, e.g. "`../../car5.k`" or "`/home/jim/ex4a/car6.k`" are not copied to the run directories and no parameter substitutions will be made in these files. This is solely to prevent unnecessary file proliferation. The user must however ensure that files, which are to be distributed to remote nodes through a queuing system *(see Appendix H.3 [1])*, do not contain any path specifications.

[1] BETA CAE Systems S.A.

These files are automatically transmitted to the relevant nodes where the solver will be executed. See also Section 3.3.1. If parameters are specified in include files with path specifications, these files should be specified as extra files if the user wants them to be parsed and copied to the run directories, Section 3.2.2.

Apart from the stage specific parameter formats in Section 3.2.3, a User-defined parameter format is recognized in all types of input files (*see Section 5.2.5 [1]*).

3.3. Package Interfaces

3.3.1. LS-DYNA

The CP and SV material models are implemented as user materials in LS-DYNA (Chapter 8) and it is, therefore, used as the solver to conduct finite element analyses using these models.

The file `DynaOpt.inp` is created from the parameterized LS-DYNA input template file. LS-OPT creates this file for each simulation, replacing the parameter values with appropriate values of that particular sample. By default, LS-OPT appends `i=DynaOpt.inp` to the solver command. Parameterization of the input file can be done using the User-defined parameter format or the *PARAMETER keyword. Include files in input files are recognized and parsed, see below for further information.

The LS-DYNA restart command will use the same command line arguments as the starting command line, replacing the `i=`*`input file`* with `r=runrsf`.

The *PARAMETER format

This is the recommended format. The parameters specified under the LS-DYNA `*PARAMETER` keyword are recognized by LS-OPT and will be substituted with a new value for each of the multiple runs. These parameters should automatically appear in the Parameter list of the GUI upon specification of the solver input file name. LS-OPT recognizes the “**`i`**”, “**`r`**” and “**`c`**” formats for integers, real numbers and strings respectively and will replace the number or string in the appropriate format. Note that LS-OPT will ignore the `*PARAMETER_EXPRESSION` keyword so it may be used to change internal LS-DYNA parameters without interference by LS-OPT.

For details of the `*PARAMETER` format please refer to LS-DYNA User’s Manual.

LS-DYNA include files

The handling (parsing, copying and transmitting) of include files by LS-OPT is automated. The following rules apply:

1. Include files may also contain parameters and are also parsed and copied (or transmitted) if the include file is specified in the keyword file *without a path*, for example:

```
*INCLUDE

input.k
```

2. If a path is specified for an include file, e.g.

   ```
   *INCLUDE
   ```

   C:\path\myinputfiles\input.k

   the file will not be copied, parsed or transmitted.
3. If the main input file is placed in a subdirectory of the main working directory and is specified with a relative path, e.g. `myinputfiles/input.k`, the directory (in this case `myinputfiles`) becomes a file environment for any include files which may also be placed in this directory. Therefore all include files specified without a path will automatically be copied (or transmitted) from this sub-directory (`myinputfiles`) to the run directories.

LS-DYNA/MPP

The LS-DYNA MPP (Message Passing Parallel) version can be run using the LS-DYNA option in the ”Stage” dialog of LS-OPTui. The following run command is an example of how an MPP command can be specified:

```
mpirun -np 2 lsdynampp
```

where `lsdynampp` is the name of the MPP executable.

3.3.2. LS-PREPOST

The file `LsPrepostOpt.inp` is created from the LS-PREPOST input template file. LS-OPT automatically appends “`–nographics c=LsPrepostOpt.inp 2> /dev/null > /dev/null`” to the command.

LS-PREPOST input file example with include:

test01.cfile:

```
$# LS-PrePost command file created by LS-PREPOST 3.0(Beta) -
31Mar2010(17:08)
$# Created on Apr-06-2010 (13:42:14)
cemptymodel
openc command "para01.cfile"
genselect target node
occfilter clear
genselect clear
genselect target node
occfilter clear
genselect clear
meshing boxshell create 0.000000 0.000000 0.000000 &size &size &size
&num &num &num
ac
meshing boxshell accept 1 1 1 boxshell
genselect target node
occfilter clear
```

```
refcheck modelclean 9
ac
mesh
save keyword "lsppout"
exit
```

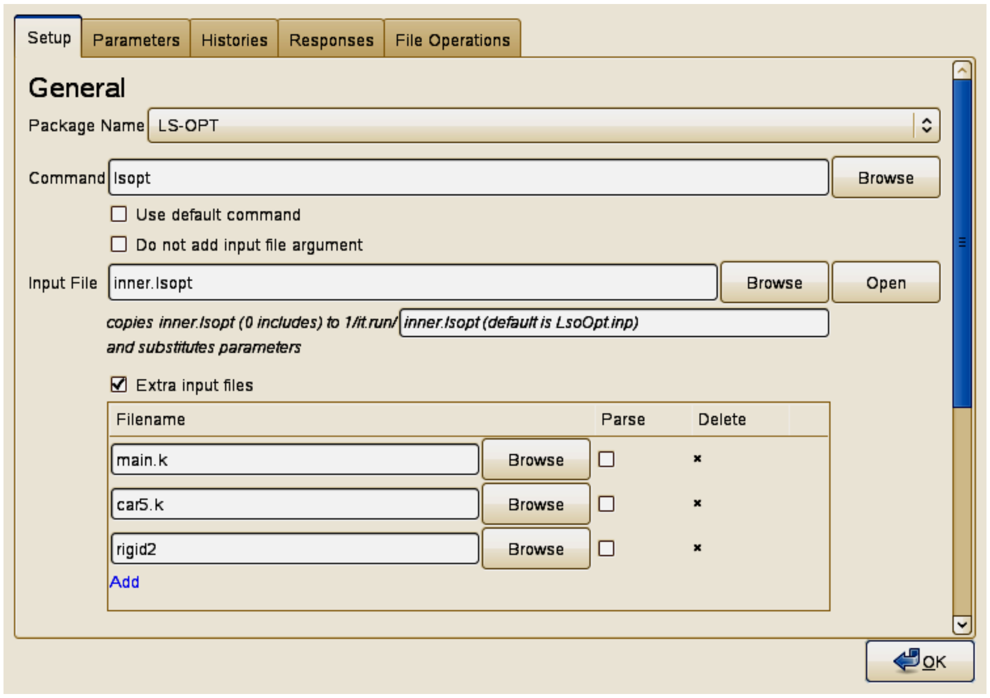
para01.cfile

```
parameter size 1.0
parameter num 2
```

3.3.3. LS-OPT

The LS-OPT stage allows one to extract optimized LS-OPT response values, which can then be used in another optimization with respect to a different set of variables. The LS-OPT stage can also be used to call a reliability task from an optimization task, e.g. tolerance optimization. It forms the basic foundation of multilevel optimization and plays an integral part in ICME, which is inherently a multilevel process (see Figure 1-1 and Chapter 10).

The LS-OPT stage simply executes another instance of the LS-OPT software in a nested optimization framework. Thus, it allows a user to set up a *Multilevel Optimization* problem, explained in Section **7.2**. The LS-OPT stage setup dialog is shown in Figure 3-3.



Package Name: LS-OPT

Command: lsopt

☐ Use default command

☐ Do not add input file argument

Input File: inner.lsopt

copies inner.lsopt (0 includes) to 1/it.run/ *inner.lsopt (default is LsoOpt.inp)*

and substitutes parameters

☑ Extra input files

Filename	Parse	Delete
main.k	☐	×
car5.k	☐	×
rigid2	☐	×

Figure 3-3: LS-OPT stage interface

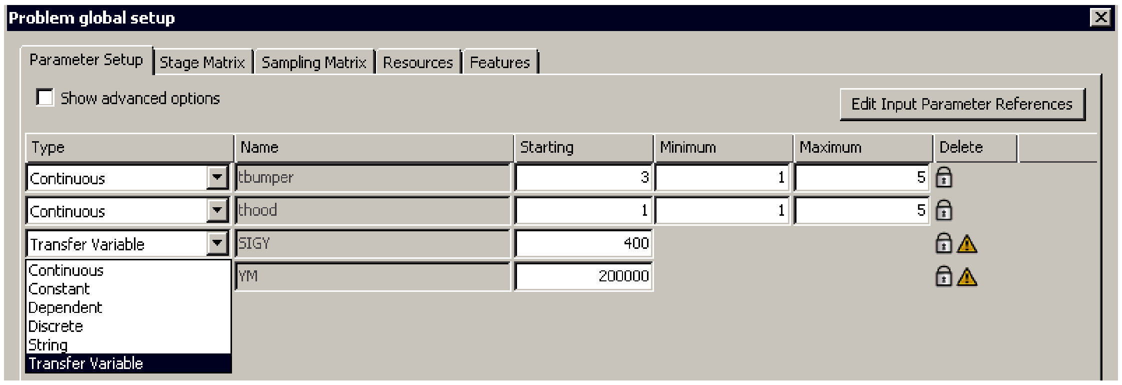
The fields that need to be specified for an LS-OPT stage are as follows.

1. Command: Like all other solver interfaces, the user needs to provide the command to run LS-OPT. There is a *Use default command* option that automatically fills in the path to the LS-OPT executable being used for the setup.

2. Input file: The input file for an LS-OPT stage is a .lsopt file itself that contains the setup for an inner level LS-OPT sub-problem. The file `LsoOpt.inp` (or a user specified name) is created from the LS-OPT input template file. By default, LS-OPT appends `LsoOpt.inp` to the solver command. Parameterization of the input file is done using *Transfer Variables* (Figure 3-4).
3. Extra Files: An important aspect to note in the LS-OPT stage setup is the use of extra input files with the *Parse* option unchecked (Figure 3-7). This is important because the input files of the lower level(s) need to be passed down from the upper level while not considering the lower level variables in the upper level. The details of the directory structure for multilevel problems are presented in Section **6.4**.

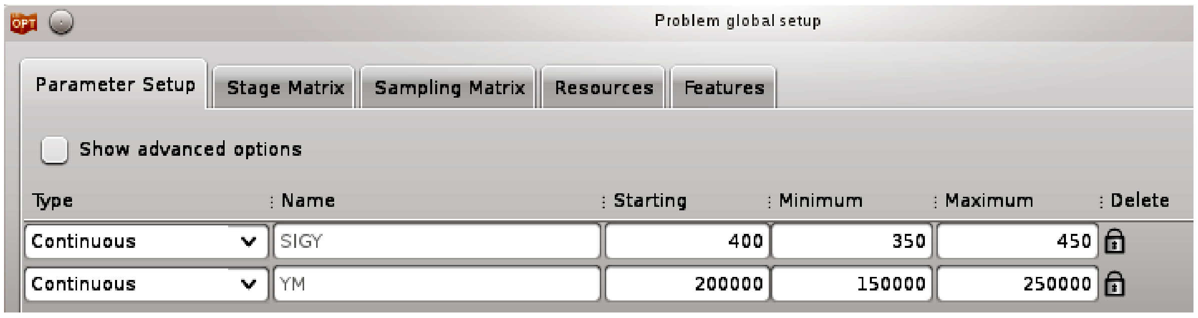
LS-OPT input file parameterization

The LS-OPT input file, i.e. the .lsopt file, is parameterized using *Transfer Variables*. The transfer variables are indicated using *type="iconstant" in the* LS-OPT stage input file. *Continuous* and *Discrete* variables can be set as a *Transfer Variables* using the LS-OPT GUI (Figure 3-4); these are then considered as constants at that level, but can be set as variables in preceding levels. These variables are automatically detected as constants by LS-OPT and populate the outer level *Global Setup* (for which the parameterized *.lsopt* file is a stage input file). The user can either use them as constants in the outer level or set them as variables.



Type	Name	Starting	Minimum	Maximum	Delete
Continuous	tbumper	3	1	5	🔒
Continuous	thood	1	1	5	🔒
Transfer Variable	SIGY	400			🔒⚠
	YM	200000			🔒⚠

Figure 3-4: Parameterization of inner level LS-OPT setup using Transfer Variables. The values of transfer variables are passed down from the upper level(s).



Type	Name	Starting	Minimum	Maximum	Delete
Continuous	SIGY	400	350	450	🔒
Continuous	YM	200000	150000	250000	🔒

Figure 3-5: Outer level global setup. `SIGY` and `YM` are automatically detected in the input file (i.e. inner level .lsopt file) and locked as they are Transfer Variables in the inner level.

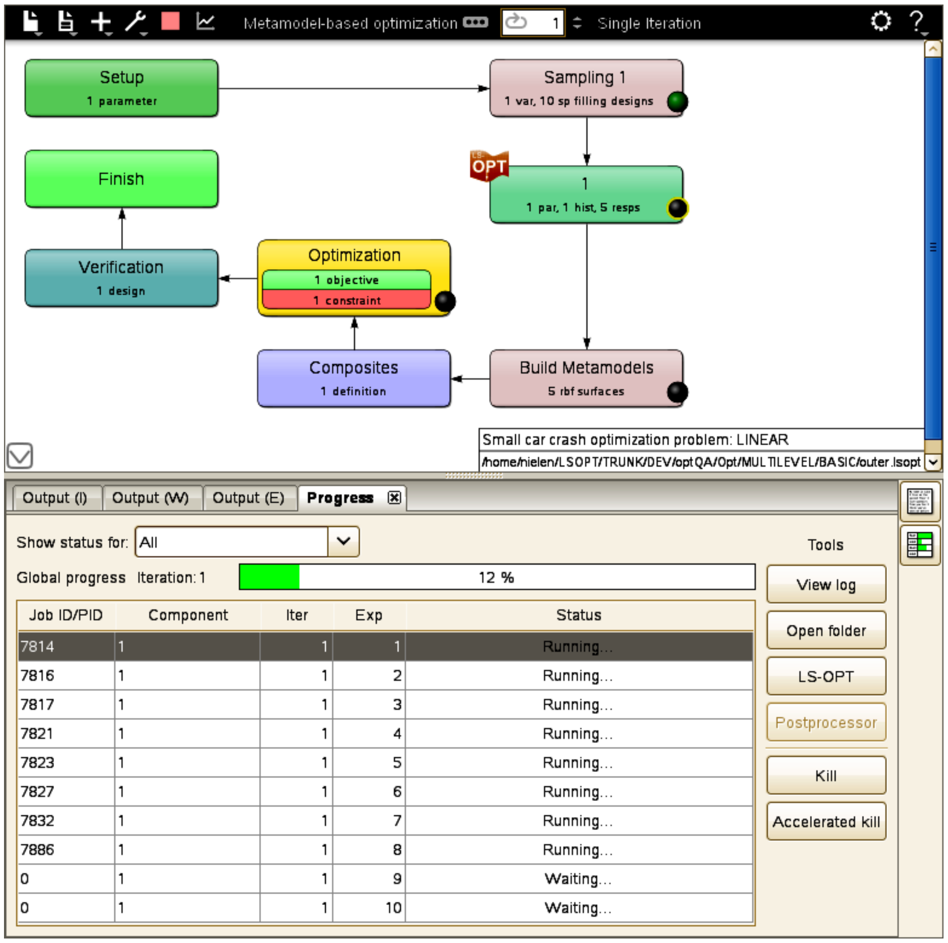
Remarks:

1. The user-defined parameter format << name>> is not allowed for the LS-OPT stage.
2. LS-OPT stage responses are extracted using the *LSOPT* response type (Section 5.3).

Navigating to view lower level setups and progress

Because of the complex recursive nature of a multilevel setup, simple navigation options are provided so that lower level setups can be inspected or edited recursively starting at the main (upper level) setup. During runtime, job progress can also be viewed recursively starting at the main progress window.

1. The *Open* button opposite the *Input file* text box allows the user to navigate down to the next level and will display the GUI for inner.lsopt, see Figure 3-3.
2. While a multilevel run is in progress, the user can also navigate to display the progress of a selected lower level job by clicking on the *LS-OPT* button in the progress dialog. Lower level job progress can also be monitored using the *View log* button to display the text output, see Figure 3-6.



Small car crash optimization problem: LINEAR

/home/nielen/LSOPT/TRUNK/DEV/optQA/Opt/MULTILEVEL/BASIC/outer.lsopt

Show status for: All

Global progress Iteration: 1 12 %

Job ID/PID	Component	Iter	Exp	Status
7814	1	1	1	Running...
7816	1	1	2	Running...
7817	1	1	3	Running...
7821	1	1	4	Running...
7823	1	1	5	Running...
7827	1	1	6	Running...
7832	1	1	7	Running...
7886	1	1	8	Running...
0	1	1	9	Waiting...
0	1	1	10	Waiting...

Figure 3-6: Progress window for the LS-OPT stage. Selecting the* LS-OPT *button for a selected job displays the LS-OPT GUI for it and allows the monitoring of a lower level.

3.3.4. User-defined program

A user-defined solver or preprocessor can be specified by selecting User-defined in LS-OPT*ui*. The `command` can either execute a command, or a script. The substituted input file `UserOpt.inp` will automatically be appended to the command or script. Variable substitution will be performed in the input file (which will be renamed `UserOpt.inp)`. The specification of an input file is optional. In its simplest form, the `prepro own` preprocessor can be used in combination with the design point file: `XPoint` to read the design variables from the run directory.

If the solver does not generate a 'Normal' termination command to standard output, the solver command must execute a script that has as its last statement the command: `echo 'N o r m a l'`.

3.3.5. Third Party solvers

LS-OPT supports certain popular Finite Element Analysis solvers under the *User-defined* solver type. For these solver types all the syntax rules (e.g. recursive include files, parameter keywords, etc.) associated with the input file are obeyed so that parameters can automatically be imported to the LS-OPT setup dialog.

LS-OPT recognizes the solver type by initially parsing the first line of the main input file. This line should be a comment line which contains the name of the package it represents.

Special response interfaces are not available, but response and history extraction are supported using

- GenEx (Chapter 7 [1])
- the user-defined post-processor (3.3.6)
- commercially available post-processors supported by LS-OPT (*see e.g. Section 5.3.8 – μETA (BETA CAE Systems SA)[1]*).

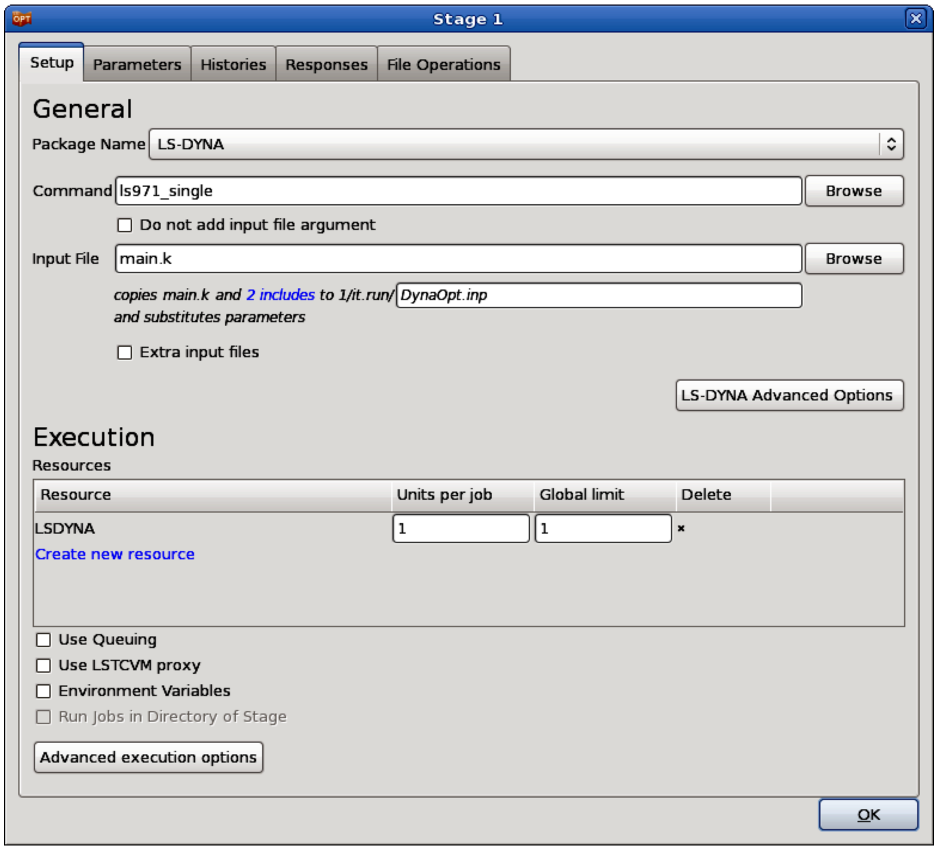
3.3.6. User-defined post-processor

The postprocessor allows extraction of data from any database it supports, so makes LS-OPT accessible to interface with any such supported solvers. This allows the postprocessor to read results from the solver database and place them in a simple text file or files for individual extraction of results.

In the case of user-defined post-processor, the full command needs to be provided, because LS-OPT does not internally construct the command using the input, database and result files. The output file needs to be written in the same format as for the μETA package (*see Section 5.3.14 [1]*).

Setting up an LS-OPT problem is similar to μETA stage, except that **User-defined Postprocessor** is selected as the package, and the session file and database path need not be provided as the related information is available in the command. It is also possible to run μETA as a user-defined post-processor (*see Section 5.3.14 [1]*). It is not necessary to provide the input and database files separately in this case. The output file name must however be specified.

3.4. Solver Execution


Stage 1

Setup | Parameters | Histories | Responses | File Operations

General

Package Name: LS-DYNA

Command: ls971_single

☐ Do not add input file argument

Input File: main.k

copies main.k and 2 includes to 1/it.run/ DynaOpt.inp *and substitutes parameters*

☐ Extra input files

Execution

Resources

Resource	Units per job	Global limit	Delete
LSDYNA	1	1	×
Create new resource			

☐ Use Queuing
☐ Use LSTCVM proxy
☐ Environment Variables
☐ Run Jobs in Directory of Stage

Figure 3-7: Stage dialog Setup panel

Table 3-3: Stage dialog Setup options: Execution options

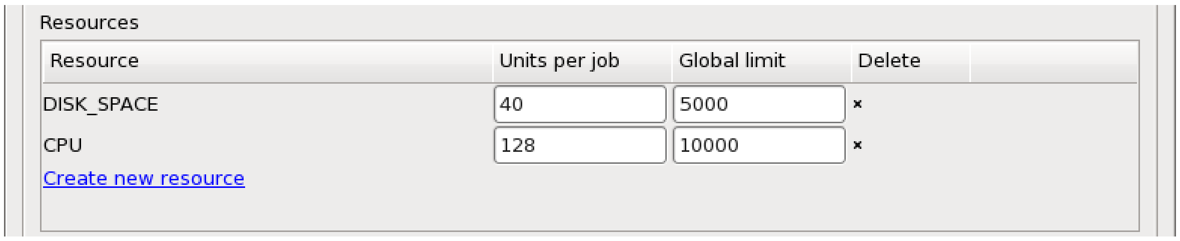
Option	Description	Reference
Resources	Settings for concurrent processing	Section 3.4.1
Use Queuing	Interfacing with load sharing facilities to enable running simulation jobs across a network.	Section 3.4.2
Environment Variables	Environment variables that will be set before executing a solver command.	*Section 5.4.4 [1]*
Run jobs in Directory of Stage	If multiple stages are defined, the command can be executed in the directory of another stage.	-
Recover Files	List of files to be recovered from remote machine, only available if a queuing system interface is used	*Section 5.4.5 [1]*

3.4.1. Specifying Computing Resources for Concurrent Processing

Multiple resource limits can be defined for each stage. The resource attributes consist of Units per job as well as the Global limit (see Figure 3-8). This feature is non-dimensional and therefore allows the user to specify limits on any type of computing resource such as number of processors, disk space, memory, available licenses, etc.

Example:

A user has 10,000 processors available and wants to execute an optimization run using MPP simulations requiring 128 CPUs per job. She therefore specifies the units per job as 128 and the global limit as 10,000. For this same optimization run, the user has 5,000Gb disk space available while using 40 Gb of disk space per job (which is deleted after the completion of each job). A second resource therefore has to be specified with attribute values 40 units per job and a global limit of 5,000. The resource setup is shown in Figure 3-8. The job scheduler will launch jobs that will not exceed any of these two limits.


Resources

Resource	Units per job	Global limit	Delete
DISK_SPACE	40	5000	×
CPU	128	10000	×

Create new resource

Figure 3-8: Definition of Resources for a Stage

Resources must be defined at the Stage level, but can be viewed in the Resource tab of the Setup dialog *(see Section 8.4 — Resources [1])*. The limits can be changed in either the Stage or Setup dialogs.

Stages can share resources. For instance, as part of an MDO problem, the same resource can be defined for multiple stages.

When using multiple computer clusters, independent resources are typically defined for each cluster. Jobs will then be run concurrently on all clusters within the limits defined for each cluster.

A single resource with a default of 1 Units per job and a Global limit of 1 is assumed for each stage at the beginning of the creation process. The default name is the solver type name. That also implies that if multiple stages use the same solver type, there will by default be only one resource definition. Resources can then be added or deleted as desired. To change a resource name, a new resource has to be added and the old resource deleted.

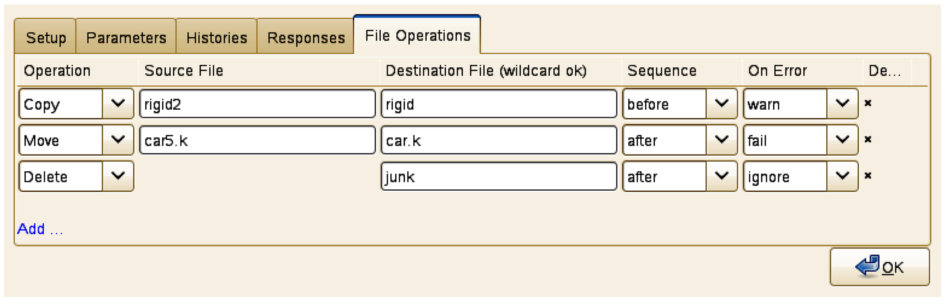
Remark

A resource definition related to e.g. the number of processors to be used for a simulation run does not replace the specification of the number of processors as a command line option or in the command script. The resource definitions are only used to calculate the number of jobs that are submitted concurrently.

3.4.2. Interfaces to Queuing Systems

The LS-OPT Queuing Interface interfaces with load sharing facilities (e.g. LSF[2] or LoadLeveler[3]) to enable running simulation jobs across a network. LS-OPT will automatically copy the simulation input files to each remote node, extract the results on the remote directory and transfer the extracted results to the local directory. The interface allows the progress of each simulation run to be monitored via LS-OPT*ui. See Appendix H.5 – Using an external queuing or job scheduling system [1]* for information on how to setup the interface. The supported queuing systems in LS-OPT are LSF, PBS, PBSPRO, SLURM, LoadLeveler, NQE, NQS, Black-box, SGE and User-defined.

3.5. File Operations



Operation	Source File	Destination File (wildcard ok)	Sequence	On Error	De...
Copy	rigid2	rigid	before	warn	
Move	car5.k	car.k	after	fail	
Delete		junk	after	ignore	

Figure 3-9: File Operations within a Stage run directory

LS-OPT allows file operations between Stages or within a Stage.

The requested Stage file operations are executed for all the run directories related to the Stage, e.g. `CRASH/1.1`, `CRASH/1.2`, etc. Within a Stage run directory, several file operations can be executed on files previously copied to the run directories or generated by the stage command before or after executing the stage command. See Figure 3-9 and *(Table 3-4: File transfer options between stages [1]).*

[2] Registered Trademark of Platform Computing Inc.
[3] Registered Trademark of International Business Machines Corporation

Table 3-4: File Operations

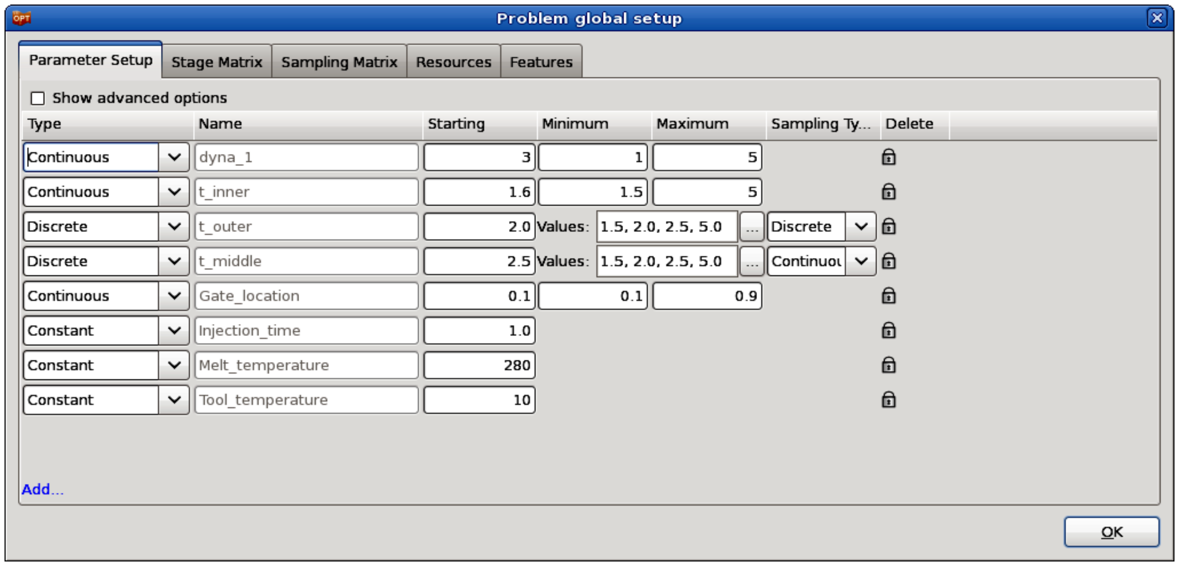
Option	Selections	Description
Operation	Copy Move Delete	Available operations
Source File		Name of source file
Destination File		Name of destination file, wildcards are supported
Sequence	before after	Execute operation before or after executing the stage command
On Error	fail warn ignore	What to do if operation fails

4. Setup Dialog – Defining the Variables

This chapter discusses the conversion of parameters defined in input files to design variables of different types. Graphical features allow the user to view file sources of parameters and the activation or de-activation of variables for selected samplings. Resource definitions and other global features are also available in this dialog.

4.1. Parameter Setup

Parameters defined in the input files of the stages are automatically displayed in the **Parameter Setup** panel, Figure 4-1. The names of these parameters are not editable, and they cannot be deleted as indicated by the lock symbol displayed in the *Delete* column. If only a name and value are specified in the stage input file, the parameter type is set to *Constant* by default.



Type	Name	Starting	Minimum	Maximum	Sampling Ty...	Delete
Continuous	dyna_1	3	1	5		🔒
Continuous	t_inner	1.6	1.5	5		🔒
Discrete	t_outer	2.0	Values:	1.5, 2.0, 2.5, 5.0	Discrete	🔒
Discrete	t_middle	2.5	Values:	1.5, 2.0, 2.5, 5.0	Continuou	🔒
Continuous	Gate_location	0.1	0.1	0.9		🔒
Constant	Injection_time	1.0				🔒
Constant	Melt_temperature	280				🔒
Constant	Tool_temperature	10				🔒

Figure 4-1: Setup Dialog – Parameter Setup panel in LS-OPTui

Other attributes such as parameter values or discrete sets defined in the input files are also displayed here, but can be overridden. The desired parameter type and other appropriate options can also be specified, Table 4-1.

Advanced non-mandatory options, e.g. initial range, can be specified by selecting the *Show advanced options* checkbox, Table 4-2.

Additional (non-file) parameters, can be defined using the *Add* button at the bottom of the panel.

Table 4-1: Parameter Setup options to be specified for each parameter

Option	Description		Reference
Type	Parameter type:		
	Continuous	Continuous variable	-
	Constant	Constant value	Section 4.1.1
	Dependent	Parameter depending on other parameters	Section 4.1.2
	Discrete	Discrete variable	Section 4.1.3
	String	Categorical Variable	Section 4.1.3
	String Constant	Constant using string values	Section 4.1.1
	Transfer Variable	Parameter treated as variable at upper level and constant at lower level (multi-level optimization)	Section 4.1.4
	Transfer String Variable	Transfer Variable using string values	Section 4.1.4
	Response Variable	Variable which inherits the value of a response	Section 4.1.5
	Noise	Probabilistic variable described by a statistical distribution	Section 4.1.5
Name	Parameter name. If the parameter is imported from a stage input file, the name is not editable		-
Starting	Initial value of the variable, used in baseline 1.1 run		-
Minimum	Lower bound of the design space		-
Maximum	Upper bound of the design space		-
Values	List of allowable values for discrete and string variable		Section 4.1.3

Definition	Mathematical expression specifying a dependent parameter	Section 4.1.2
Distribution	Statistical distribution of a probabilistic variable	*Sec. 8.1.7 [1]*
Sampling Type	Sampling type for discrete variable: continuous or discrete	Section 4.1.3
Edit Input Parameter References	Set the relation of a transfer variable with another variable	Section 4.1.4

Table 4-2: Parameter Setup advanced options

Option	Description	Reference
Init. Range	Design space subregion size used in the first iteration	*Sec. 8.1.8 [1]*

Table 4-3: Parameter Setup options

Option	Description	Reference
Show advanced options	Shows Init. Range and Saddle Direction option for each parameter	Table 4-2
Noise Variable Subregion Size (in Standard Deviations)	Bounds are required for noise variables to construct the metamodels. The bounds are taken to a number of standard deviations away from the mean; the default being two standard deviations of the distribution. In general, a noise variable is bounded by the distribution specified and does not have upper and lower bounds similar to control variables.	-
Enforce Variable Bounds	Assigning a distribution to a control value may result in designs exceeding the bounds on the control variables. The default is not to enforce the bounds.	-

4.1.1. Constants

Each variable above can be modified to be a constant. A constant can be a number or a string. Constants are used:

1. to define constant values in the input file such as π, e or any other constant that may relate to the optimization problem, e.g. initial velocity, event time, integration limits, etc.

2. if native parameters defined in the input file are not to be used as optimization parameters.
3. to convert a variable to a constant. This requires only changing the designation variable to constant in the command file without having to modify the input template. The number of optimization variables is thus reduced without interfering with the template files. Variables can also be eliminated by unchecking them in the Sampling matrix *(see Section 8.3: Sampling Matrix [1])*

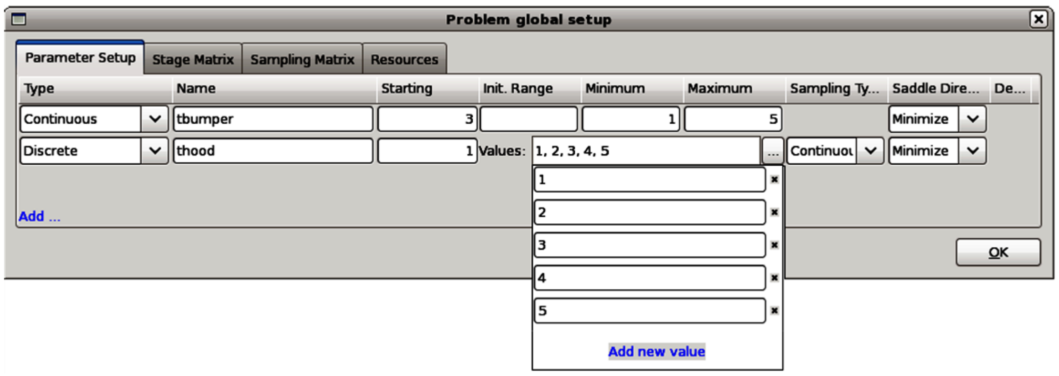
4.1.2. Dependent variables

Dependent variables are functions of the basic variables and are required to define quantities that have to be replaced in the input template files, but which are dependent on the optimization variables. They do therefore not contribute to the size of the optimization problem. Dependents can be functions of dependents.

Dependent variables are specified using mathematical expressions *(see Appendix F: Mathematical Expressions [1])*.

The dependent variables can be specified in an input template and will therefore be replaced by their actual values.

4.1.3. Discrete and String variables



Type	Name	Starting	Init. Range	Minimum	Maximum	Sampling Ty...	Saddle Dire...	De...
Continuous	tbumper	3		1	5		Minimize	
Discrete	thood	1	Values: 1, 2, 3, 4, 5			Continuo	Minimize	

1
2
3
4
5

Figure 4-2: Definition of discrete values

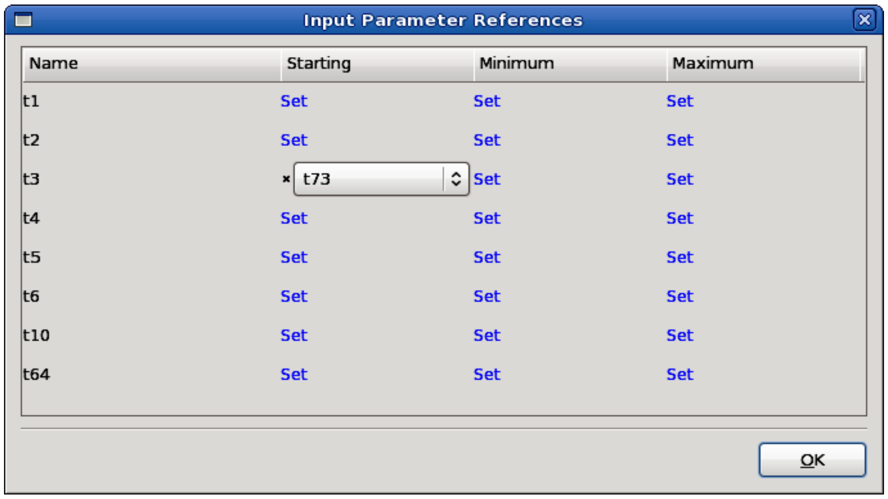
For Discrete variables, a list of allowable values has to be specified. This can be done in the **Parameter Setup** dialog using the **…** button to the right of the *Values* textfield of the respective parameter, Figure 4-2. A list opens up showing the already defined values, a textfield to enter a new value appears by selecting the *Add new value* button or by using the return key.

For *String* variables, allowable string values are defined in the same way. The string values are internally treated as integers in LS-OPT. The mapping of these integer values and the actual strings are stored in the *StringVar.lsox* database in the work directory.

In addition to a list of values, the sampling type has to be specified for discrete variables. By default, the discrete variables are treated as continuous variables for generating experimental designs. The optimal values will assume an allowable value. If discrete sampling is selected, all experimental design points use allowable values. If possible, a continuous sampling is recommended, because it usually leads to a better distribution of the points within the design space and hence to a better metamodel quality.

4.1.4. Transfer variables

Transfer variables are used in the context of multilevel optimization (see Section **3.3.3**). These variables are sampled in one of the levels, but these sample values are passed down to the lower levels where these are treated as constants. Transfer variables can be referenced by preceding higher levels or by other variables in the same level. Within the same level, a transfer variable can be the starting value or the lower/upper bound for another variable (Figure 4-3).



Name	Starting	Minimum	Maximum
t1	Set	Set	Set
t2	Set	Set	Set
t3	t73	Set	Set
t4	Set	Set	Set
t5	Set	Set	Set
t6	Set	Set	Set
t10	Set	Set	Set
t64	Set	Set	Set

Figure 4-3: Input Parameter References. Transfer Variable t73 is set as the initial value for t3.

4.1.5. Response variables

Response variables are used to define variables which inherit the values of responses. As such they are critical to the ICME setup. The main purpose is to allow substitution of response values in input files. The response must be calculated in an ancestor of the stage in which the substitution is done.

1. The main parameter setup allows the user to link a parameter to a response (See Figure 4-6). This selection causes the selected parameter value to be replaced by a response value defined in an ancestor stage. The transferred response value is substituted into the input file(s) of stages downstream where the parameters are defined.
2. The response value to be linked can be any response value which was directly extracted from the solver database or a mathematical expression involving any variables, dependents, histories or responses defined in any parent stages.
3. Response variables can be transferred between any two stages of a particular thread. They do not need to be consecutive as long as the response is defined in a stage which comes before the stage where the substitution is done.
4. A specific response can be linked to any number of parameters.

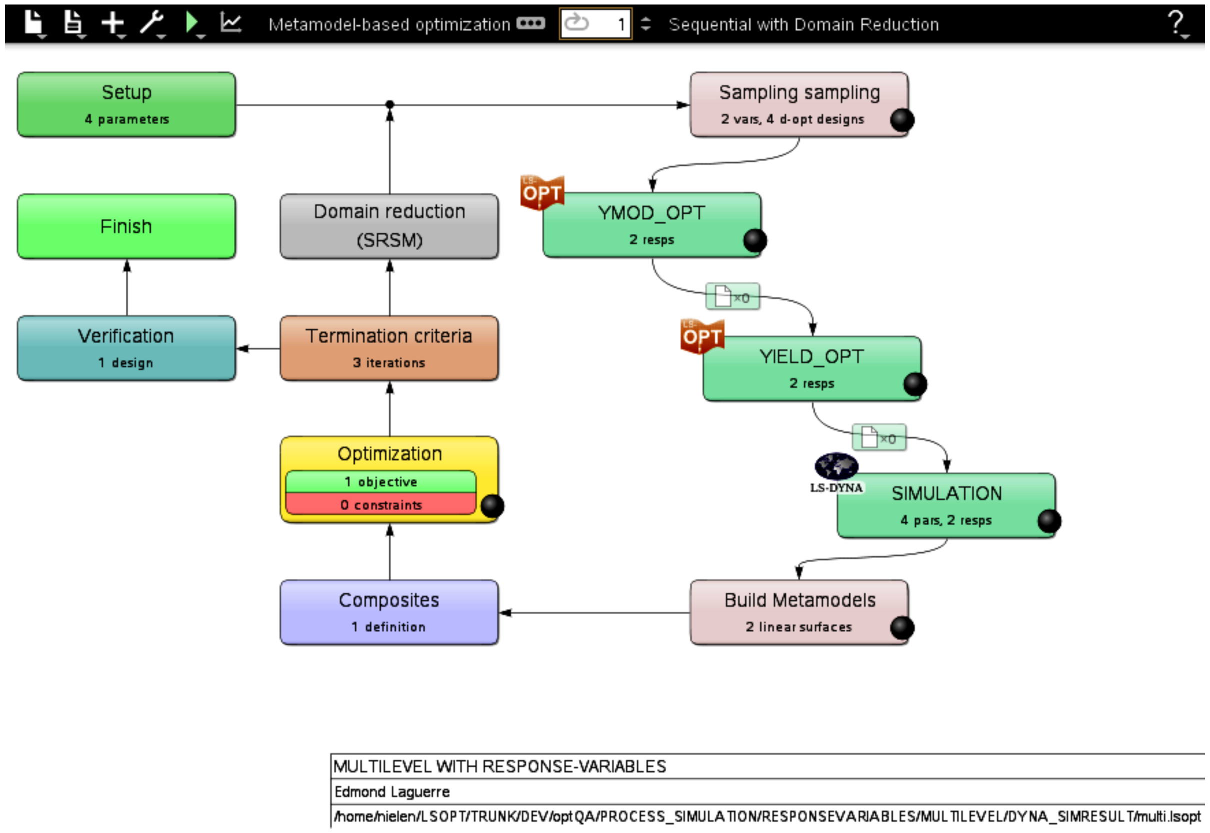
5. Response variables are not independent design variables, so have no effect on the sampling.

Example

The example is explained using the series of figures below. The optimization consists of an outer loop with three stages. The first stage is also an optimization loop which calibrates a parameter YMod to produce YMod_OPT. The second stage uses the optimized YMod_OPT as a constant parameter but optimizes a second variable Yield to produce Yield_OPT.

After the first two stages, YMod_OPT and Yield_OPT are converted using mathematical expressions and then transferred as material constants to a vehicle simulation stage. The outer loop optimizes the vehicle design variables tbumper and thood.

Figure 4-4 through Figure 4-11 show various parts of the problem setup.

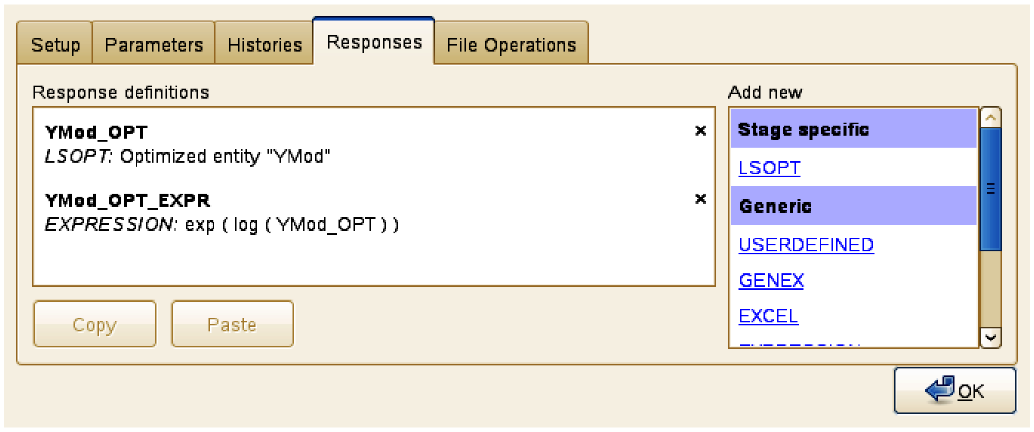


MULTILEVEL WITH RESPONSE-VARIABLES

Edmond Laguerre

/home/nielen/LSOPT/TRUNK/DEV/optQA/PROCESS_SIMULATION/RESPONSEVARIABLES/MULTILEVEL/DYNA_SIMRESULT/multi.lsopt

Figure 4-4: LS-OPT Problem multilevel setup. The first two stages (YMOD_OPT) and (YIELD_OPT) are sublevel optimization stages. YMOD_OPT produces an optimal material parameter YMod_OPT and converts it to YMod_OPT_EXPR using an expression. This value is transferred to the parameter YModRV defined as an input parameter to the YIELD_OPT stage. The YIELD_OPT stage therefore uses this value as a constant but optimizes a second variable Yield to produce Yield_OPT which is then converted to Yield_OPT_EXPR. Both YMod_OPT_EXPR and Yield_OPT_EXPR are then transferred to the SIMULATION stage as input parameters. The outer loop depicted here optimizes over design variables tbumper and thood to minimize vehicle intrusion.



Response definitions

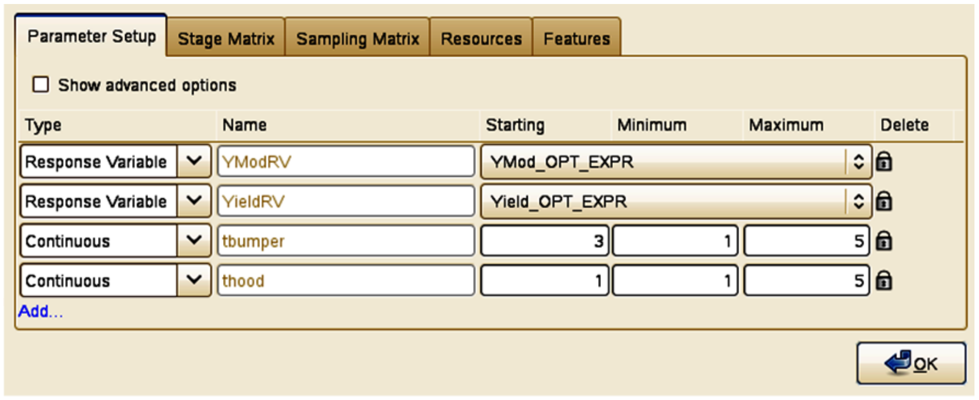
YMod_OPT
LSOPT: Optimized entity "YMod"

YMod_OPT_EXPR
EXPRESSION: exp (log (YMod_OPT))

Add new

Stage specific
LSOPT
Generic
USERDEFINED
GENEX
EXCEL

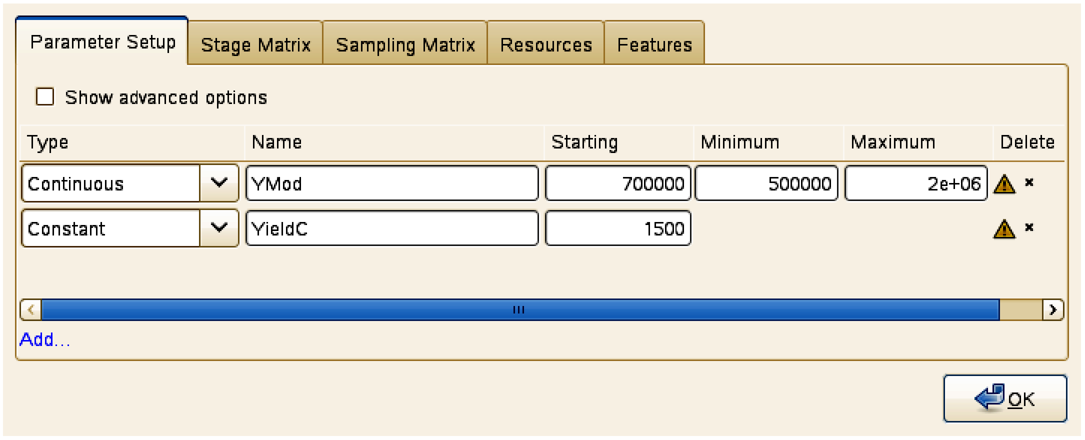
Figure 4-5: Response output definition for Stage YMOD_OPT.



☐ Show advanced options

Type	Name	Starting	Minimum	Maximum	Delete
Response Variable	YModRV	YMod_OPT_EXPR			
Response Variable	YieldRV	Yield_OPT_EXPR			
Continuous	tbumper	3	1	5	
Continuous	thood	1	1	5	

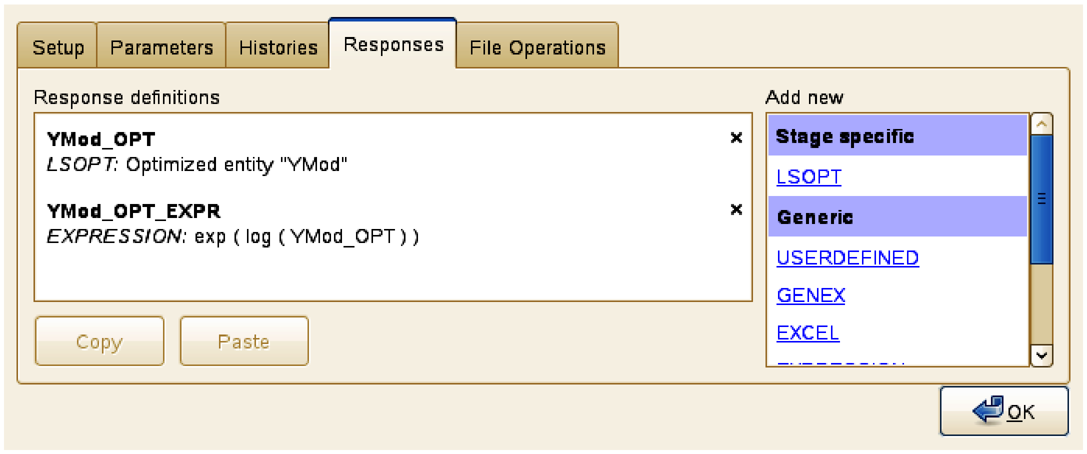
Figure 4-6: The main parameter setup (clicking green box at top left of Figure 4-4) to define two response-variables YModRV and YieldRV. These respectively link to YMod_OPT_EXPR and Yield_OPT_EXPR produced by the parent optimization stages. The parameters tbumper and thood are optimization variables used in the outer loop.



☐ Show advanced options

Type	Name	Starting	Minimum	Maximum	Delete
Continuous	YMod	700000	500000	2e+06	
Constant	YieldC	1500			

Figure 4-7: Input parameters for the YMOD_OPT stage. YMod is an optimization variable defined in this stage while YieldC is a constant.

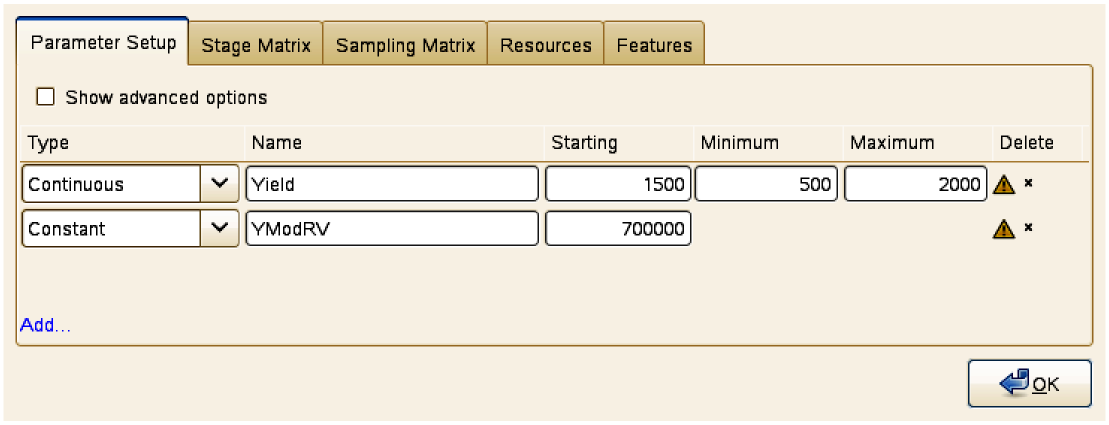


Response definitions

YMod_OPT
LSOPT: Optimized entity "YMod"

YMod_OPT_EXPR
EXPRESSION: exp (log (YMod_OPT))

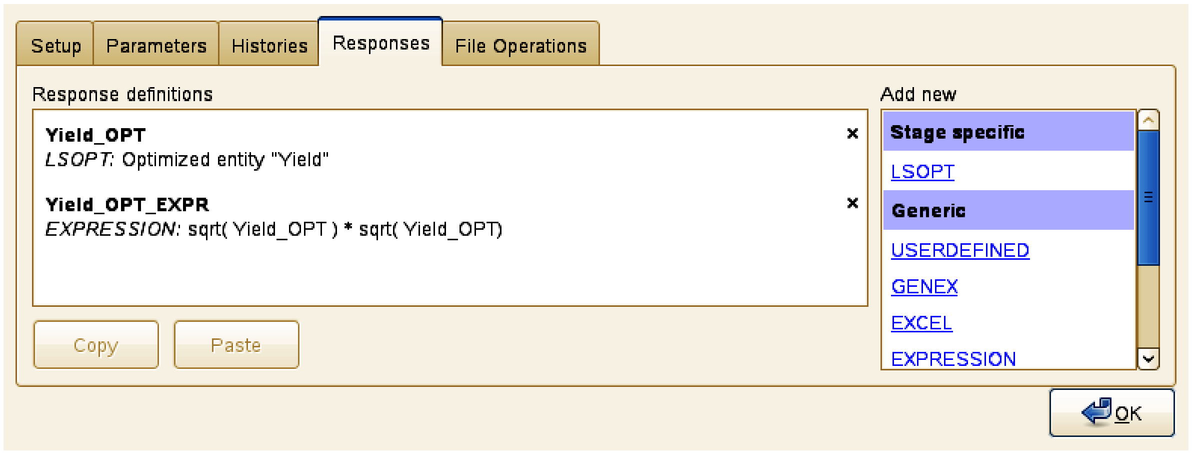
Figure 4-8: Response output definition for Stage YMOD_OPT.



☐ Show advanced options

Type	Name	Starting	Minimum	Maximum	Delete
Continuous	Yield	1500	500	2000	⚠ ×
Constant	YModRV	700000			⚠ ×

Figure 4-9: Input parameters for the YIELD_OPT stage. Yield is an optimization variable defined in this stage. YModRV is a response-variable replaced by YMod_OPT_EXPR (see Figure 4-6 for definition).

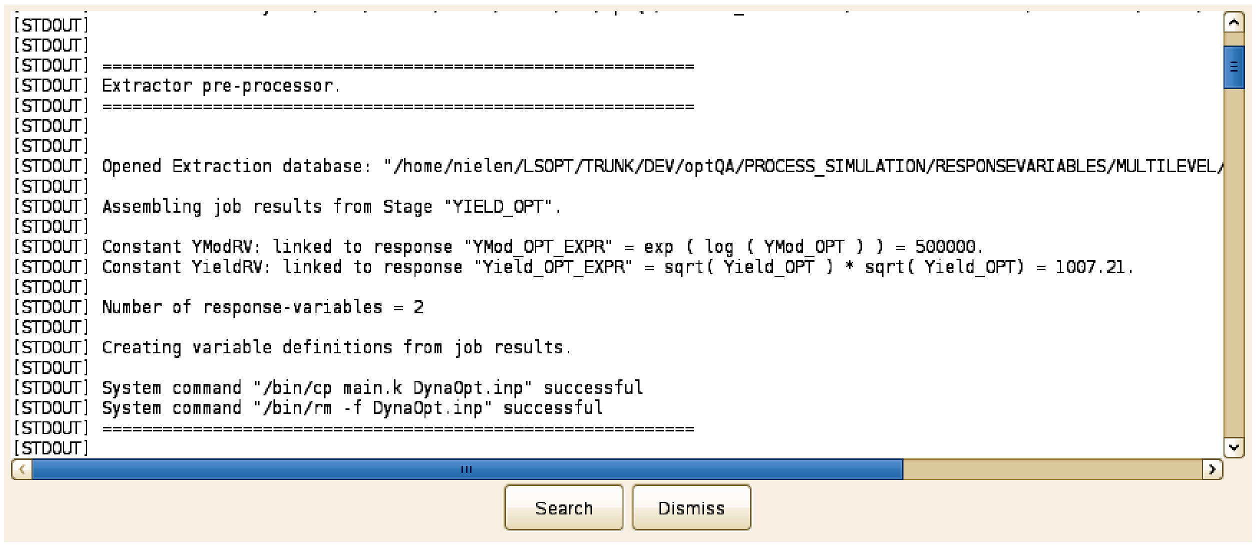


Response definitions

Yield_OPT
LSOPT: Optimized entity "Yield"

Yield_OPT_EXPR
EXPRESSION: sqrt(Yield_OPT) * sqrt(Yield_OPT)

Figure 4-10: Response output definition for Stage YIELD_OPT.



```
[STDOUT]
[STDOUT]
[STDOUT] ============================================================
[STDOUT] Extractor pre-processor.
[STDOUT] ============================================================
[STDOUT]
[STDOUT]
[STDOUT] Opened Extraction database: "/home/nielen/LSOPT/TRUNK/DEV/optQA/PROCESS_SIMULATION/RESPONSEVARIABLES/MULTILEVEL/
[STDOUT]
[STDOUT] Assembling job results from Stage "YIELD_OPT".
[STDOUT]
[STDOUT] Constant YModRV: linked to response "YMod_OPT_EXPR" = exp ( log ( YMod_OPT ) ) = 500000.
[STDOUT] Constant YieldRV: linked to response "Yield_OPT_EXPR" = sqrt( Yield_OPT ) * sqrt( Yield_OPT) = 1007.21.
[STDOUT]
[STDOUT] Number of response-variables = 2
[STDOUT]
[STDOUT] Creating variable definitions from job results.
[STDOUT]
[STDOUT] System command "/bin/cp main.k DynaOpt.inp" successful
[STDOUT] System command "/bin/rm -f DynaOpt.inp" successful
[STDOUT] ============================================================
[STDOUT]
```

Figure 4-11: Job log of SIMULATION stage of the example (the display represents the pre-processor phase prior to simulation). Note the linking of the two parameters to responses.

5. History and Response Results

This chapter describes the specification of the history or response results to be extracted from the stage database. A history is a vector or curve data, whereas a response is a scalar value. Responses can be used to define objectives or constraints *(Chapter 11: Optimization Dialog – Objectives, Constraints and Algorithms[1])*. Interfaces for result extraction from LS-DYNA and MSC-NASTRAN output files are available, as well as mathematical expressions, file import, an interface for extraction of values from ASCII database and a user-defined interface where any program may be used for result extraction. The dialogs are accessible from the **Stage** dialog **Histories** and **Responses** tab, respectively.

5.1. Defining histories and responses

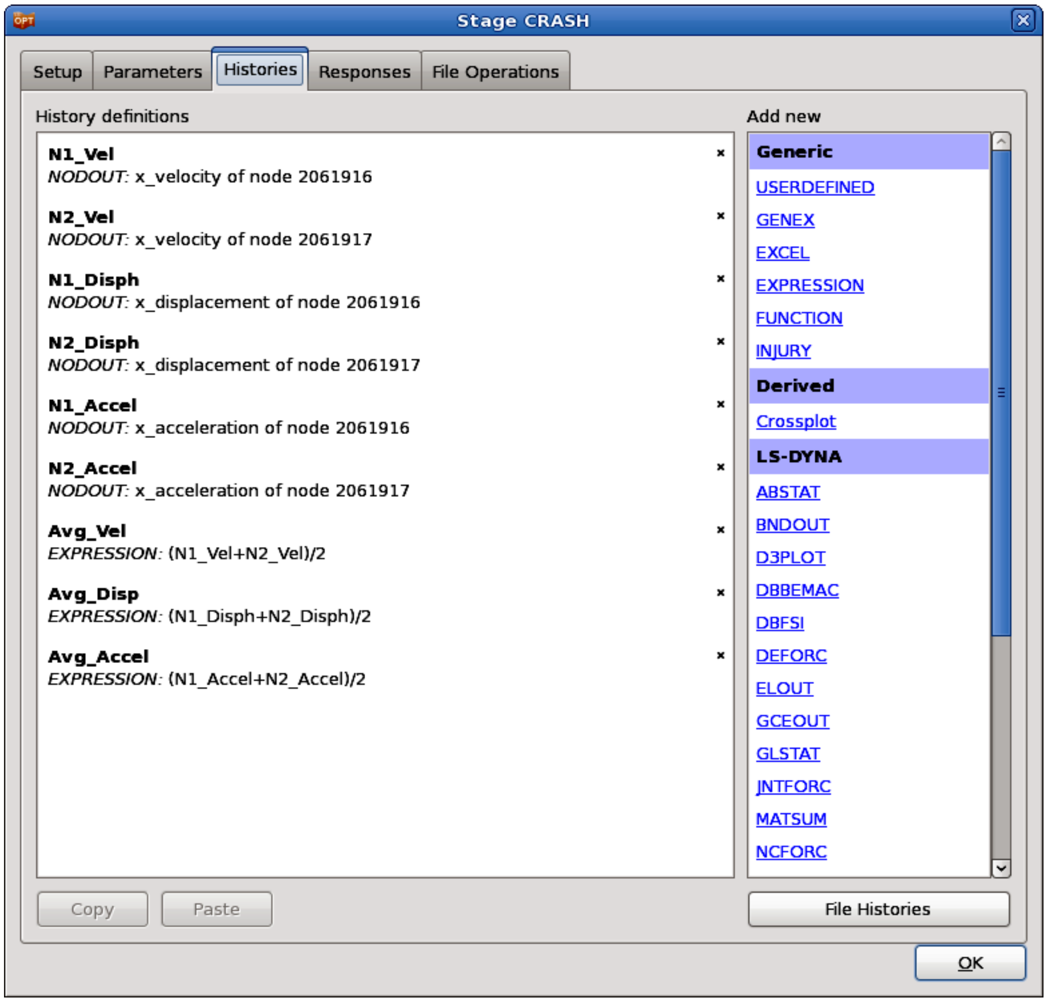
A history or a response can be defined by using the interfaces in the **Histories** and **Responses** tab of the **Stage** dialog, respectively, Figure 5-1. To add a new definition, select the respective interface from the list on the right. The available interfaces are explained in Table 5-1. To edit an already defined history or response, double-click on the respective entry from the list on the left. Histories and responses may be deleted using the *delete* icon on the right of the respective definition.

There are five types of interfaces:

- Standard LS-DYNA, MSC-Nastran or LS-OPT result interfaces. These interfaces provide access to the LS-DYNA binary databases (d3plot or binout, d3hsp or d3eigv), the Nastran and LS-OPT database, respectively. The interfaces are an integral part of LS-OPT.
- User specified interface programs. These can reside anywhere. The user specifies the full path.
- Mathematical expressions.
- GenEx. This interface allows the user to extract selected field values from a text file.
- Excel.

The extraction of responses consists of a definition for each response and a single extraction command or mathematical expression. A response is often the result of a mathematical operation of a response history, but can be extracted directly using the standard LS-DYNA interface (see Section 5.1.1) or a user-defined interface.

Each extracted response is identified by a name and the settings to be specified using the respective interface.



N1_Vel
NODOUT: x_velocity of node 2061916

N2_Vel
NODOUT: x_velocity of node 2061917

N1_Disph
NODOUT: x_displacement of node 2061916

N2_Disph
NODOUT: x_displacement of node 2061917

N1_Accel
NODOUT: x_acceleration of node 2061916

N2_Accel
NODOUT: x_acceleration of node 2061917

Avg_Vel
EXPRESSION: (N1_Vel+N2_Vel)/2

Avg_Disp
EXPRESSION: (N1_Disph+N2_Disph)/2

Avg_Accel
EXPRESSION: (N1_Accel+N2_Accel)/2

Figure 5-1: Histories definition in the GUI

Table 5-1: Interfaces for Response and History extraction

Option		Description	Reference
Generic	USERDEFINED	Result extraction using any script or program	*Section 6.13 [1]*
	FILE	Result extraction from a text file (responses only)	*Section 6.14 [1]*
	GENEX	Tool for extracting results from text files	*Chapter 7 [1]*

	EXCEL	Result extraction from an Excel document	*Section 6.17[1]*
	EXPRESSION	Definition of mathematical expressions using previously defined entities	*Section 6.4.1 [1]*
	FUNCTION	Expressions using previously defined histories	*Section 6.4.3 [1]*
	INJURY	Injury criteria	*Section 6.5 [1]*
Derived	Crossplot	Crossplot (History only)	*Section 6.4.1 [1]*
LS-DYNA	ABSTAT	Binout interface	Section 5.2.1
	BNDOUT	Binout interface	Section 5.2.1
	D3PLOT	D3plot interface	Section 5.2.2
	DBBEMAC	Binout interface	Section 5.2.1
	DBFSI	Binout interface	Section 5.2.1
	DEFORC	Binout interface	Section 5.2.1
	ELOUT	Binout interface	Section 5.2.1
	FLD	Metal Forming results (Response only)	*Section 6.3.2 [1]*
	FREQUENCY	D3eigv interface (Response only)	*Section 6.2.5 [1]*
	GCEOUT	Binout interface	Section 5.2.1
	GLSTAT	Binout interface	Section

		5.2.1
JNTFORC	Binout interface	Section 5.2.1
MASS	D3hsp interface (Response only)	Section 5.2.3
MATSUM	Binout interface	Section 5.2.1
NCFORC	Binout interface	Section 5.2.1
NODOUT	Binout interface	Section 5.2.1,
NODFOR	Binout interface	Section 5.2.1
PSTRESS	Metal Forming results (Response only)	*Section 6.3.3 [1]*
RBDOUT	Binout interface	Section 5.2.1
RCFORC	Binout interface	Section 5.2.1
RWFORC	Binout interface	Section 5.2.1
SBTOUT	Binout interface	Section 5.2.1
SECFORC	Binout interface	Section 5.2.1
SPCFORC	Binout interface	Section 5.2.1
SPHOUT	Binout interface	Section 5.2.1
SWFORC	Binout interface	Section 5.2.1

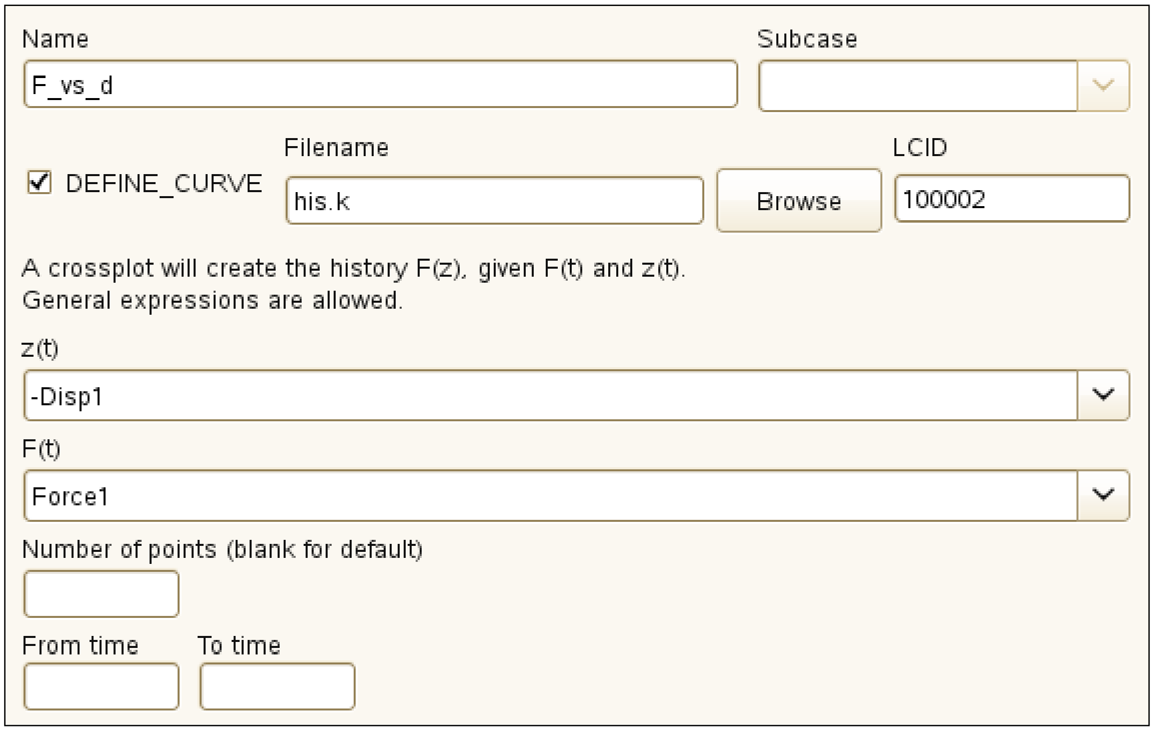
	THICK	Metal Forming results (Response only)	*Section 6.3.1 [1]*
LS-OPT	LSOPT	Optimized inner level variables, responses, composites, objective functions, constraints, histories and reliability statistics	Section 5.3.2, Section 5.3.1
	LSOPT_STATISTICS	Statistical values produced by a Monte Carlo analysis (Response only)	*Section 6.16.3 [1]*
File Histories		Global file histories (History only)	*Section 6.19 [1]*

5.1.1. Result extraction

Each simulation run is immediately followed by a result extraction to create the `history`.*n* and `response`.*n* files for that particular design point. For distributed simulation runs, this extraction process is executed on the remote machine. The `history`.*n* and `response`.*n* files are subsequently transferred to the local run directory. If the extraction on the remote machine is not successful, it is done again on the local machine. Hence programs and scripts needed for result extraction do not have to be accessible from the remote machine. These results are stored in the `AnalysisResults_`*`n`*`.lsox` database.

5.1.2. Creating a history file with an LS-DYNA *DEFINE_CURVE keyword

The DEFINE_CURVE selection allows the creation of an LS-DYNA include file (e.g. `his.k`) with the *DEFINE_CURVE keyword and history data. The LCID, which represents the load curve ID required by LS-DYNA, should be entered in the appropriate text box. See Figure 5-2.


Name

F_vs_d

Subcase

☑ DEFINE_CURVE

Filename

his.k

LCID

100002

A crossplot will create the history F(z), given F(t) and z(t).
General expressions are allowed.

z(t)

-Disp1

F(t)

Force1

Number of points (blank for default)

From time

To time

Figure 5-2: Interface to define a crossplot history. An option has been selected to produce the curve in the LS-DYNA *DEFINE_CURVE format output to the file his.k.This file can be used as an *INCLUDE file.

5.2. Extracting history and response quantities: LS-DYNA

LS-OPT provides interfaces for history and response result extraction from the binout, d3plot, d3hsp and d3eigv databases. The user must ensure that the LS-DYNA program will provide the output files required by LS-OPT.

The options for the extraction of LS-DYNA responses and histories are identical, except for the selection attribute.

Aside of the standard interfaces that are used to extract any particular data item from the database, specialized responses for metal-forming are also available. The computation and extraction of these secondary responses are discussed in *Section 6.3: Extracting metal forming response quantities: LS-DYNA [1]*.

5.2.1. LS-DYNA binout results

All LS-DYNA history and response result extraction options except for D3PLOT, MASS and FREQUENCY interface with the LS-DYNA binout output. The BINARY flag in the respective *DATABASE_OPTION card and the desired entity ID in the *DATABASE_HISTORY_OPTION card has to be set correctly in the LS-DYNA input file. Note that the LS-DYNA executable is interpreted as a single process (SMP) by LS-OPT, hence the default binary flag value 0 is not supported.

The response options are an extension of the history options – a history will be extracted as part of the response extraction.

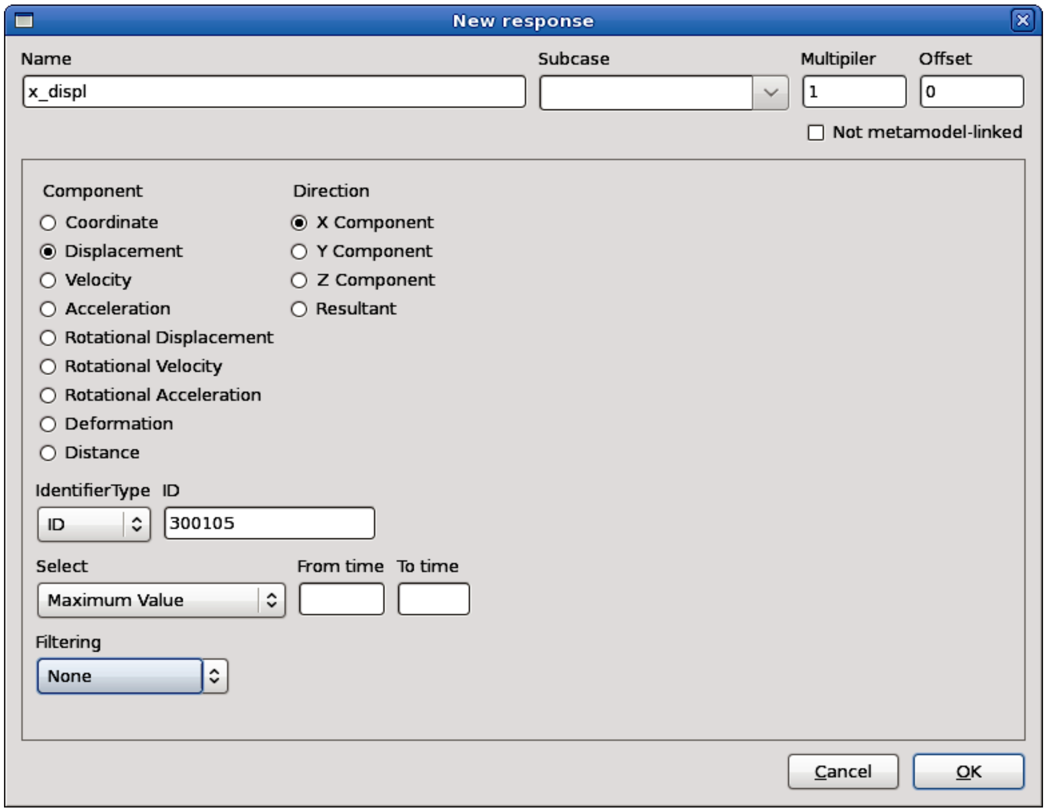
Results can be extracted for the whole model or a finite element entity such as a node or element. For shell and beam elements the through-thickness position can be specified as well.

Filtering and averaging options are available for histories and responses.

For responses, the *Select* attribute has to be specified to extract a scalar value from the curve. The optional attributes *From time* and *To time* can be specified to slice the curve before extracting the requested scalar value. The defaults are 0 and the end value of the history.

These operations will be applied in the following order: averaging or filtering, and slicing.

The available results types and components are listed in *Appendix A: LS-DYNA Binout Commands [1]* and *Appendix B: LS-DYNA Binout Components [1]*.



New response

Name: x_displ
Subcase:
Multipiler: 1
Offset: 0
☐ Not metamodel-linked

Component: ○ Coordinate [✓ ● Displacement] ○ Velocity ○ Acceleration ○ Rotational Displacement ○ Rotational Velocity ○ Rotational Acceleration ○ Deformation ○ Distance

Direction: [✓ ● X Component] ○ Y Component ○ Z Component ○ Resultant

IdentifierType: ID
ID: 300105

Select: Maximum Value
From time:
To time:

Filtering: None

Figure 5-3: Response extraction: LS-DYNA NODOUT interface

5.2.2. LS-DYNA d3plot results

The D3PLOT interface is related to the Binout interface. The D3PLOT results differ from the Binout commands in that a response or history can be collected over a whole part. For example, the maximum stress can be evaluated in a part or over the whole model. Results can also be extracted for a finite element entity such as a node or element. For shell and beam elements the through-thickness position can be specified as well. Element results such as stresses will be averaged in order to create the NODE results.

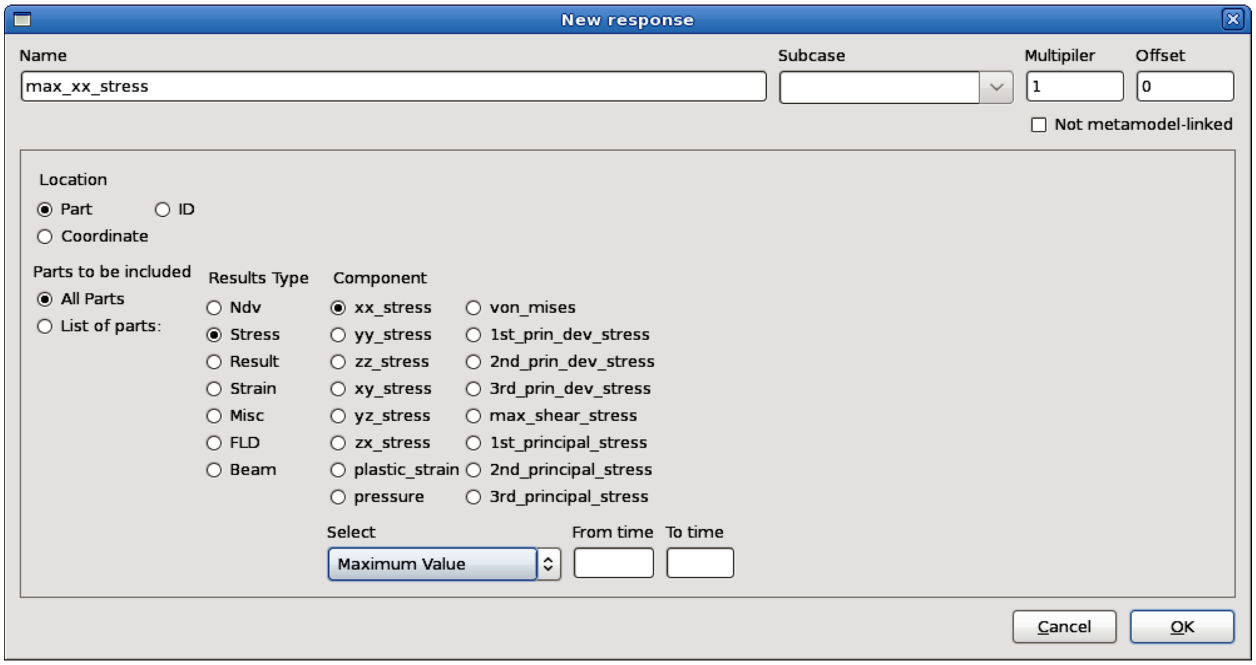
If the location of extraction is specified by *x,y,z* coordinates, the quantity will be extracted from the element nearest to *x,y,z* at the time of reference state. Only elements included in the `*SET_SOLID_GENERAL` element set are considered (only the `PART` and `ELEMENT` options).

The response options are an extension of the history options – a history will be extracted as part of the response extraction. For responses, the *Select* attribute has to be specified to extract a scalar value from the curve. The optional attributes *From time* and *To time* can be specified to slice the curve before extracting the requested scalar value. The defaults are 0 and the end value of the history.

If the selection must be done over parts as well, the maximum, minimum or average can be selected for the part, followed by the selection of the maximum, minimum, or average over time.

The available results types and components are listed in *Appendix C: LS-DYNA D3Plot Commands [1]* and *Appendix D: LS-DYNA D3Plot Components [1]*.

The LS-PREPOST fringe plot capability can be used for the graphical exploration and troubleshooting of the data.


New response

Name: max_xx_stress
Subcase:
Multipiler: 1
Offset: 0
☐ Not metamodel-linked

Location: [✓ Part] ID Coordinate

Parts to be included: [✓ All Parts] List of parts:

Results Type: Ndv [✓ Stress] Result Strain Misc FLD Beam

Component: [✓ xx_stress] yy_stress zz_stress xy_stress yz_stress zx_stress plastic_strain pressure von_mises 1st_prin_dev_stress 2nd_prin_dev_stress 3rd_prin_dev_stress max_shear_stress 1st_principal_stress 2nd_principal_stress 3rd_principal_stress

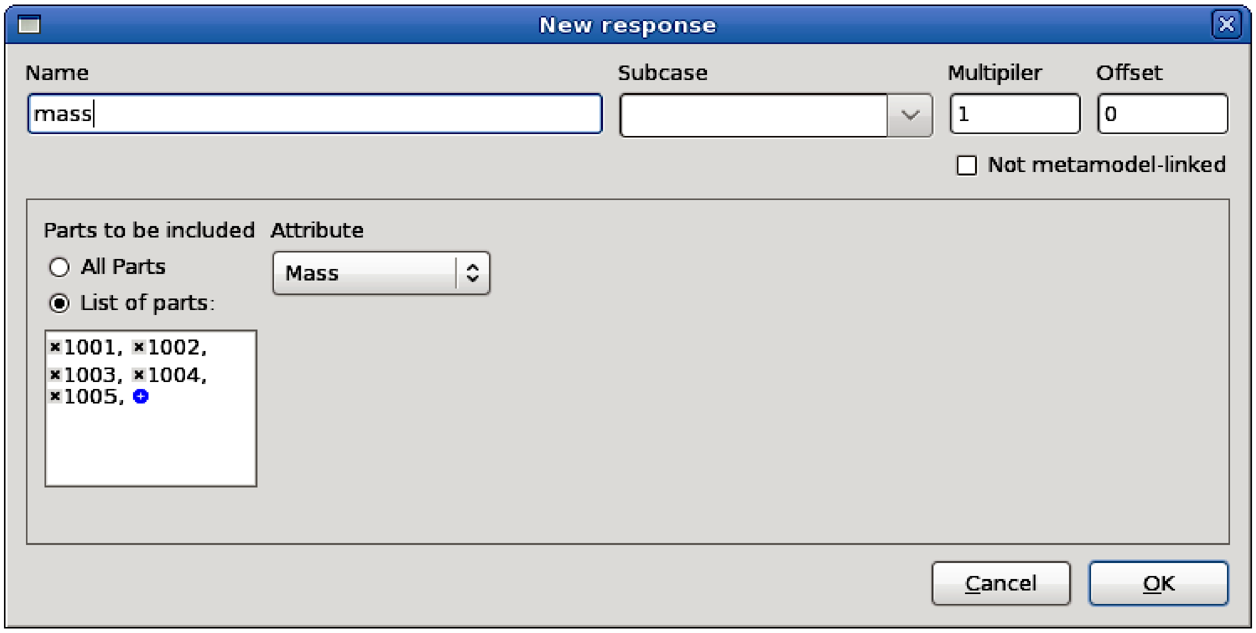
Select: Maximum Value
From time:
To time:

Figure 5-4: Response extraction from d3plot

5.2.3. Mass – Interfacing with d3hsp

The `MASS` response interfaces with the LS-DYNA output file `d3hsp`. The Mass and related entities, Figure 5-5 and Table 5-2, can be extracted for the whole model or a list of parts.

Values are summed if more than one part is specified (so only the mass value will be correct). However for the full model (part specification omitted) the correct values are given for all the quantities.



Name: mass
Subcase:
Multipiler: 1
Offset: 0
☐ Not metamodel-linked
Parts to be included
○ All Parts
● List of parts:
1001, 1002, 1003, 1004, 1005,
Attribute: Mass

Figure 5-5: Interface for extraction of Mass and related entities from LS-DYNA output d3hsp

Table 5-2: Mass item description

Item	Description	
Parts to be included	Entity is extracted for the entire model or for the part IDs specified in the list.	
Attribute	Type of mass quantity:	
	Mass	Mass
	Principal Inertias	Component `I11, I22, I33`
	Inertia Tensor	Component `IXX, IXY, IXZ, IYX, IYY, IYZ, IZX, IZY, IZZ`
	Mass Center	Component *X*-Coordinate, *Y*-Coordinate or *Z*-Coordinate of mass center

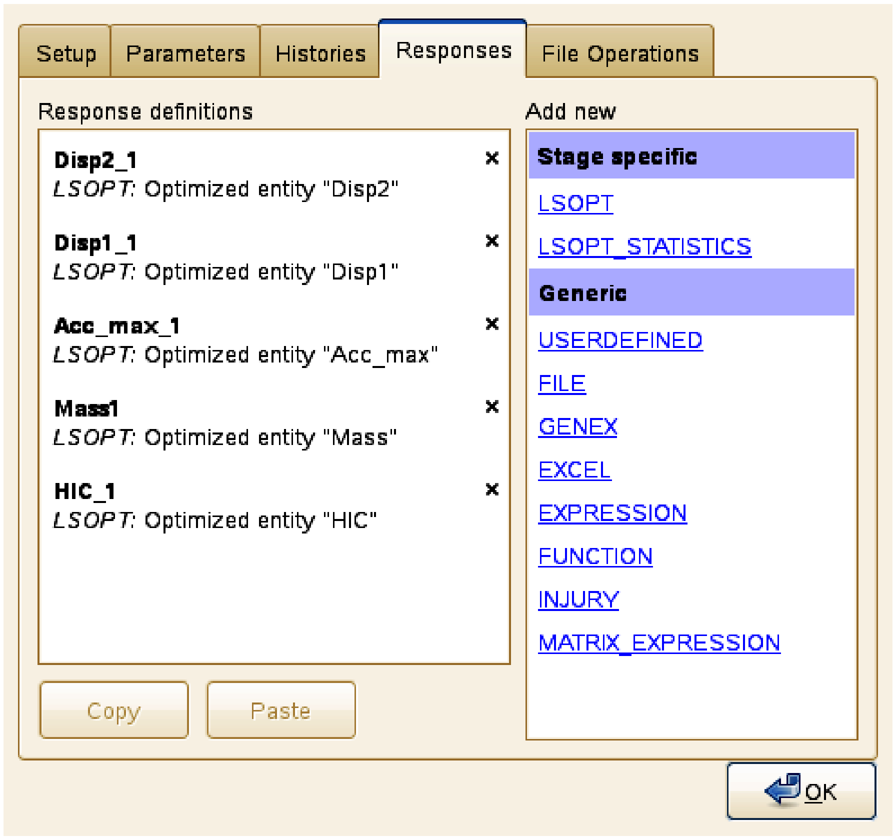
5.3. Extraction of LS-OPT entities

5.3.1. LS-OPT responses

The LS-OPT stage is used in the context of multilevel optimization, which involves running an inner level optimization within an outer level optimization. Each outer level sample evaluation, i.e. LS-OPT stage evaluation, involves an inner optimization. The results of these evaluations consist of entities that are optimized with respect to the inner level variables, which can be defined by the user as responses for the outer level LS-OPT setup. The LS-OPT stage and its responses are critical to the ICME setup, as it involves several nested optimization steps.

The response dialog of the LS-OPT stage type provides the option to define an *LSOPT* response, which lists the available entities optimized in the inner level. These entities can be the optimized inner level variables or the corresponding optimized responses, composites, objective functions or constraints (Figure 5-7). It is also possible to extract responses at any specific inner level iteration by clicking the '*Iteration*' radio button and providing the required iteration number.

Since the inner level can also be a Monte Carlo analysis, statistical values such as standard deviation, mean and probability of failure are available in the *LSOPT_STATISTICS* interface.



Disp2_1
LSOPT: Optimized entity "Disp2"

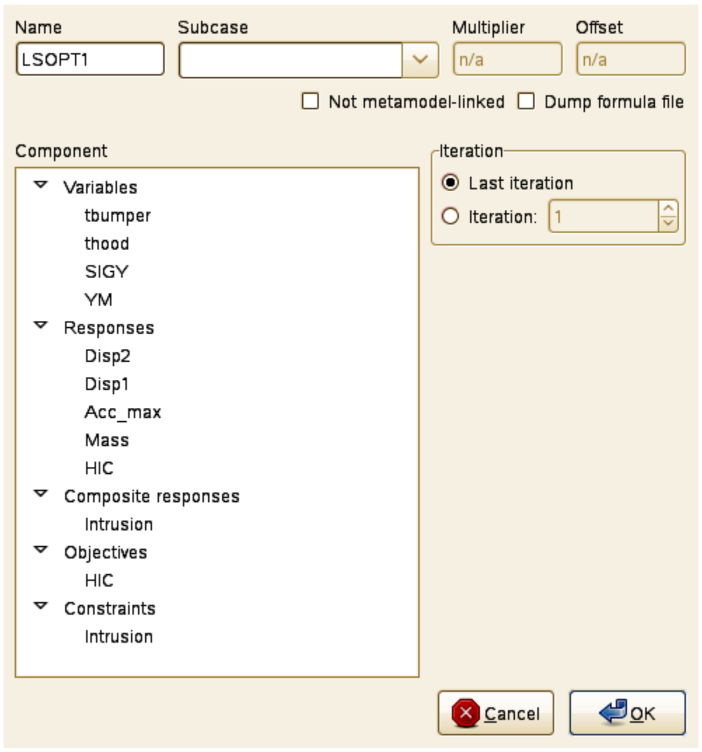
Disp1_1
LSOPT: Optimized entity "Disp1"

Acc_max_1
LSOPT: Optimized entity "Acc_max"

Mass1
LSOPT: Optimized entity "Mass"

HIC_1
LSOPT: Optimized entity "HIC"

Figure 5-6: Main dialog for the extraction of LS-OPT stage responses. A special category (LSOPT STATISTICS) is available for statistical results produced by a Monte Carlo analysis.


Name: LSOPT1
Subcase:
Multiplier: n/a
Offset: n/a

☐ Not metamodel-linked ☐ Dump formula file

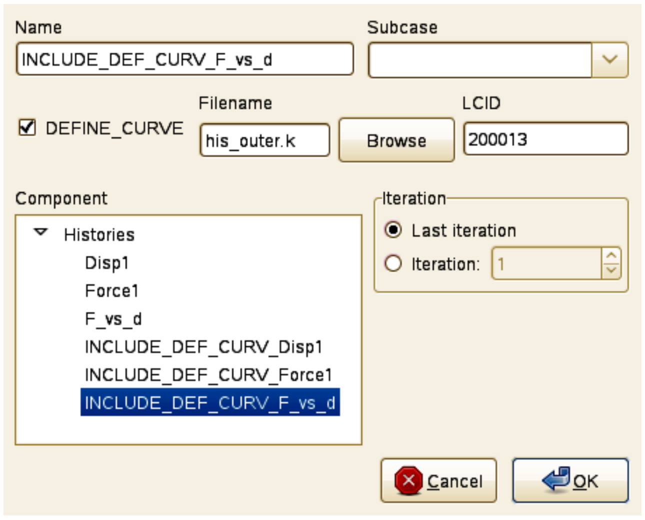
Component
- Variables
  - tbumper
  - thood
  - SIGY
  - YM
- Responses
  - Disp2
  - Disp1
  - Acc_max
  - Mass
  - HIC
- Composite responses
  - Intrusion
- Objectives
  - HIC
- Constraints
  - Intrusion

Iteration
[✓ Last iteration] Iteration: 1

Figure 5-7: Dialog for the extraction of LS-OPT optimal response results

5.3.2. LS-OPT histories

Figure 5-8 depicts the dialog for defining an LS-OPT history. Optimal histories produced by an optimization run can be extracted and converted to an LS-DYNA `*DEFINE_CURVE` keyword file. This file can then be inserted into a subsequent stage analysis as an include file. Multiple `*DEFINE_CURVE` data sets can be dumped in the same file.


Name: INCLUDE_DEF_CURV_F_vs_d
Subcase:

☑ DEFINE_CURVE
Filename: his_outer.k
LCID: 200013

Component
- Histories
  - Disp1
  - Force1
  - F_vs_d
  - INCLUDE_DEF_CURV_Disp1
  - INCLUDE_DEF_CURV_Force1
  - INCLUDE_DEF_CURV_F_vs_d

Iteration
[✓ Last iteration] Iteration: 1

Figure 5-8: Dialog for defining an LS-OPT history. The DEFINE_CURVE option has been selected to produce an LS-DYNA keyword file.

6. Running the Design Task

This chapter explains simulation job-related information and how to execute a design task from the graphical user interface as well as monitoring the status of the task and the simulation runs from the GUI.

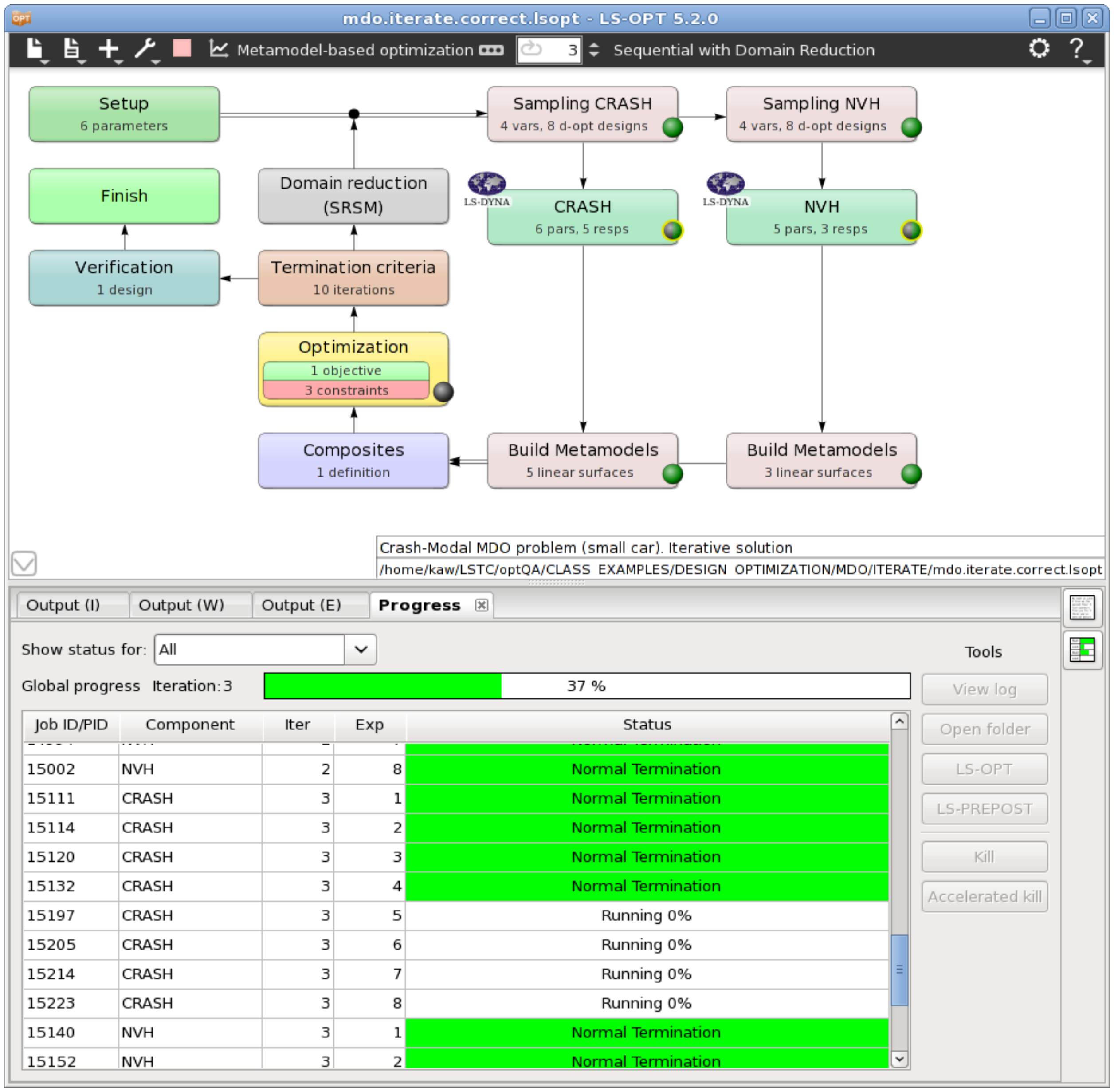
6.1. Running the design task

After setting up the task, run the design task using *Normal Run* or *Baseline Run* from the **Run** menu (▶) in the control bar of the main GUI as described in *Section 3.3: Run LS-OPT [1]*. If needed, previous results can be deleted using the **Clean** options in the Tools menu (🔧), *Section 3.4: Restarting – Clean from current iteration [1]*.

6.2. Analysis monitoring

While running LS-OPT, the status and progress of the task can be visualized in the main GUI, Figure 6-1.

The currently running iteration number is displayed in the control bar at the top (1). The stage LED of the currently running task process is highlighted (glows) in yellow while the green "pie" fraction inside the LED visualizes the solver progress. For the stage LED's, green and red is used for solver `Normal` and `Error` terminations, respectively. Double-clicking on a stage LED launches the **Progress** dialog described in Section 6.3. The status of individual jobs is also displayed in the *Progress* tab of the integrated output window, Section 6.3.2.



Metamodel-based optimization 3 Sequential with Domain Reduction

Crash-Modal MDO problem (small car). Iterative solution
/home/kaw/LSTC/optQA/CLASS_EXAMPLES/DESIGN_OPTIMIZATION/MDO/ITERATE/mdo.iterate.correct.lsopt

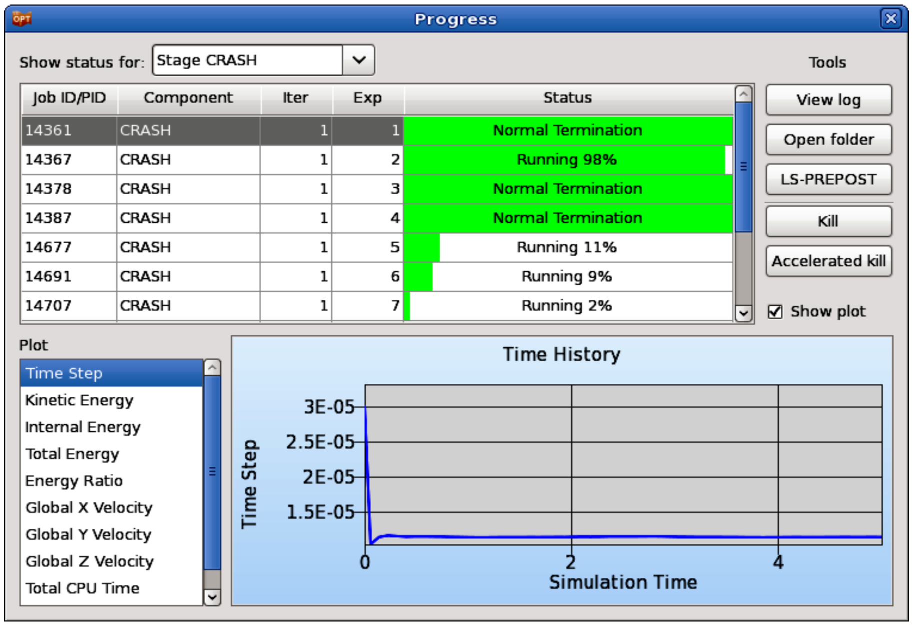
Show status for: All

Global progress Iteration: 3 37 %

Job ID/PID	Component	Iter	Exp	Status
15002	NVH	2	8	Normal Termination
15111	CRASH	3	1	Normal Termination
15114	CRASH	3	2	Normal Termination
15120	CRASH	3	3	Normal Termination
15132	CRASH	3	4	Normal Termination
15197	CRASH	3	5	Running 0%
15205	CRASH	3	6	Running 0%
15214	CRASH	3	7	Running 0%
15223	CRASH	3	8	Running 0%
15140	NVH	3	1	Normal Termination
15152	NVH	3	2	Normal Termination

Figure 6-1: Main GUI showing scheduled jobs in progress

6.3. Job monitoring – the Progress dialog



Show status for: Stage CRASH

Job ID/PID	Component	Iter	Exp	Status
14361	CRASH	1	1	Normal Termination
14367	CRASH	1	2	Running 98%
14378	CRASH	1	3	Normal Termination
14387	CRASH	1	4	Normal Termination
14677	CRASH	1	5	Running 11%
14691	CRASH	1	6	Running 9%
14707	CRASH	1	7	Running 2%

Figure 6-2: Progress dialog displaying progress of stage runs

Table 6-1: Tools for selected run

Tool	Description	Reference
View log	Opens *job_log* file of selected run	*Section 14.6 [1]*
Open folder	Opens run directory of selected job	-
LS-OPT	Opens LS-OPT GUI if solver type is LS-OPT	-
LS-PREPOST	Opens selected run in LS-PREPOST (LS-DYNA only)	-
Kill	Kills selected job	*Appendix I.2 [1]*
Accelerated kill		*Appendix I.2 [1]*
Show plot	Show Time History plot	

The progress of the simulation jobs can be displayed for a selected stage or for all stages. If a job is selected from the list, the tools described in Table 6-1 are enabled.

When using LS-DYNA, the user can also view the progress (time history) of the analysis by selecting one of the available quantities from the *Plot* list (Time Step, Kinetic Energy, Internal Energy, etc.), Figure 6-2.

The **Progress** dialog allows a graphical indication of the job progress with the green horizontal bars linked to estimated completion time, Figure 6-2. This progress is only available for LS-DYNA jobs. The job monitoring is also visible when running remotely through a supported job distribution (queuing) system. The job status is automatically reported at a regular interval.

The text screen output while running both the batch and the graphical version as well as the integrated output window, Section 6.3.2, also report the status as follows:

```
JobID Status                    PID        Remaining
----- ------                    -----      ---------
1 N o r m a l termination!
2 Running                  8427  00:01:38 (91% complete)
3 Running                  8428  00:01:16 (93% complete)
4 Running                  8429  00:00:21 (97% complete)
5 Running                  8430  00:01:13 (93% complete)
6 Running                  8452  00:21:59 (0% complete)
7 Waiting ...
8 Waiting ...
```

In the batch version, the user may also type control-C to get the following response:

```
Jobs started
Got control C. Trying to pause scheduler
Enter the type of sense switch:
sw1: Terminate all running jobs
sw2: Get a current job status report for all jobs
t: Set the report interval
v: Toggle the reporting status level to verbose
stop: Suspend all jobs
cont: Continue all jobs
c: Continue the program without taking any action
Program will resume in 15 seconds if you do not enter a choice switch:
```

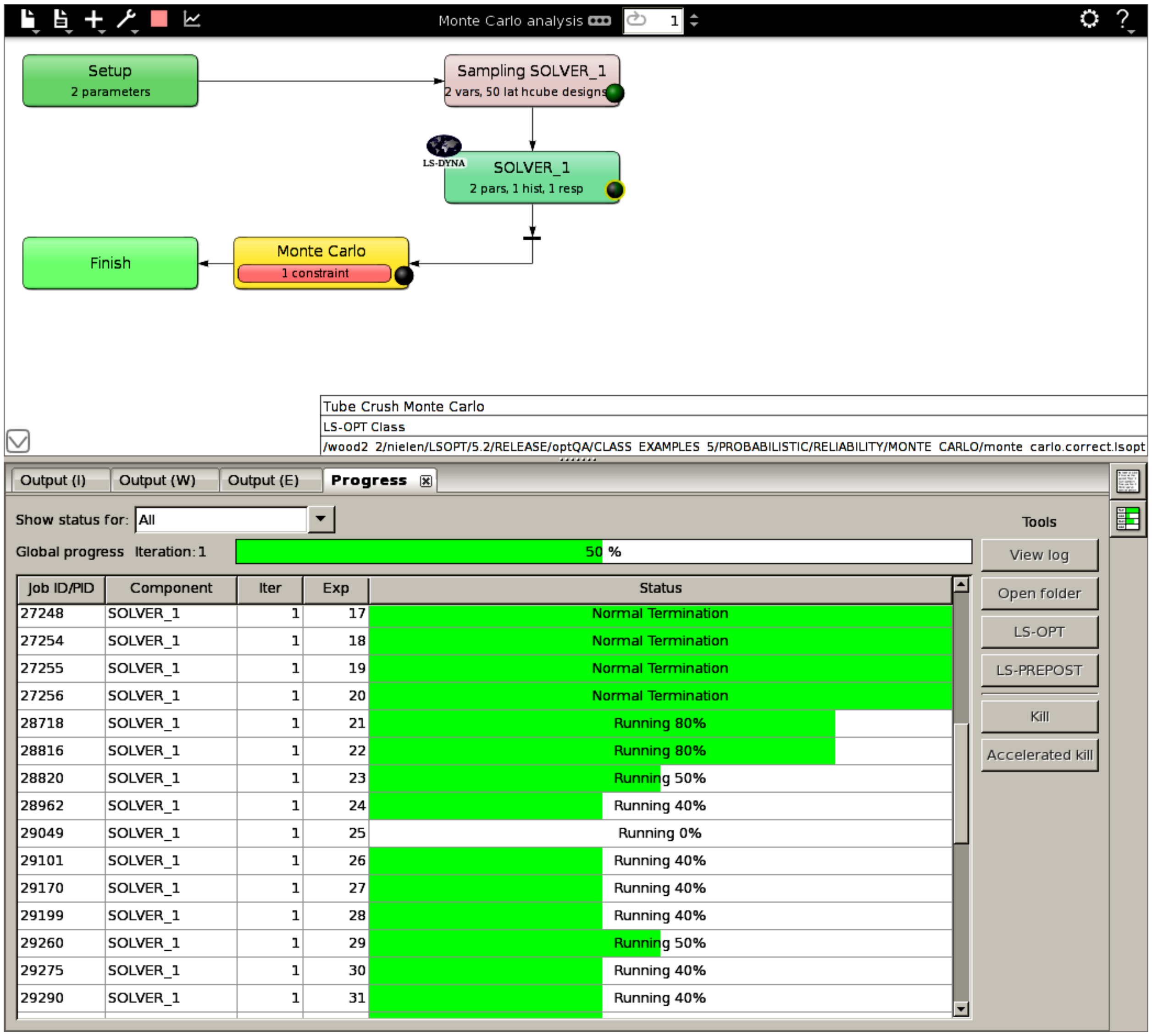
If `v` is selected, more detailed information of the jobs is provided, namely event time, time step, internal energy, ratio of total to internal energy, kinetic energy and total velocity.

6.3.1. Error termination of a solver run

The job scheduler will mark an error-terminated job to avoid termination of LS-OPT. For error-terminated solver jobs, the progress bars in the GUI are colored in red. Results of abnormally terminated jobs are ignored, hence they are not used in the optimization, e.g. to construct metamodels. If there are not enough results to continue, e.g. to construct the approximate design surfaces, LS-OPT will terminate with an appropriate error message.

6.3.2. Integrated output and display window

An integrated window which shows job progress (Figure 6-3) as well as output (comprehensive [I], warnings [W] and errors [E] — Figure 6-4) is also available. The window size can be adjusted or hidden using the ∨ above the top left corner of the progress window. Global progress is shown at the top. The tool functionality (except for *Show plot*) is the same as for the stage-based progress window shown in Figure 6-2 (see also Table 6-1).



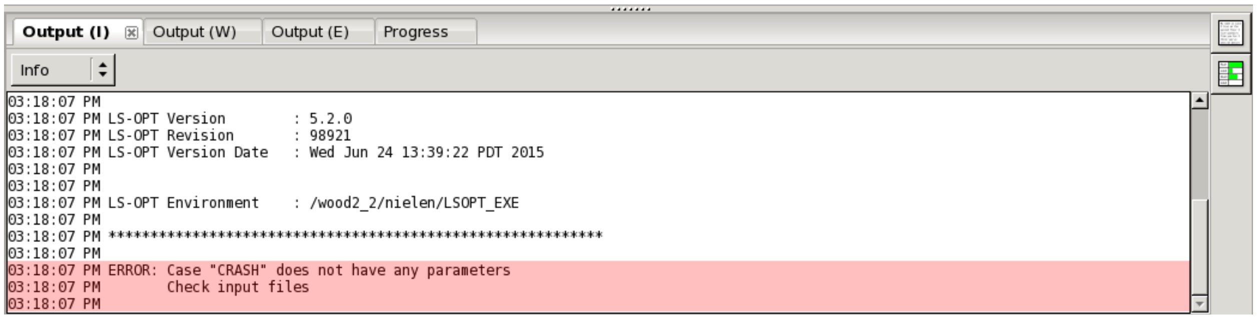
Tube Crush Monte Carlo
LS-OPT Class
/wood2 2/nielen/LSOPT/5.2/RELEASE/optQA/CLASS EXAMPLES 5/PROBABILISTIC/RELIABILITY/MONTE CARLO/monte carlo.correct.lsopt

Show status for: All

Global progress Iteration: 1 50 %

Job ID/PID	Component	Iter	Exp	Status
27248	SOLVER_1	1	17	Normal Termination
27254	SOLVER_1	1	18	Normal Termination
27255	SOLVER_1	1	19	Normal Termination
27256	SOLVER_1	1	20	Normal Termination
28718	SOLVER_1	1	21	Running 80%
28816	SOLVER_1	1	22	Running 80%
28820	SOLVER_1	1	23	Running 50%
28962	SOLVER_1	1	24	Running 40%
29049	SOLVER_1	1	25	Running 0%
29101	SOLVER_1	1	26	Running 40%
29170	SOLVER_1	1	27	Running 40%
29199	SOLVER_1	1	28	Running 40%
29260	SOLVER_1	1	29	Running 50%
29275	SOLVER_1	1	30	Running 40%
29290	SOLVER_1	1	31	Running 40%

Figure 6-3: Progress dialog



```
03:18:07 PM
03:18:07 PM LS-OPT Version        : 5.2.0
03:18:07 PM LS-OPT Revision       : 98921
03:18:07 PM LS-OPT Version Date   : Wed Jun 24 13:39:22 PDT 2015
03:18:07 PM
03:18:07 PM
03:18:07 PM LS-OPT Environment    : /wood2_2/nielen/LSOPT_EXE
03:18:07 PM
03:18:07 PM ***********************************************************
03:18:07 PM
03:18:07 PM ERROR: Case "CRASH" does not have any parameters
03:18:07 PM        Check input files
03:18:07 PM
```

Figure 6-4: LS-OPT output showing error diagnostic

6.4. Directory structure

When running an optimization, LS-OPT will generate a directory in the work directory for each sampling and for each stage using the sampling or stage name, respectively. If a sampling and a stage have the same name, the same directory will be used.

In the stage directories a subdirectory will be created for each simulation.

These sub-directories are named mmm.nnnn, where mmm represents the iteration number and nnnn is a number starting from `1`.

The work directory needs to contain at least the .lsopt file.

An example of a subdirectory name, defined by LS-OPT, is `side_impact/3.11`, where `3.11` represents the design point number 11 of iteration 3. The creation of subdirectories is automatic and the user only needs to deal with the working directory.

In the case of simulation runs being conducted on remote nodes, a replica of the run directory is automatically created on the remote machine. The `response.`*`n`* and `history.`*`n`* files will automatically be transferred back to the local run directory at the end of the simulation run. These are the only files required by LS-OPT for further processing. More files can be transferred back by using the recover files options, see *Section 5.4.5: Recovering Output Files [1]*.

Multilevel optimization. Since this manual is focused on multilevel optimization, some of the stages are of type LSOPT (Section **3.3.3**). In this case, the sub-directories mmm.nnnn act as the working directories for inner level LS-OPT processes. As a result, these directories have further sublevel directories. In Figure 6-6, the directory structure is shown for multilevel optimization with a single *LSOPT* stage named *'Stage 3'*.

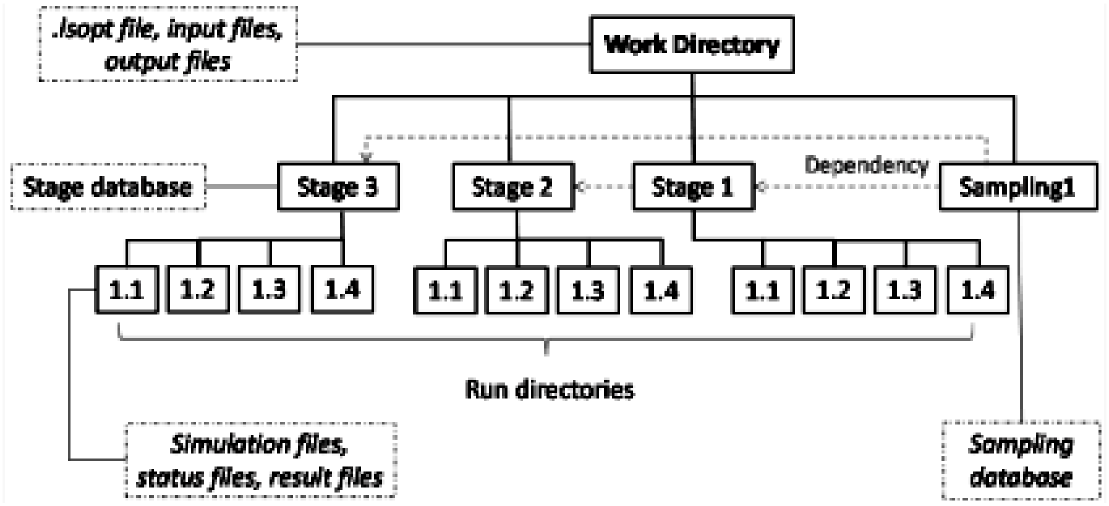


Figure 6-5 : Directory structure in LS-OPT

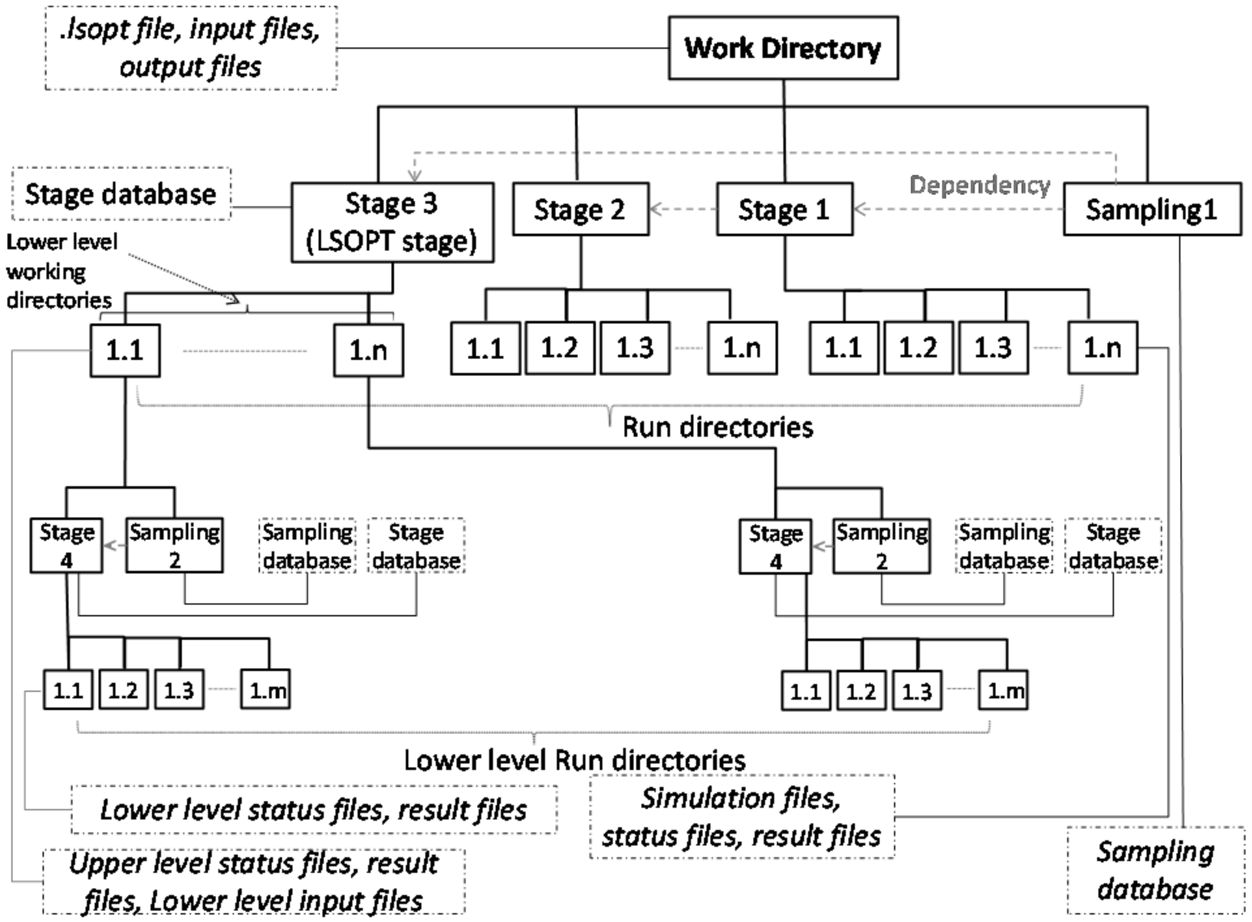


Figure 6-6 : Directory structure for multilevel optimization with one LSOPT stage.

7. Applications of Optimization

This chapter provides a brief description of some of the applications of optimization that can be performed using LS-OPT. It should be read in conjunction with the examples chapters, where the applications are illustrated with practical examples. The applications presented here are that of material parameter identification and multilevel optimization, both of which are important components of the ICME model. Therefore, before understanding the overall ICME setup (Chapter 10), it is important to understand these constituting features. Two simple examples of the multilevel optimization feature of LS-OPT, which was developed as part of this project in order to facilitate the ICME model setup, are also presented in Chapter 9.

7.1. Parameter Identification

Parameter identification problems are non-linear inverse problems which can be solved using mathematical optimization. System parameter identification is a commonly used feature of LS-OPT, especially for the purpose of calibrating material models.

The procedure consists of minimizing the mismatch between target values and corresponding solver output values, or between two curves. In the latter case, the two curves typically consist of a two-dimensional experimental target curve and a computed curve. The computed curve is a variable response, being dependent on the system parameters, e.g. material constants. It can also be a *crossplot*, constructed by combining two time histories such as *strain* and *stress (Section 6.4.2: Crossplot History [1]).*

The two main essential components of an algorithm designed for system identification are

- o optimization algorithm and
- o curve matching metric.

7.1.1. Optimization algorithm

The recommended optimization algorithm to be used to solve a parameter identification problem is the *Metamodel-based Optimization* with the strategy discussed in *Section 4.7.3: Sequential Strategy with Domain Reduction [1]*. Use linear polynomial metamodels and *D-optimal* point selection which is the default for the selected task and strategy, *Section 9.3.2: D-Optimal point selection [1]*.

7.1.2. Curve matching metric

To calculate the mismatch between the target and the computed curve, define a *Curve Matching* composite, *Section 10.5: Curve Matching Composite [1]*. There are two curve matching metrics available, *Mean Square Error* and *Curve Mapping*. *Mean Square Error* is an ordinate-based curve matching metric. Hence if the curve has steep parts or if the ordinate values are not unique, (the curve is a hysteretic curve), *Curve Mapping* is the metric of choice.

Because *Curve Mapping* uses the length of the curve to calculate the mismatch, filtering of the component history curves is recommended.

7.2. Multilevel Optimization

In multilevel optimization, the optimization problem is solved in parts at two (usually) or more levels. Each sublevel optimizes a subset of the variable set while maintaining constant values for the variables belonging to preceding levels. Multilevel optimization can be used to group variables into the different levels to make the problem easier to solve. For example, a gradient based optimizer may be used for some of the variables while a zero order method is used for the others. Similarly, metamodels may be constructed with some of the variables while the rest are optimized using a direct method. In LS-OPT, this is performed using the LS-OPT stage and by specifying some of the inner level variables as *Transfer Variables* (Section 3.3.3).

The multilevel optimization process in LS-OPT can be briefly summarized as follows. For the sake of simplicity, the summary is provided for the case consisting of two levels.

1. *Input File preparation for LS-OPT stage of outer level setup*: The input file for the LS-OPT stage is an `.lsopt` file itself. Therefore, preparing this file involves exactly the same steps as any single level problem setup. While this file is an input file for the outer level, it is also the LS-OPT setup file for solving the inner level problem. As already mentioned, the inner level optimization is performed with respect to a subset of the variables while the rest are optimized in the outer level. Therefore, these other parameters are constants for the inner level. The LS-OPT GUI is used to prepare the .lsopt file; the inner level free variables are set as Continuous or Discrete Variables, but the rest are set as Transfer Variables and are treated as constants at this level.
2. *Stage setup for outer level*: See Section 3.3.3.
3. *Response definitions for outer level*: See Section 5.3.
4. *Global Setup for outer level*: Once a `.lsopt` file parameterized with *Transfer Variables* is specified as the LS-OPT stage input file in the outer level, the outer level LS-OPT stage automatically detects these parameters and they are added to the *Global Setup* as constants. These can then be set as Continuous or Discrete Variables by the user and thus, they become outer level variables (Figure 4-3, Section 4.1.4).
5. *Running the optimization*: The outer level optimization is started by pressing the run button in the GUI or from command line, which leads to the creation of a design of

experiments for the outer level variables. A run directory is created for each outer level sample. The LS-OPT stage input file (i.e. the inner level .lsopt setup) is copied to each of these directories and named as `LsoOpt.inp` by default. The Transfer Variable values in a particular run directory are set as the corresponding outer level sample's variable values. Once the Transfer Variable values are set, they are treated as constants within a run directory and the inner optimization is carried out with respect to the free inner level variables. The optimized inner level entities are then extracted as sample responses at the outer level, thus providing the response values at each outer level sample. The outer level optimization is then carried out with respect to the remaining variables.

8. Material models (LS-DYNA$^{®}$)

8.1. Introduction

The ICME model consists of two material models, CP and SV, with different length scales. This chapter presents the LS-DYNA material card definitions for both these models. Both the models are implemented as user-defined material models. The CP model definition is further divided into BCC and FCC depending on the lattice structure.

8.2. Crystal Plasticity

8.2.1. BCC Crystal Plasticity model

***MAT_USER_DEFINED_MATERIAL_MODELS**

This is the user-defined material type 46 for the BCC crystal plasticity model.

Card 1	1	2	3	4	5	6	7	8
Variable	MID	RO	46	24				
Type	A8	F	I	I				

Card 2	1	2	3	4	5	6	7	8
Variable								
Type								

Card 3	1	2	3	4	5	6	7	8
Variable	C11	C12	C44	TAU0A	TAU0B	TAU0C	NGR	NSS
Type	F	F	F	F	F	F	F	F

Card 4	1	2	3	4	5	6	7	8
Variable	TAUSA	TAUSB	TAUSC	H0A	H0B	H0C	Q	R
Type	F	F	F	F	F	F	F	F

Card 4	1	2	3	4	5	6	7	8
Variable	AA	AB	AC	QTYP	PSI	THETA	PHI	
Type	F	F	F	F	F	F	F	

VARIABLE	DESCRIPTION
MID	Material identification. A unique number or label not exceeding 8 characters must be specified.
RO	Mass density.
C11	The 1,1 term in the 6×6 constitutive matrix. Here 1 corresponds to the *a* material direction
C12	The 1,2 term in the 6×6 constitutive matrix. Here 2 corresponds to the *b* material direction
C44	The 4,4 term in the 6×6 constitutive matrix
TAU0A	Initial slip resistance for slip systems 1–12
TAU0B	Initial slip resistance for slip systems 13–24

VARIABLE	DESCRIPTION
TAU0C	Initial slip resistance for slip systems 25–48
NGR	Number of grains
NSS	Number of slip systems
TAUSA	Saturation value of slip resistance for slip systems 1–12
TAUSB	Saturation value of slip resistance for slip systems 13–24
TAUSC	Saturation value of slip resistance for slip systems 25–48
H0A	Initial hardening rate for slip systems 1–12
H0B	Initial hardening rate for slip systems 13–24
H0C	Initial hardening rate for slip systems 25–48
Q	Latent hardening ratio
R	Yield function coefficient
AA	Exponent in hardening rate evolutionary function for slip systems 1–12
AB	Exponent in hardening rate evolutionary function for slip systems 13–24
AC	Exponent in hardening rate evolutionary function for slip systems 25–48
QTYP	Euler angle type: EQ.1: Bunge EQ.2: Roe EQ.3: Kocks LT.0: Use random orientation with Euler angle type |QTYP|
PSI	First Euler angle (not used if QTYP<0)
THETA	Second Euler angle (not used if QTYP<0)
PHI	Third Euler angle (not used if QTYP<0)

Remarks:

This material implements the combined-constraints crystal plasticity model of Zamiri and Pourboghrat [1] for BCC crystals. It is implemented for solid elements and for explicit analysis only.

8.2.2. FCC Crystal Plasticity model

***MAT_USER_DEFINED_MATERIAL_MODELS**

This is the user-defined material type 45 for the FCC crystal plasticity model.

Card 1	1	2	3	4	5	6	7	8
Variable	MID	RO	45	16				
Type	A8	F	I	I				

Card 2	1	2	3	4	5	6	7	8
Variable								
Type								

Card 3	1	2	3	4	5	6	7	8
Variable	C11	C12	C44	TAU0	TAUS	H0	A	Q
Type	F	F	F	F	F	F	F	F

Card 4	1	2	3	4	5	6	7	8
Variable	NGR	QTYP	PSI	THETA	PHI	R		
Type	F	F	F	F	F	F		

VARIABLE	DESCRIPTION
MID	Material identification. A unique number or label not exceeding 8 characters must be specified.
RO	Mass density.
C11	The 1,1 term in the 6×6 constitutive matrix. Here 1 corresponds to the *a* material direction
C12	The 1,2 term in the 6×6 constitutive matrix. Here 2 corresponds to the *b* material direction
C44	The 4,4 term in the 6×6 constitutive matrix
TAU0	Initial slip resistance
TAUS	Saturation value of slip resistance
H0	Initial hardening rate
A	Exponent describing shape of hardening rate evolutionary function
Q	Latent hardening ratio
NGR	Number of grains
QTYP	Euler angle type: EQ.1: Bunge EQ.2: Roe EQ.3: Kocks LT.0: Use random orientation with Euler angle type |QTYP|
PSI	First Euler angle (not used if QTYP<0)

VARIABLE	DESCRIPTION
THETA	Second Euler angle (not used if QTYP<0)
PHI	Third Euler angle (not used if QTYP<0)
R	Yield function coefficient

Remarks:

This material implements the combined-constraints crystal plasticity model of Zamiri and Pourboghrat [1] for FCC crystals. It is implemented for solid elements and for explicit analysis only.

8.3. State Variable model

8.3.1. QP980 and 3Mn steel

***MAT_USER_DEFINED_MATERIAL_MODELS**

This is the user-defined material type 50 for the PNNL State Variable model for QP980 and 3Mn steel.

Card 1	1	2	3	4	5	6	7	8
Variable	MID	RO	50	24				
Type	A8	F	I	I				

Card 2	1	2	3	4	5	6	7	8
Variable								
Type								

Card 3	1	2	3	4	5	6	7	8
Variable	EF	PRF	YLDF	EM	PRM	YLDM	EA	PRA
Type	F	F	F	F	F	F	F	F

Card 4	1	2	3	4	5	6	7	8
Variable	YLDA	EN	PRN	YLDM	A	B	N	VOLF
Type	F	F	F	F	F	F	F	F

Card 4	1	2	3	4	5	6	7	8
Variable	VOLM	VOLA0						
Type	F	F						

VARIABLE	DESCRIPTION
MID	Material identification. A unique number or label not exceeding 8 characters must be specified.
RO	Mass density.
EF	Youngs modulus of ferrite
PRF	Poisson ratio of ferrite
YLDF	Load curve for yield strength vs effective plastic strain for ferrite
EM	Youngs modulus of martensite
PRM	Poisson ratio of martensite

VARIABLE	DESCRIPTION
YLDM	Load curve for yield strength vs effective plastic strain for martensite
EA	Youngs modulus of austenite
PRA	Poisson ratio of austenite
YLDA	Load curve for yield strength vs effective plastic strain for austenite
EN	Youngs modulus of new martensite
PRN	Poisson ratio of new martensite
YLDN	Load curve for yield strength vs effective plastic strain for new martensite
A	Parameter in Olson-Cohen model for phase transformation kinetics. GT.0: Load curve ID specifying A as function of triaxiality LT.0: Constant value |A|
B	Parameter in Olson-Cohen model for phase transformation kinetics. GT.0: Load curve ID specifying A as function of triaxiality LT.0: Constant value |B|
N	Exponent in Olson-Cohen model for phase transformation kinetics. GT.0: Load curve ID specifying A as function of triaxiality LT.0: Constant value |N|
VOLF	Volume fraction of ferrite
VOLM	Volume fraction of martensite
VOLA0	Initial volume fraction of austenite

Remarks:

The PNNL State Variable model for QP980 and 3Mn steel considers three phases in the material initially: ferrite, martensite and austenite. It then uses the Olson-Cohen model [2] of phase transformation kinetics to incrementally calculate the transformation of austenite to new martensite via shear band interactions, as follows:

$$\dot{f}_{sb} = a(1 - f_{sb})\dot{\varepsilon}_p$$

$$A = abn(f_{sb})^{n-1}(1 - f_{sb})$$

$$\dot{f}_{nm} = A(1 - f_{nm})\dot{\varepsilon}_p$$

$$V_{nm} = V_{a0}f_{nm}$$

where

ε_p: equivalent plastic strain

f_{sb}: normalized shear band volume fraction $(0 \leq f_{sb} \leq 1)$

f_{nm}: normalized new martensite volume fraction $(0 \leq f_{nm} \leq 1)$

V_{a0}: initial austenite volume fraction

V_{nm}: new martensite volume fraction

A simple homogenization model is used to calculate the properties of the material from its constituent phases.

This material model can be used for both QP980 and 3Mn steel with appropriate choices of the material parameters.

The model has been implemented for solid, shell and thick shell elements, and can be used with either explicit or implicit analysis. Additionally, it can also be used with one-step forming (see *CONTROL_FORMING_ONESTEP)

This material has the following history variables that are meaningful to the user:

1. New martensite volume fraction
2. Austenite volume fraction
3. Triaxiality

8.3.2. 10Mn steel

***MAT_USER_DEFINED_MATERIAL_MODELS**

This is the user-defined material type 47 for the PNNL State Variable model for 10Mn steel.

Card 1	1	2	3	4	5	6	7	8
Variable	MID	RO	47	24				
Type	A8	F	I	I				

Card 2	1	2	3	4	5	6	7	8
Variable								
Type								

Card 3	1	2	3	4	5	6	7	8
Variable	EM	PRM	YLDM	EA	PRA	YLDA	EE	PRE
Type	F	F	F	F	F	F	F	F

Card 4	1	2	3	4	5	6	7	8
Variable	YLDE	EL	PRL	YLDL	A	B	N	VOLM
Type	F	F	F	F	F	F	F	F

Card 4	1	2	3	4	5	6	7	8
Variable	VOLA0							
Type	F							

VARIABLE	DESCRIPTION
MID	Material identification. A unique number or label not exceeding 8 characters must be specified.
RO	Mass density.
EM	Youngs modulus of martensite
PRM	Poisson ratio of martensite
YLDM	Load curve for yield strength vs effective plastic strain for martensite
EA	Youngs modulus of austenite
PRA	Poisson ratio of austenite
YLDA	Load curve for yield strength vs effective plastic strain for austenite
EE	Youngs modulus of ε -martensite
PRE	Poisson ratio of ε -martensite
YLDE	Load curve for yield strength vs effective plastic strain for ε -martensite
EL	Youngs modulus of α-martensite
PRL	Poisson ratio of α-martensite
YLDL	Load curve for yield strength vs effective plastic strain for α-martensite
A	Parameter in Olson-Cohen model for phase transformation kinetics. GT.0: Load curve ID specifying A as function of triaxiality LT.0: Constant value |A|
B	Parameter in Olson-Cohen model for phase transformation kinetics. GT.0: Load curve ID specifying A as function of triaxiality LT.0: Constant value |B|
N	Exponent in Olson-Cohen model for phase transformation kinetics. GT.0: Load curve ID specifying A as function of triaxiality LT.0: Constant value |N|
VOLM	Volume fraction of martensite

VOLA0	Initial volume fraction of austenite

Remarks and formulation

The PNNL State Variable model for 10Mn steel considers two phases in the material initially: martensite and austenite. It then uses a variation of the Olson-Cohen model [1] of phase transformation kinetics to incrementally calculate the transformation of austenite first into ε-martensite, and then to α-martensite, via shear band interactions, as follows:

$$\dot{f}_{sb} = a(1 - f_{sb})\dot{\varepsilon}_p$$

$$A = abn(f_{sb})^{n-1}(1 - f_{sb})$$

$$\dot{f}_{\alpha m} = A(1 - f_{\varepsilon m})\dot{\varepsilon}_p$$

$$V_{\alpha m} = V_{a0} f_{\alpha m}$$

$$V_{sb} = V_{a0} f_{sb}$$

$$V_a = V_{a0} - V_{sb}; \qquad V_{\varepsilon m} = V_{sb} - V_{\alpha m}$$

where

ε_p: equivalent plastic strain

f_{sb}: normalized shear band volume fraction $(0 \leq f_{sb} \leq 1)$

$f_{\alpha m}$: normalized α-martensite volume fraction $(0 \leq f_{\alpha m} \leq f_{sb})$

V_{a0}: initial austenite volume fraction

$V_{\alpha m}$: α-martensite volume fraction

$V_{\varepsilon m}$: ε-martensite volume fraction

V_{α}: austenite volume fraction

V_{sb}: shear band volume fraction

A simple homogenization model is used to calculate the properties of the material from its constituent phases.

The model has been implemented for solid, shell and thick shell elements, and can be used with either explicit or implicit analysis. Additionally, it can also be used with one-step forming (see *CONTROL_FORMING_ONESTEP)

This material has the following history variables that are meaningful to the user:

1. α-martensite volume fraction
2. ε-martensite volume fraction
3. Austenite volume fraction
4. Triaxiality

8.4. References

[1] Zamiri, A. R. and Pourboghrat, F., “A novel yield function for single crystals based on combined constraints optimization”, *International Journal of Plasticity,* 26:731–746, 2010.
[2] Olson, G. B. and Cohen, M., “Kinetics of strain-induced martensitic nucleation”, *Metallurgical Transactions A*, 6A:791–795, 1975.

9. Examples – Simple Multi-level Optimization

9.1. Multilevel Optimization using both Direct method and Metamodel

This example uses the same finite element model, but the optimization problem is modified to include two more variables. These variables are the material Young’s modulus *YM* and the yield stress *SIGY*. The optimization problem is given in Equation 16-2. However, the optimization is solved in two levels – the outer level optimizes `SIGY` and `YM` using a single iteration metamodel-based method (Equation 8-3) and the inner level optimizes the thickness values `thood and tbumper` using direct GA (Equation 8-4).

$$\boxed{\begin{gathered}\min_{thood,tbumper,SIGY,YM}[HIC(15ms)] \\ subject\ to \\ Intrusion\ (50ms)\ <\ 550mm\end{gathered}} \qquad (8\text{-}1)$$

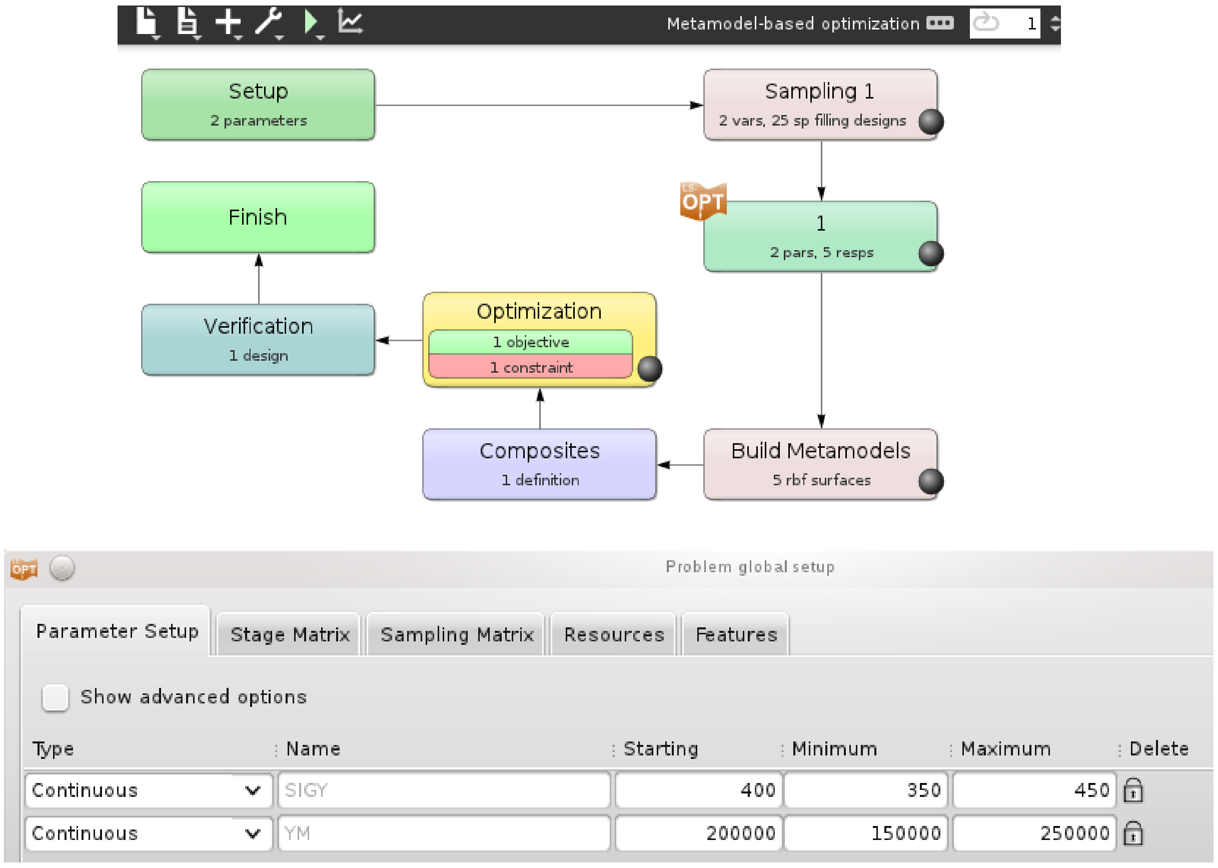
The outer level optimization problem is:

$$\boxed{\begin{gathered}\min_{SIGY,YM}\ (HIC_{thood_{opt},tbumper_{opt}}15\text{ms}) \\ subject\ to \\ Intrusion_{thood_{opt},tbumper_{opt}}(50ms)\ <\ 550mm\end{gathered}} \qquad (8\text{-}2)$$

where $HIC_{thood_{opt},tbumper_{opt}}$ and $Intrusion_{thood_{opt},tbumper_{opt}}$ are the HIC and intrusion values obtained as the results of the inner level optimization problem with respect to variables `thood and tbumper` given by Equation 8-3. $HIC_{thood_{opt},tbumper_{opt}}$, $Intrusion_{thood_{opt},tbumper_{opt}}$ are obtained for every outer level sample (`SIGY-YM` pair) by running an inner level optimization for each sample. The inner level optimization problem for the j^{th} outer level sample is:

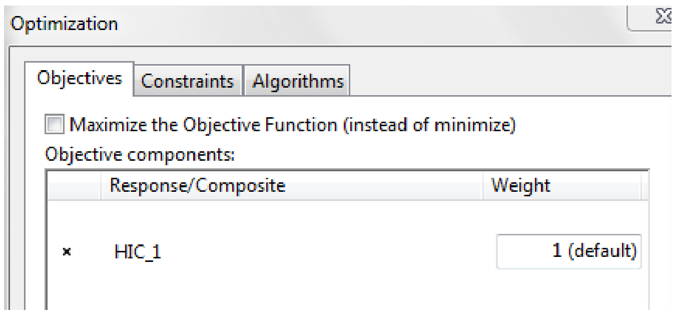
$$\boxed{\begin{gathered}\min_{thood,tbumper} HIC(thood, tbumper, YM_j\,, SIGY_j)(15ms)\\ subject\ to\\ Intrusion\ (thood, tbumper|YM_j, SIGY_j)\ (50ms)\ <\ 550mm\end{gathered}} \qquad (8\text{-}3)$$

The LS-OPT GUI for outer level problem setup is shown in Figure 9-1. The optimization problem setup is shown in Figure 9-2; HIC_1 and $Intru$ are optimized responses calculated in the inner level.



Type	Name	Starting	Minimum	Maximum	Delete
Continuous	SIGY	400	350	450	
Continuous	YM	200000	150000	250000	

Figure 9-1: Multilevel Optimization outer level setup

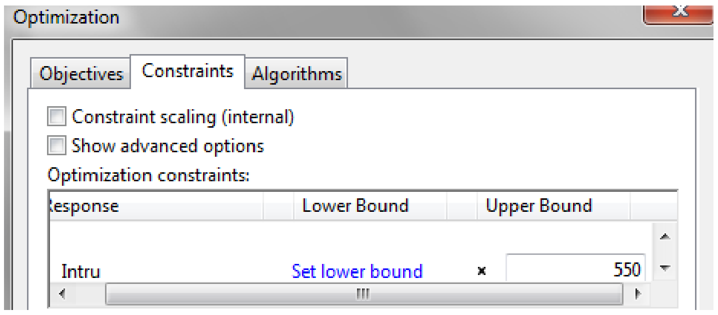

Optimization

Objectives | Constraints | Algorithms

☐ Maximize the Objective Function (instead of minimize)

Objective components:

	Response/Composite	Weight
×	HIC_1	1 (default)


Optimization

Objectives | Constraints | Algorithms

☐ Constraint scaling (internal)

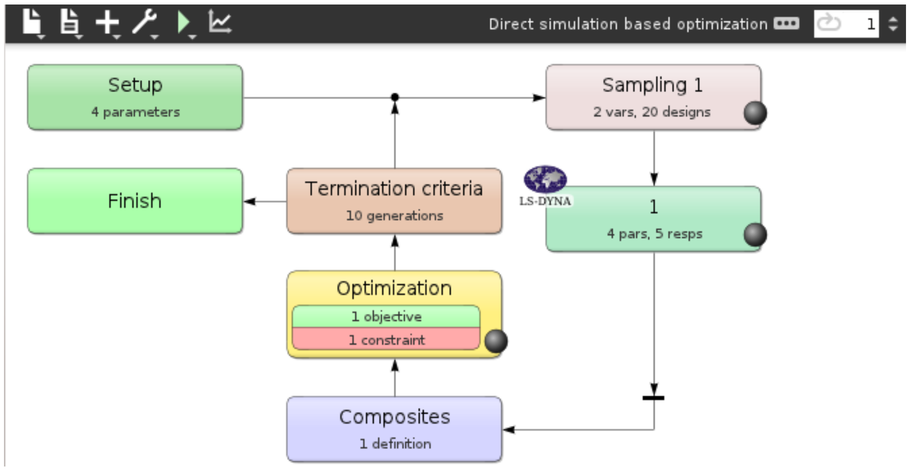
☐ Show advanced options

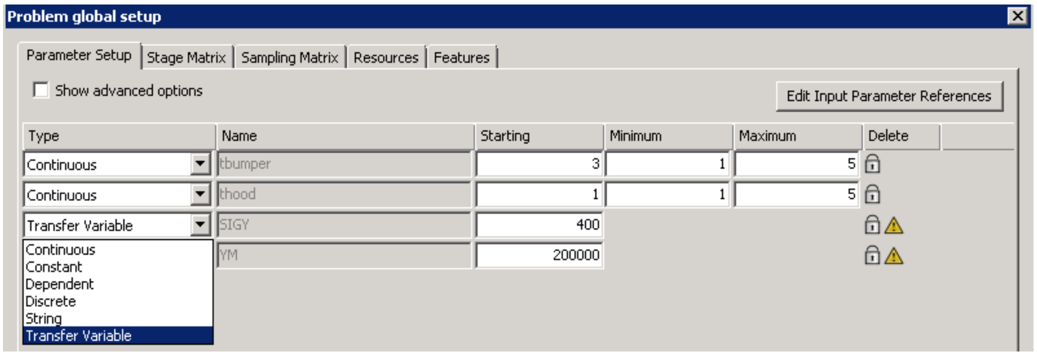
Optimization constraints:

esponse	Lower Bound		Upper Bound
Intru	Set lower bound	×	550

Figure 9-2: Multilevel Optimization outer level optimization problem

The LS-OPT GUI for inner level problem setup is shown in Figure 9-3. The optimization problem setup is shown in Figure 9-4. It should be noted that the outer level variables are *Transfer Variables* in the inner level and are treated as constants for the optimization.




Problem global setup

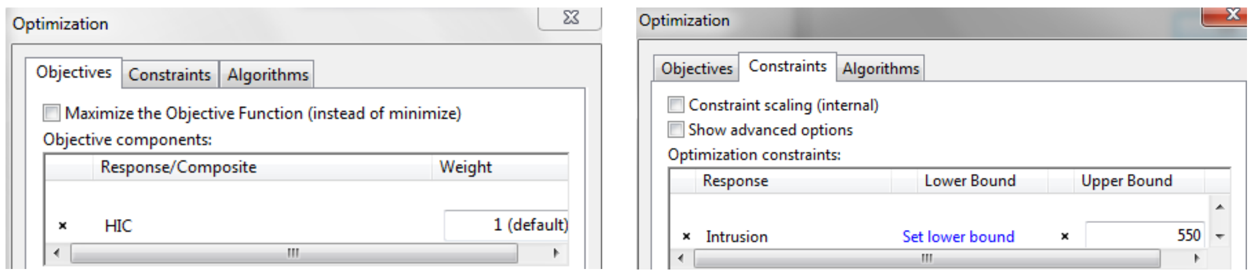
Parameter Setup | Stage Matrix | Sampling Matrix | Resources | Features

☐ Show advanced options

Edit Input Parameter References

Type	Name	Starting	Minimum	Maximum	Delete
Continuous	tbumper	3	1	5	
Continuous	thood	1	1	5	
Transfer Variable	SIGY	400			
	YM	200000			

Figure 9-3: Multilevel Optimization inner level setup


Optimization

Objectives | Constraints | Algorithms

☐ Maximize the Objective Function (instead of minimize)

Objective components:

	Response/Composite	Weight
×	HIC	1 (default)

Optimization

Objectives | Constraints | Algorithms

☐ Constraint scaling (internal)

☐ Show advanced options

Optimization constraints:

	Response	Lower Bound		Upper Bound
×	Intrusion	Set lower bound	×	550

Figure 9-4: Multilevel Optimization inner level optimization problem

9.1.1. Results

The optimum solution is obtained at `SIGY` = 412, `YM` = 2.5 × 10^5, `tbumper` = 4.85, `thood` = 1.57. The corresponding HIC value is 105 and there is no constraint violation at the solution.

The metamodel for HIC, with respect to outer level variables `YM` and `SIGY,` is shown in Figure 9-5. The optimum is also plotted on the figure (purple cube). The inner level optimization history is depicted in Figure 9-6 for the outer level sample 2.1 (i.e. the sample with optimized `YM` and `SIGY`).

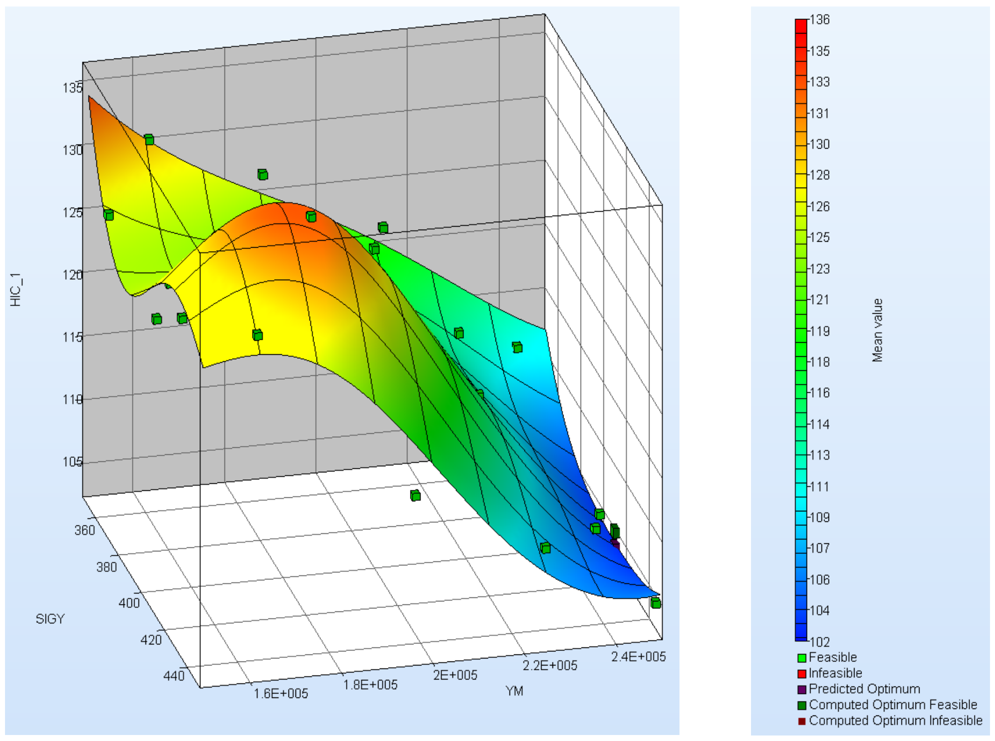


Figure 9-5: Multilevel Optimization. Metamodel for objective function (HIC)

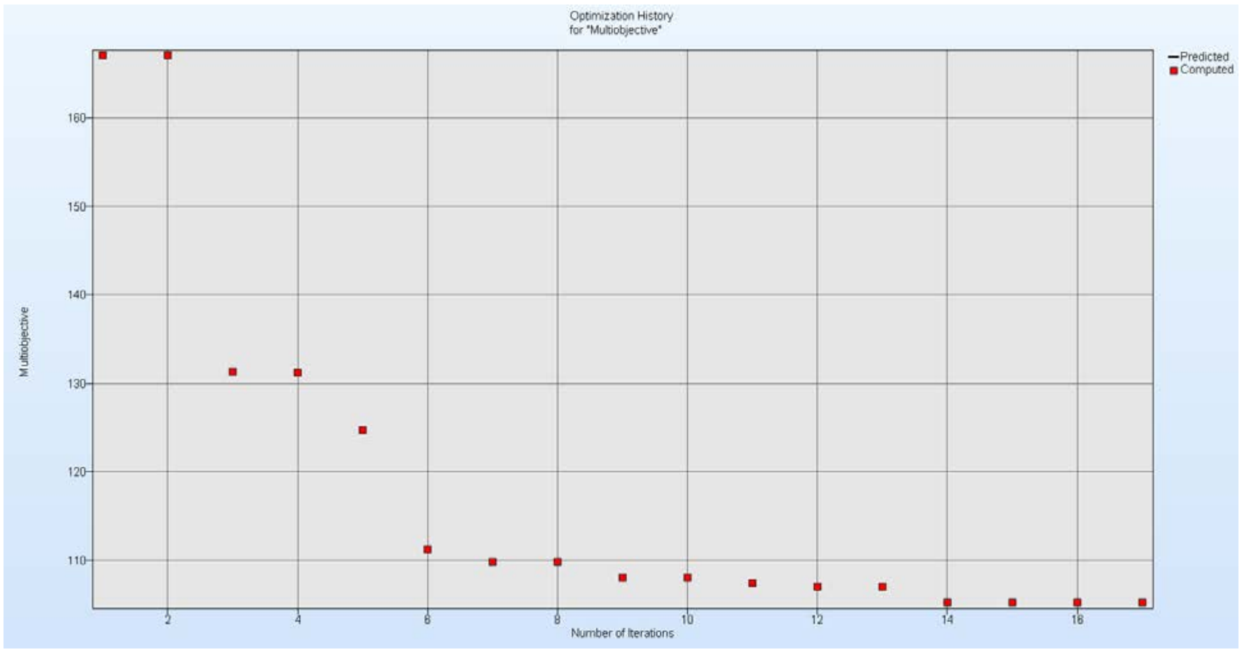


Figure 9-6: Inner level optimization history for the last (optimal) outer level sample.

9.2. Multilevel Optimization using continuous and string variables

Multilevel optimization can be used to optimize different sets of variables using different methods. For example, direct optimization is often preferred for string or categorical variables while metamodel-based methods are often used for other variables. In this example, two of the variables are continuous while two other variables are strings. The continuous variables represent component thicknesses `thood` and `tbumper` and the string variables `mat_b` and `mat_hood` are the names of include files with different material properties. Two string constants `m1` and `material3` are also used in the example. Different methods of parameterizing string variables and constants (native LS-DYNA parameterization and user-defined) are demonstrated through this example.

The optimization problem is given in Equation (8-4). However, the optimization is solved in two levels – the outer level optimizes `thood and tbumper` using a domain reduction metamodel-based method (Equation 8-5) and the inner level optimizes the thickness values `mat_hood and mat_b` using direct GA (Equation 8-6).

$$\boxed{\begin{gathered}\min_{thood,tbumper,mat\_b,mat\_hood} Mass \\ subject\ to \\ Intrusion\ (50ms)\ <\ 550mm\end{gathered}} \qquad (8\text{-}4)$$

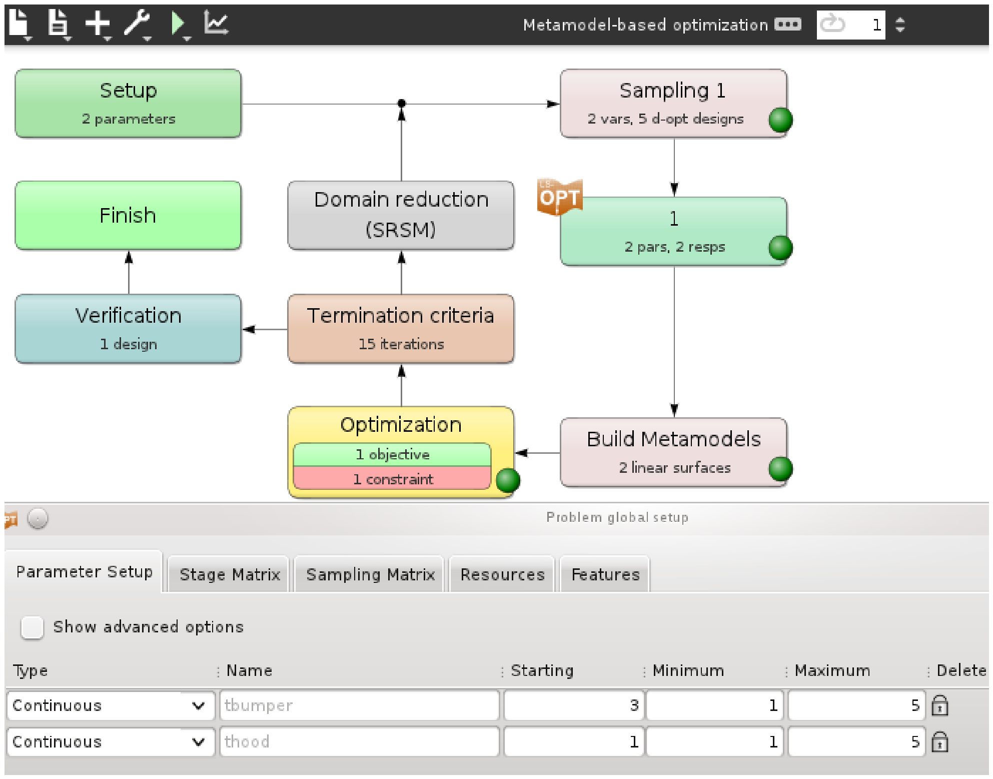
The outer level optimization problem is:

$$\boxed{\begin{gathered}\min_{tbumper,thood} \left(Mass_{mat\_b_{opt},mat\_hood_{opt}}\right) \\ subject\ to \\ Intrusion_{mat\_b_{opt},mat\_hood_{opt}}(50ms) < 550mm\end{gathered}} \qquad (8\text{-}5)$$

where $Mass_{mat\_b_{opt},mat\_hood_{opt}}$ and $Intrusion_{mat\_b_{opt},mat\_hood_{opt}}$ are the mass and intrusion values obtained as the results of the inner level optimization problem (Equation 8-6) with respect to variables `mat_hood` and `mat_b`. $Mass_{mat\_b_{opt},mat\_hood_{opt}}$ and $Intrusion_{mat\_b_{opt},mat\_hood_{opt}}$ are obtained for every outer level sample (`tbumper-thood` pair) by running an inner level optimization for each sample. The inner level optimization problem for the j^{th} outer level sample is:

$$\boxed{\begin{gathered}\min_{mat\_b,mat\_hood} Mass(mat\_b, mat\_hood|tbumper_j, thood_j) \\ subject\ to \\ Intrusion\ (mat\_b, mat\_hood|tbumper_j, thood_j) < 550mm\end{gathered}} \qquad (8\text{-}6)$$

The outer level LS-OPT setup consists of an LS-OPT stage parameterized using two transfer variables. These variables, `tbumper` and `thood`, are constinuous variable in the outer level (Figure 9-7).



Type	Name	Starting	Minimum	Maximum	Delete
Continuous	tbumper	3	1	5	
Continuous	thood	1	1	5	

Figure 9-7: Outer level optimization setup

The inner level consists of two string variables and two string constants, in addition to the two transfer variables whose values are passed down from the outer level. The LS-DYNA input deck is parameterized as follows. `tbumper`, `thood`, `m1` and `mat_b` are parameterized using the *PARAMETER card. The string parameters are indicated using "c" before the variable names.

```
*PARAMETER
rtbumper,3.0,rthood,1.0,cm1,mat1,cmat_b,mat_b_o
```

Two other string parameters are defined using the user-defined format. The parameter thood appears at two places in the LS-DYNA deck:

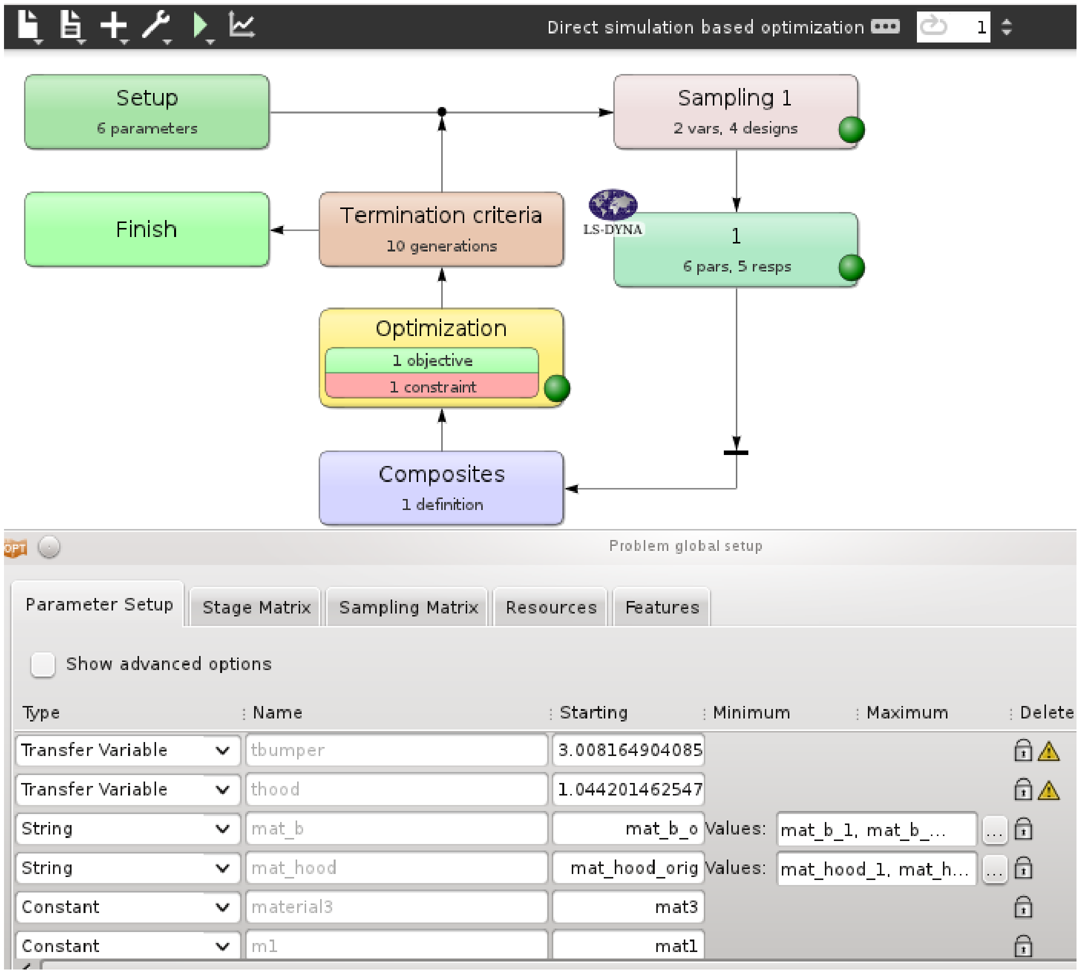
```
*include
<<mat_hood:0>>
*include
<<mat_hood:30>>
```

`<<:0>>` indicates that the entire replacement string will be printed without any additional spaces. `<<:30>>` indicates that if the length of the replacement string for `mat_hood` is longer than 30 then it will be truncated. Also, if the replacement string for `mat_hood` is shorter than 30 then it would be padded with spaces while printing.

The parameter `material3` is defined without a colon and has the same meaning as `<<:0>>`.

```
*include
<<material3>>
```

The inner level LS-OPT GUI setup is shown in (Figure 9-8).

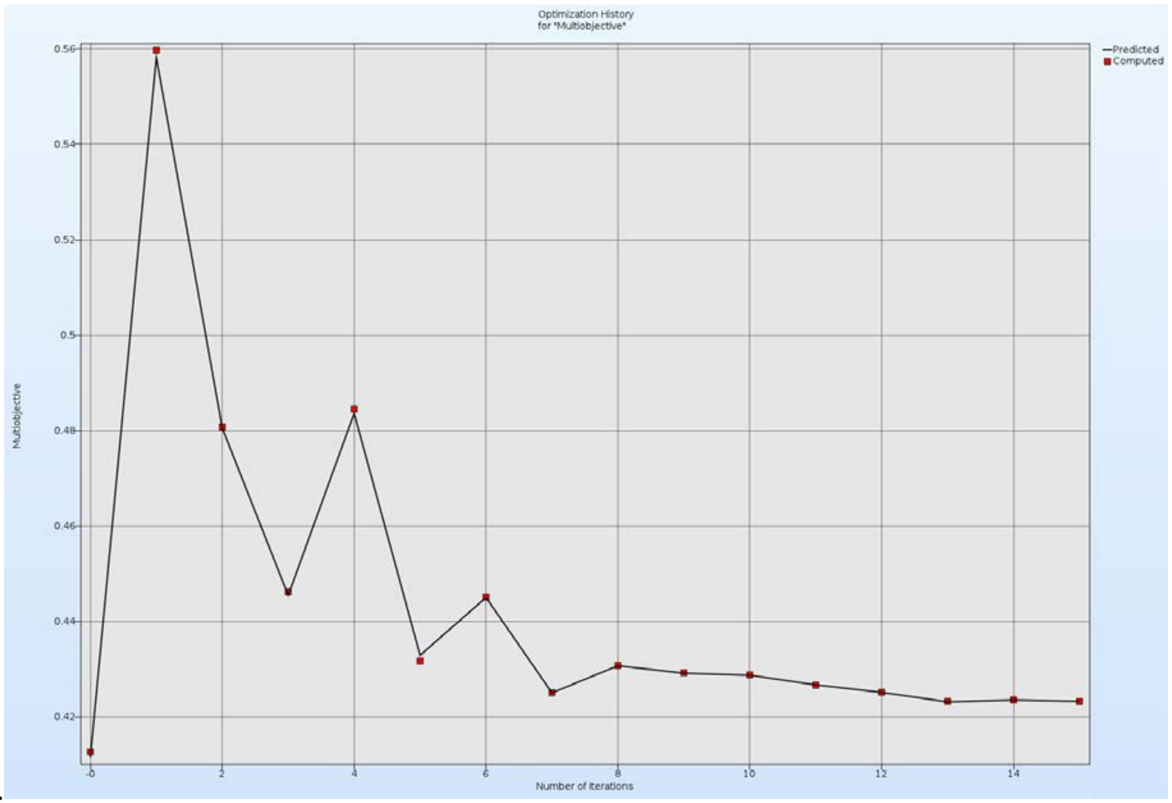


Type	Name	Starting	Minimum	Maximum	Delete
Transfer Variable	tbumper	3.008164904085			
Transfer Variable	thood	1.044201462547			
String	mat_b	mat_b_o	Values:	mat_b_1, mat_b_...	
String	mat_hood	mat_hood_orig	Values:	mat_hood_1, mat_h...	
Constant	material3	mat3			
Constant	m1	mat1			

Figure 9-8: Inner level optimization setup with string and transfer variables

9.2.1. Results

The optimum solution is obtained at `tbumper` = 3.01, `thood` = 1.04 `mat_b` = “mat_b_3”, `mat_hood` = “mat_hood_3”. The corresponding Mass value is 0.42 and there is no constraint violation at the solution. The outer level optimization history for the SRSM method is depicted in Figure 9-9.



.

Figure 9-9: Outer level optimization history.

10. Example — ICME Model

This chapter presents the ICME model implementation in LS-OPT. First, a summary of the ICME model is given along with an explanation of the various levels in the multi-level framework (Section 10.1). An example showing the LS-OPT setups for the different levels of the ICME model is then presented in Section 10.2. It demonstrates the ability to integrate the multi-level, multi-phase and multi-step material calibration with a multi-disciplinary vehicle analysis using LS-OPT. The demonstration has been given for a single baseline vehicle analysis, but the setup can also be used for performing a multi-disciplinary design optimization.

10.1. Summary of the ICME model implementation using LS-OPT® and LS-DYNA®

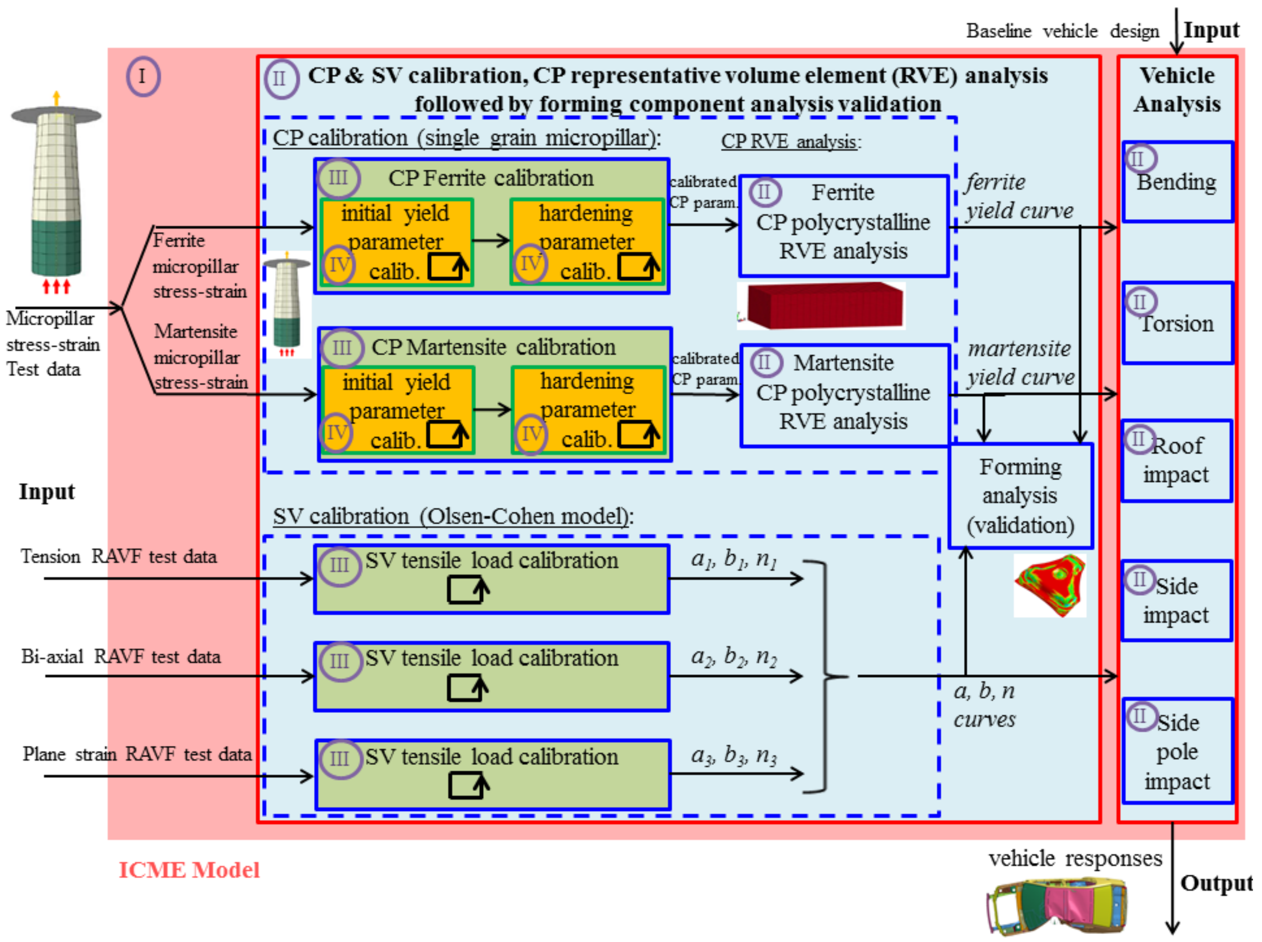
An introduction to the ICME model was provided in Chapter 1, where it was mentioned that it is a multilevel model that integrates material engineering and design optimization. A recapitulation of the integrated model is given in this section along with a demarcation of the various levels involved in the setup. Figure 10-1 presents a summary of the ICME process flow used to calibrate the material models at two length scales and to use the calibrated materials for the analysis a baseline vehicle design. It consists of four nested levels and each level consists of various stages connected as components of a series, parallel or a mixed process. The different levels are depicted using different colors. The stages belonging to a particular level are outlined with a different shade of the same color, but the fill color is the same only if the stage does not have nested levels (i.e. non LS-OPT type stages). A simpler representation of the same multilevel ICME framework was also shown in Figure 1-1. The four levels in the setup are explained below.

1. The red box represents the outermost level (level I). The level I consists of two stages represented using red outlined boxes – material calibration stage, and vehicle analysis stage. The material calibration stage takes the experimental test data as input and provides the individual phase yield curves and the calibrated state variable (SV) model as output. These act as input to run the vehicle analysis. Both the stages in level I (material stage and vehicle analysis stage) are of type LS-OPT and consist of nested sub-processes. Thus, both constitute inner levels (level II) and are shown using blue boxes. To avoid ambiguity, these will be referred to as material level II and vehicle level II.
2. The vehicle level II consists of five LS-DYNA stages with different loadings and is the innermost level for the vehicle optimization. It receives the calibrated LS-DYNA material cards as input and performs an optimization with respect to the design parameters. Here the demonstration of the integrated setup is given for a single material (QP980 steel), but the methodology is general enough to calibrate multiple materials and to use the material as another optimization variable. The material level II branches into two stages representing parallel calibration of the CP and SV models. The CP calibration stages provide the optimal CP parameters for each phase, which are passed on to phase-

specific polycrystalline analysis. The polycrystalline analysis using LS-DYNA provides the phase-specific yield curves, which are used for a forming analysis used as a validation step. These yield curves are also one of the outputs of material level II. The SV calibration stages for three load cases provide the optimized SV parameters as additional material level II outputs.

3. The phase-specific CP calibration and load specific SV calibration are both LS-OPT type stages and consist of inner levels (level III). As these stage stages represent level II sub-processes, they are filled with the green color. In the case of SV model, the third level is the innermost level and consists of an iterative optimization (represented with ⬆) to find the best parameter values.

4. The CP calibration for each phase is performed in two steps – (1) only the initial yield part of the curve is considered to estimate some of the parameters in the first step, and (2) the hardening parameters are calibrated in the second step while fixing the parameters calibrated in the previous step. Therefore, unlike the SV calibration, the level III phase-specific CP calibration stage consists of another nested level (level IV) with two stages outlined in green, which represent the two steps involved in CP calibration (initial yield parameter calibration and hardening parameter calibration). Therefore, the stages representing the initial yield and hardening CP calibration are filled with the yellow color to indicate the fourth level. The fourth level is the innermost CP calibration level and consists of an iterative optimization (⬆).



Baseline vehicle design
Input
I
II CP & SV calibration, CP representative volume element (RVE) analysis followed by forming component analysis validation
Vehicle Analysis
CP calibration (single grain micropillar):
CP RVE analysis:
III CP Ferrite calibration
initial yield parameter IV calib.
hardening parameter IV calib.
calibrated CP param.
II Ferrite CP polycrystalline RVE analysis
ferrite yield curve
II Bending
Ferrite micropillar stress-strain
Micropillar stress-strain Test data
Martensite micropillar stress-strain
III CP Martensite calibration
initial yield parameter IV calib.
hardening parameter IV calib.
calibrated CP param.
II Martensite CP polycrystalline RVE analysis
martensite yield curve
II Torsion
Forming analysis (validation)
II Roof impact
Input
SV calibration (Olsen-Cohen model):
Tension RAVF test data
III SV tensile load calibration
a_1, b_1, n_1
II Side impact
Bi-axial RAVF test data
III SV tensile load calibration
a_2, b_2, n_2
a, b, n curves
Plane strain RAVF test data
III SV tensile load calibration
a_3, b_3, n_3
II Side pole impact
ICME Model
vehicle responses
Output

Figure 10-1: Summary of the ICME model. Here the demonstration is given for a single material without loss of generality. In the presence of multiple materials, there will be several replicas of the material level II with different input test data. The outputs of all the material level II setups will be transferred to a single vehicle optimization level II. In reality the yield curves for two more phases, i.e. Austenite and Transformed Martensite, are required for the final analysis. The CP parameters were not calibrated for these phases in this work due to the absence of single grain micropillar test data. The yield curves from a previously calibrated EPSC model were used as input instead.

10.2. LS-OPT® setup and results

10.2.1. Outer Level I: Integrated material calibration and vehicle design optimization

The outer level (level 1) setup consists of two LS-OPT stages involving material calibration and vehicle thickness optimization. Since the calibrated material is used in the vehicle optimization stage, the two stages are laid out in series with file transfer operations to copy the optimized material data to the vehicle optimization stage. A design of experiments (DOE) task consisting of a single sample is defined using a dummy variable as the sole goal of this level is to run the underlying processes involving material calibration and vehicle optimization. If required, the optimum vehicle-based responses obtained in the inner level can be extracted as responses of this level. The LS-OPT setup files of the inner levels and only the main LS-DYNA input files with parameter definitions are copied to the lower level directories using ‘extra input files' option of the stage setup. The rest of the include files of the LS-DYNA runs can be defined using relative paths to save disk space. The LS-OPT setup of the outer level is shown in **Error! Reference source not found.**.

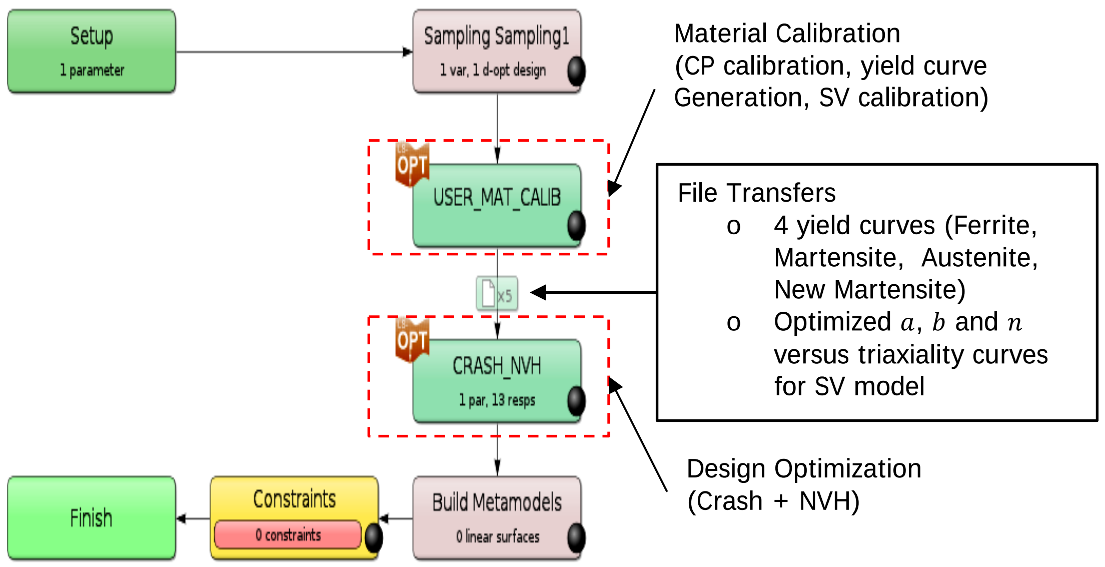


Figure 10-2: Outermost level (Level I): Integrated material calibration and vehicle design optimization.

10.2.2. Inner Material Level II: material calibration and validation

This is the first level of material calibration and consists of an assembly of a multiscale framework for material calibration and validation. Figure 10-3 shows the assembly of stages involved in this level.

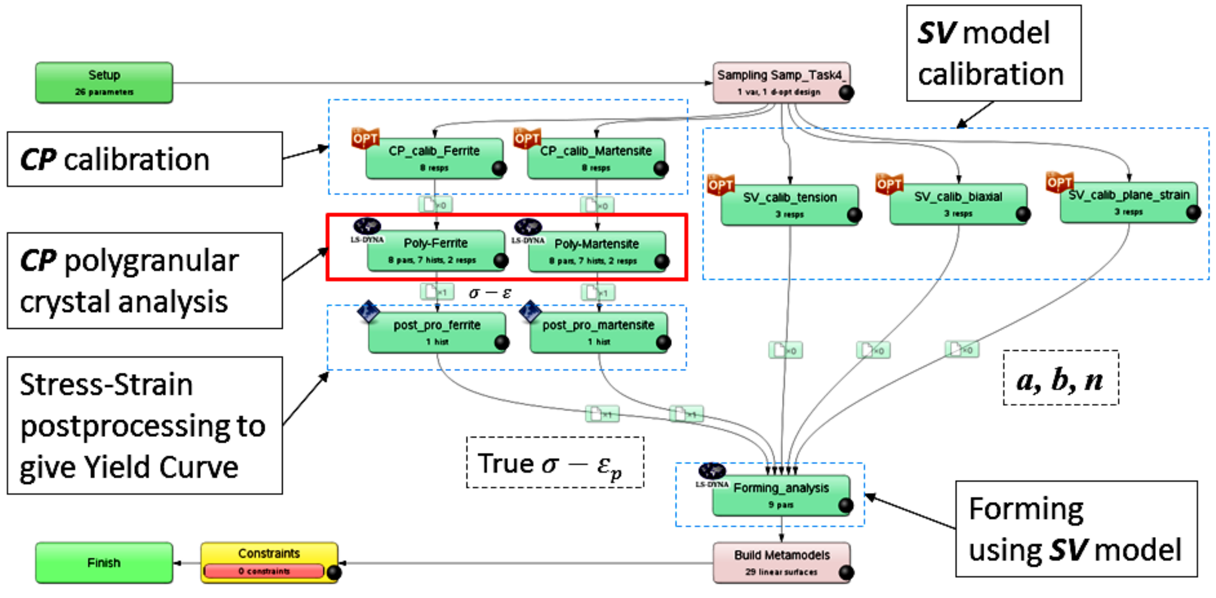


Figure 10-3: Inner Material level II: Material calibration and forming component validation analysis.

The crystal plasticity (CP) model is calibrated for Ferrite and Martensite phases in the inner material level II, as shown by two the LS-OPT stages in the setup. The resulting optimum material parameters are extracted as responses and fed to the LS-DYNA analysis of the polycrystalline model using the response-variable parameter type. The polycrystalline analysis consists of a representational volume element with multiple grains with random orientations. The true stress-strain curves for Ferrite and Martensite obtained from the respective polycrystalline analysis are plotted in Figure 10-4 along with the final deformed shapes. Stress localization and softening were observed despite using the new two-step CP calibration procedure explained earlier. The stress-strain curves for Austenite and Transformed Martensite are obtained using EPSC model instead of CP due to the unavailability of single grain Austenite and Transformed Martensite crystals for calibration. These EPSC model-based curves were available a priori, which is the reason we do not see those two phases as part of the LS-OPT setup. Those curves are fed directly to the “Forming_analysis” stage and to the vehicle analysis.

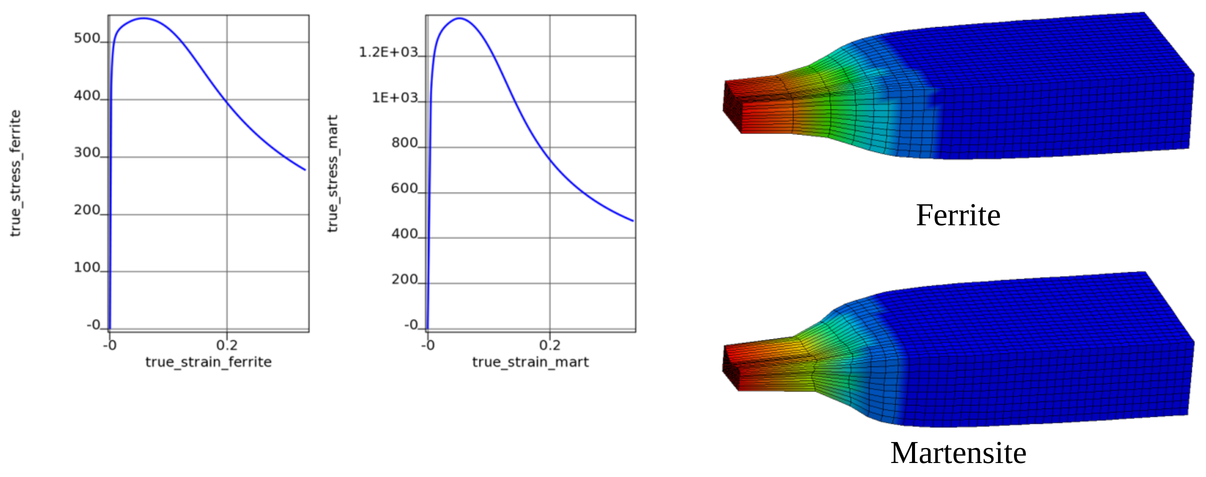


Figure 10-4: True stress-strain curve for Ferrite and Martensite obtained from polycrystalline CP analysis and the corresponding final deformed shapes. The CP analysis is done using the optimal parameters values obtained in the stages “CP_calib_Ferrite” and “CP_calib_Martensite”.

As the SV model, which is used for component and system analysis, requires yield curves instead of the complete true stress-strain data, the stress-strain data obtained from the polycrystalline analysis of both Ferrite and Martensite phase are converted into yield curves (i.e. stress vs plastic strain curves starting at the yield point) using a post processor. The conversion is based on detecting the onset of yield by examining the curvature of the stress-strain curve. This conversion is performed in the stages “post_pro_ferrite” and “post_pro_martensite”.

The SV model is calibrated for three stress states (biaxial, plane strain, tension) based on the Olson-Cohen model and the resulting optimum values of a, b and n constants of each stress state are extracted as responses. The yield curves obtained from the polycrystalline analysis and the optimum a, b and n values of three stress states are fed to the forming analysis for validation (Figure 10-5). This data is also transferred to the vehicle design optimization stage, as shown by the file transfer operations defined in the outer level.

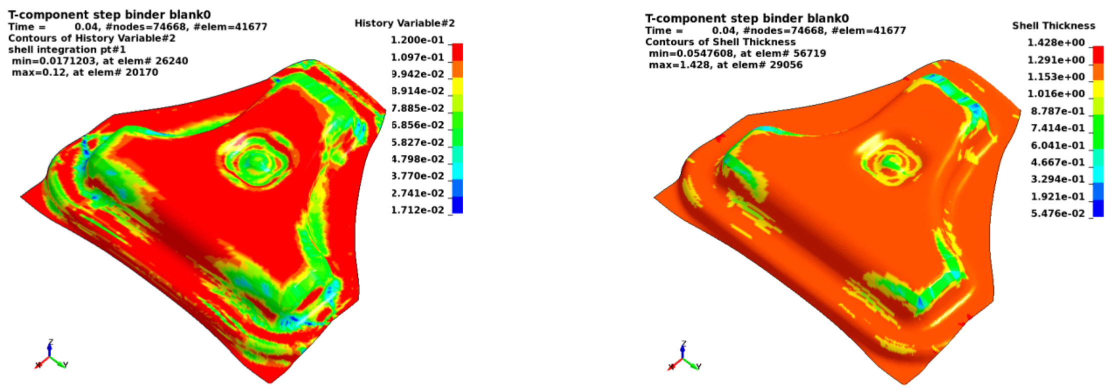


Figure 10-5: Fringe plots of RAVF (left) and shell thickness (right) for the forming component using SV model analysis.

It should be noted that there is no optimization at the material level II. A DOE task with a single sample and a dummy variable is defined to run the inner level processes and the LS-DYNA runs (polycrystalline and forming analyses) of this level. The actual optimization of both CP and SV models are performed in nested inner levels.

10.2.3. Inner Material Level III: calibration of CP and SV models

This level consists of material calibration of CP model for Ferrite and Martensite phases and calibration of SV model for three stress states. In the case of the CP calibration, a two-step method (Section 10.2.4) is followed and therefore the actual optimization does not take place in level III. However, the optimization of the SV model parameters takes place at this level. The SV model is calibrated for three stress states, biaxial, tension and plane strain. The retained austenitic volume fraction (RAVF) is calculated based on the Olson-Cohen model and the RAVF vs. equivalent plastic strain data is matched with the experimental data using a, b and n parameters as the design variables. The LS-OPT setup for SV material calibration for one stress state is shown in Figure 10-6.

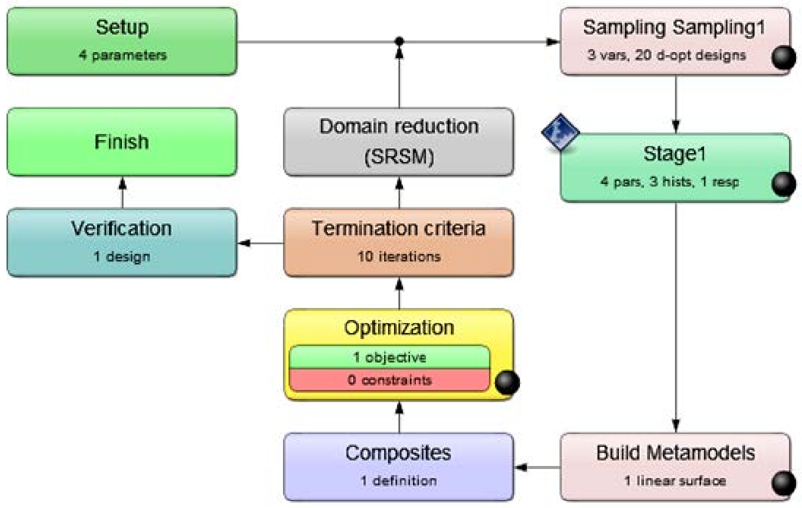


Figure 10-6: Inner material level III for SV model calibration.

The calibrated parameter values are presented in Table 10-1 and Figure 10-7, and Figure 10-8 shows the final fit between computed curves and experimental data for the three load cases.

Parameter / Load case	a	b	n
Tension	21.10	1.31	1.93
Plane strain	27.26	0.92	1.36
Biaxial load	5.14	1.52	1.00

Table 10-1: Calibrated SV model parameter values for three load cases.

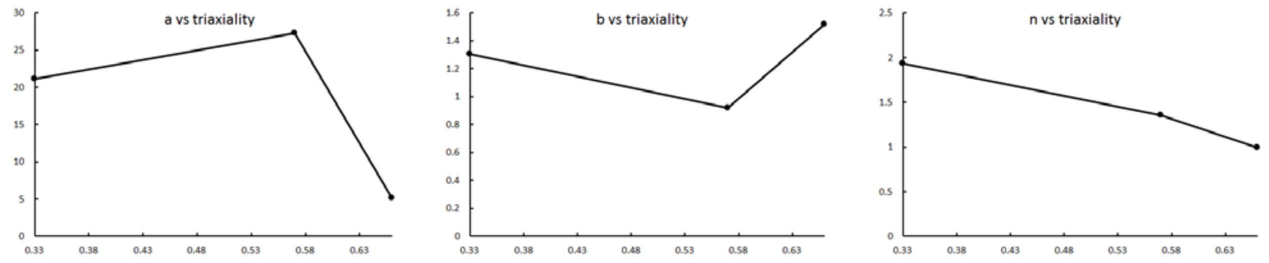


Figure 10-7: Calibrated SV model parameter values for three load cases plotted versus the respective triaxiality values.

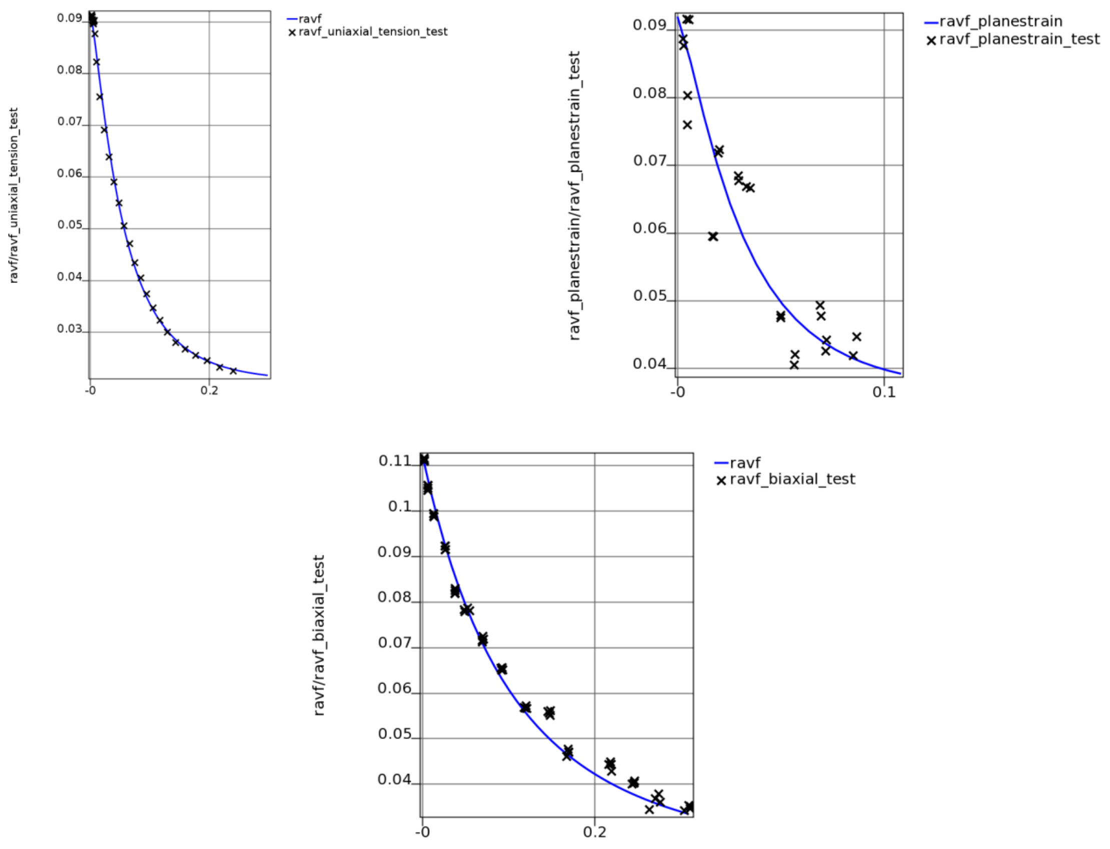


Figure 10-8: Calibration results of retained austenitic volume fraction vs. equivalent plastic strain for tension (top left), plane strain (top right) and biaxial (bottom) stress states.

10.2.4. Inner Material Level IV: two-step calibration of CP model

The CP model is calibrated in this level using compression test data (stress-strain) from single grain Ferrite and Martensite micropillars with known orientations. A constrained two-step calibration method is used.

1. The first step is used to calibrate the parameters affecting the elastic behavior and the yield stress (foundation elastic modulus, τ_{01} and τ_{02}) only. Only the elastic region and the beginning of plastic region are used as test data. The following constraint is applied.

$$\tau_{01} \leq \tau_{02}$$

2. The hardening part of the curve is used in step 2 to calibrate τ_{s1}, τ_{s2}, h_{01} and h_{02}. The optimum values of τ_0 and the foundation elasticity modulus obtained in step 1 are used here. The following constraints are applied to the optimization problems:

$$\tau_{s1} \le \tau_{s2}$$
$$\tau_{01} \le \tau_{s1}$$
$$\tau_{02} \le \tau_{s2}$$
$$\tau_{s1} - \tau_{01} \le h_{01}$$
$$\tau_{s2} - \tau_{02} \le h_{02}$$

The LS-OPT setup for the two step process is shown in Figure 10-9. The setup consists of two LS-OPT stages, which represent the two calibration steps. The stage "initial_yield" in Figure 10-9 calibrates the foundation elastic modulus, τ_{01} and τ_{02} while the stage "hardening" computes the optimal values of τ_{s1}, τ_{s2}, h_{01} and $h_{02.}$ As this setup is used to run the inner level calibration processes, a DOE task is selected along with 'baseline run only' option from the task selection window. The optimum values of the foundation elastic modulus, τ_{01} and τ_{02} from the inner level are extracted as responses of the first stage ("initial_yield"). These responses are then linked to the parameters of the second stage ("hardening") using the response variable parameter type. The parameters τ_{s1}, τ_{s2}, h_{01} and h_{02} are then calibrated based on these previously calibrated values of the foundation elastic modulus, τ_{01} and $\tau_{02.}$

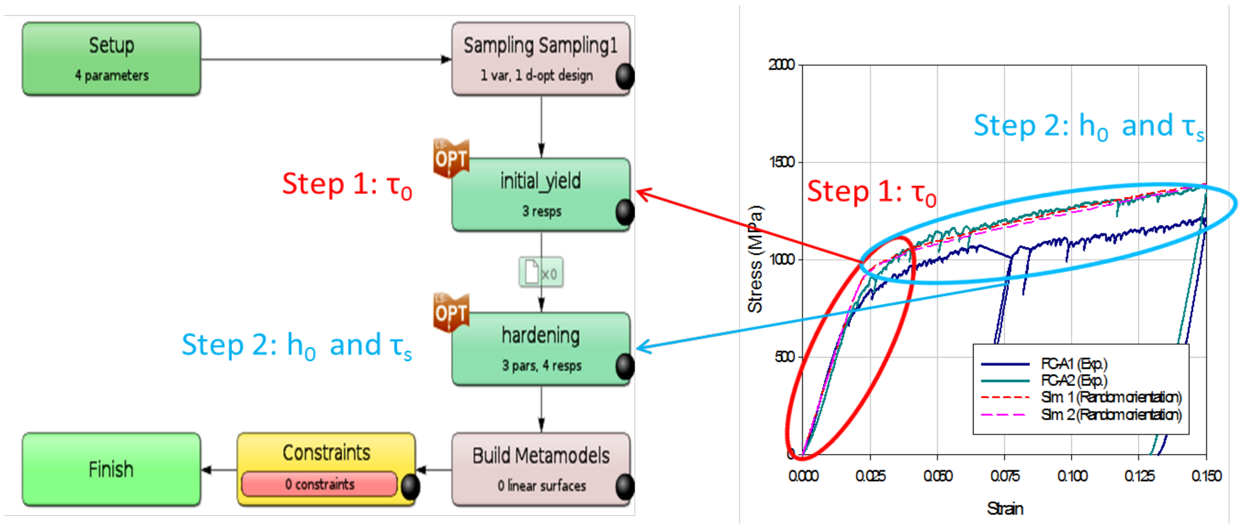


Figure 10-9: LS-OPT setup for two step CP calibration. The optimal values of the parameters calibrated in the stage "initial_yield" remain constant during the calibration of hardening parameters in the stage "hardening".

The inner levels for the two stages "initial_yield" and "hardening" are shown in Figure 10-10 for the Ferrite phase. The setup for Martensite calibration is also similar. It is noteworthy that the step 1 calibration corresponding to "initial_yield" (Figure 10-10 top) has 3 optimization parameters (see sampling dialog) while the second step (Figure 10-10 bottom) has 4 optimization parameters.

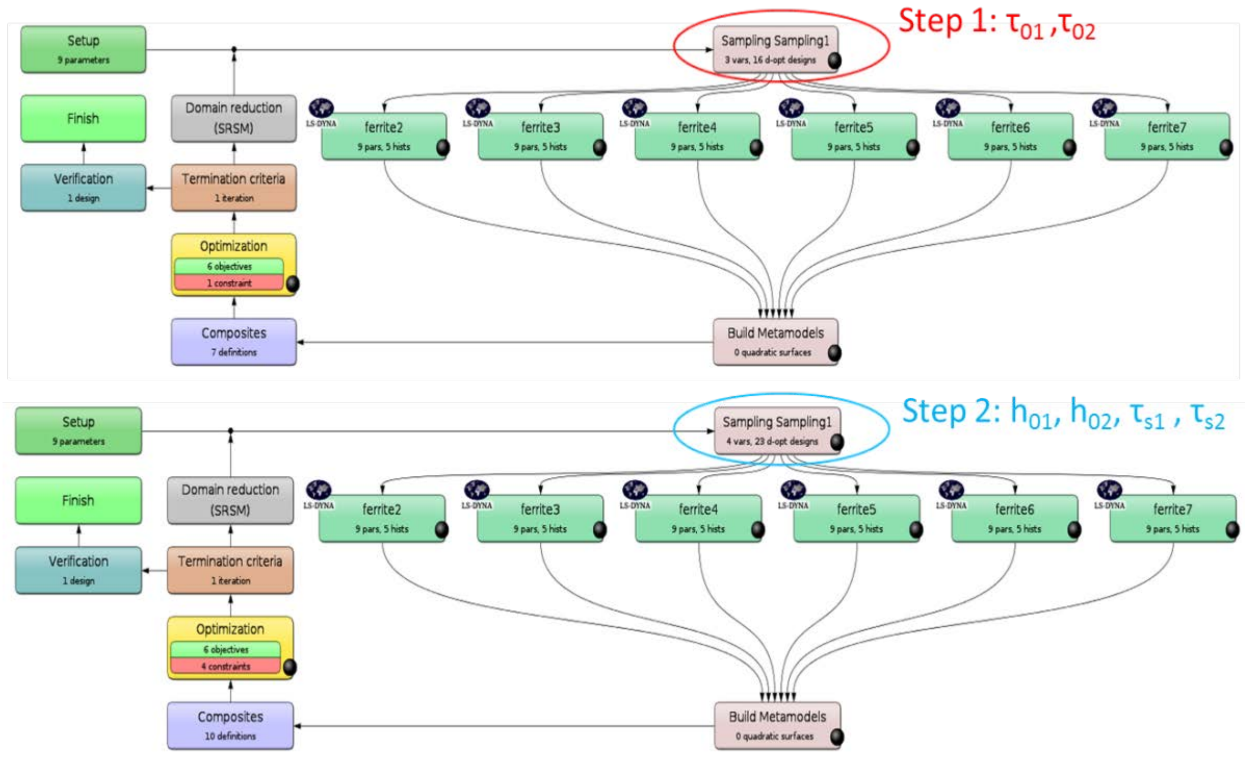
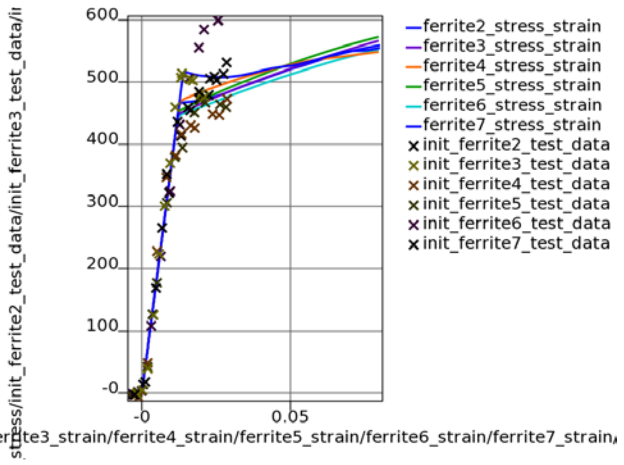


Figure 10-10: Innermost levels for Ferrite CP calibration. The top setup is the innermost level for calibrating the initial yield parameters while the bottom setup pertains to the calibration of hardening parameters.

At this level, most of the setup for optimizing the initial stress parameters and the hardening evolution parameters is similar, however, with a minor difference in terms of number of design variables, constraints and the experimental data considered for matching. In the case of Ferrite, the stress-strain data obtained from the single grain micropillar tests at different orientations are matched with the corresponding experimental data, as shown by multiple load cases in the LS-OPT setup. Figure 10-10 shows the LS-OPT setup for calibrating the Ferrite phase at this level. The calibration results for Ferrite with six different grain orientations are shown in Figure 10-11. In the case of Martensite, three sets of micropillar test data with the same orientation were available, and therefore, only one LS-DYNA stage is needed in the setup. The computed stress-strain curve is plotted in Figure 10-12, overlaid with the three sets of experimental data. The calibrated parameter values for the two phases are listed in Table 10-2. The results for Ferrite are also shown separately for each orientation in Figure 10-13.



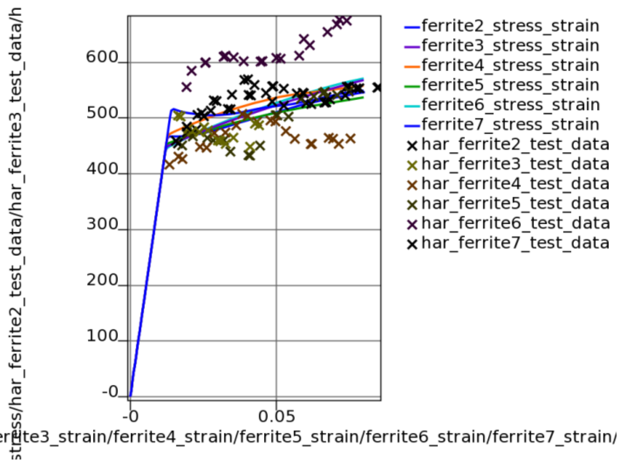
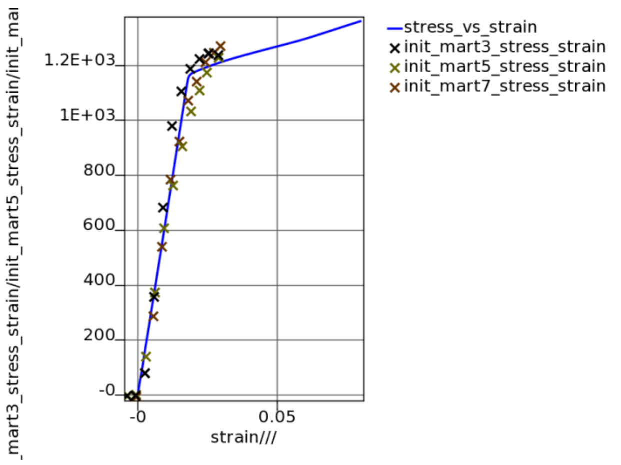


Figure 10-11: CP calibration results of step 1 (left) and step 2 (right) for Ferrite. The stress-strain curves for six different orientations are shown together.



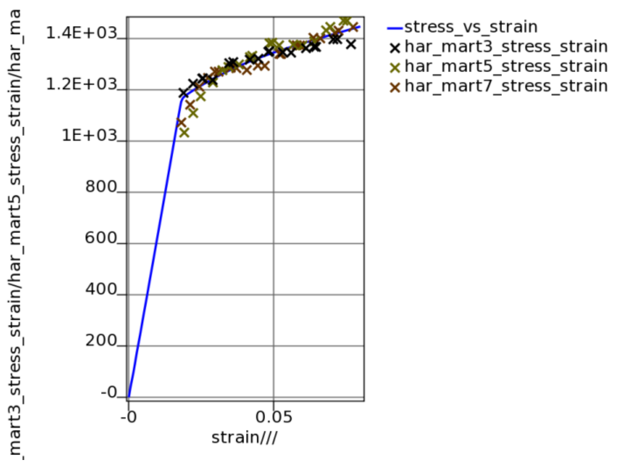


Figure 10-12: CP calibration results of step 1 (left) and step 2 (right) for Martensite.

Parameter / Phase	τ01	τ02	τs1	τs2	h01	h02
Ferrite	204.17	204.17	290.83	290.83	1056.90	87.08
Martensite	482.35	3900.02	549.04	3922.56	8000	107.92

Table 10-2: Calibrated CP parameter values for Ferrite and Martensite.

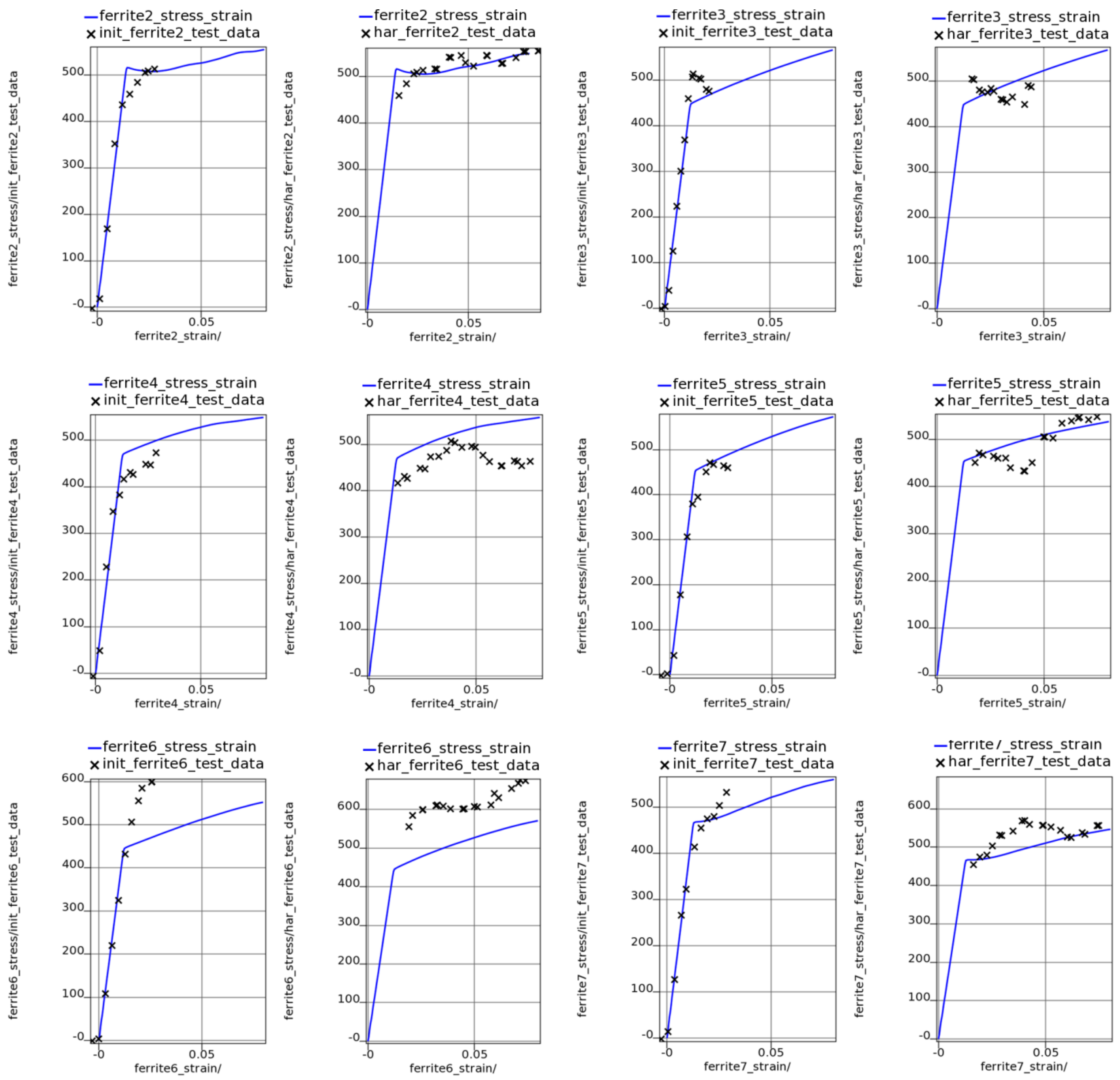


Figure 10-13: CP calibration results of step 1 and step 2 for Ferrite. The stress-strain curves for six different orientations are shown separately (ferrite2 to ferrite7).

10.2.5. Inner Vehicle Level II: vehicle design optimization

This level consists of multidisciplinary design optimization of the full-scale vehicle model. The thickness and material of the selected design parts are defined as variables with mass minimization of the parts as objective and vehicle-based responses as design constraints. Since only one material is calibrated in this example, all the selected design parts use the calibrated material and are defined using string constant parameter type of LS-OPT. If multiple material models are calibrated, string variable types can be used for selecting the optimum material for each part. The phase yield curves of the polycrystalline analyses and a, b and n versus the triaxiality curves transferred from the material calibration stage of the outer level are defined as include files for the multidisciplinary vehicle analysis. The LS-OPT setup of this level is shown in Figure 10-14. Only a baseline analysis was performed here, as it is sufficient to demonstrate the integrated material calibration and vehicle analysis. Figure 10-15 shows the deformed shapes for pole crash, side impact and roof impact. Additionally, displacement and effective stress fringe plots are also shown in Figure 10-16 for the pole crash, bending and torsion cases.

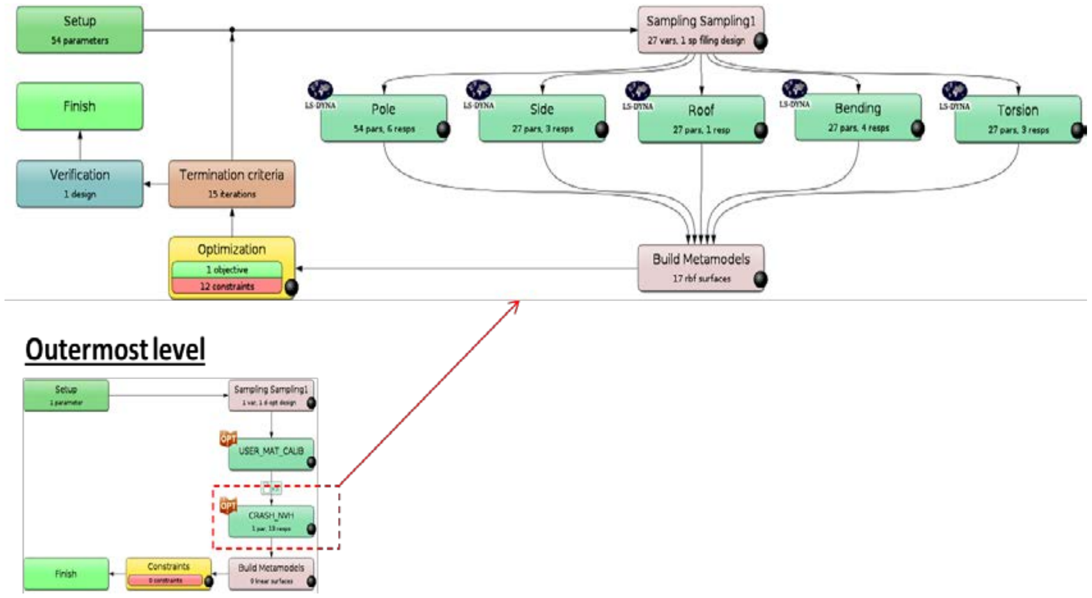


Figure 10-14: Inner vehicle level II: vehicle design optimization.

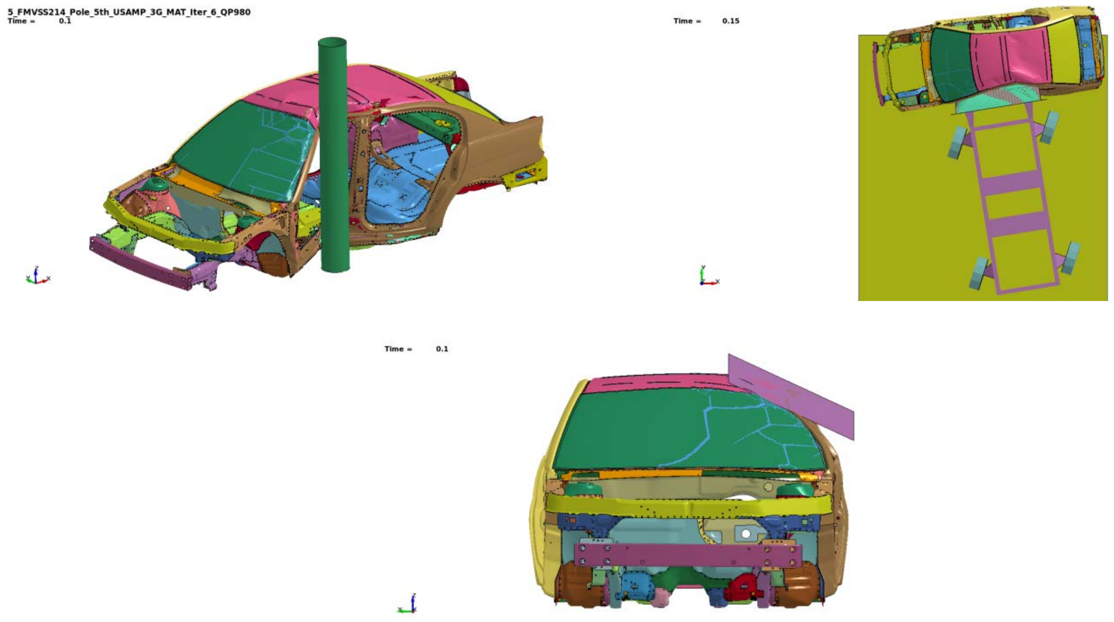


Figure 10-15: Deformed vehicle shape due to pole crash, side impact and roof impact.

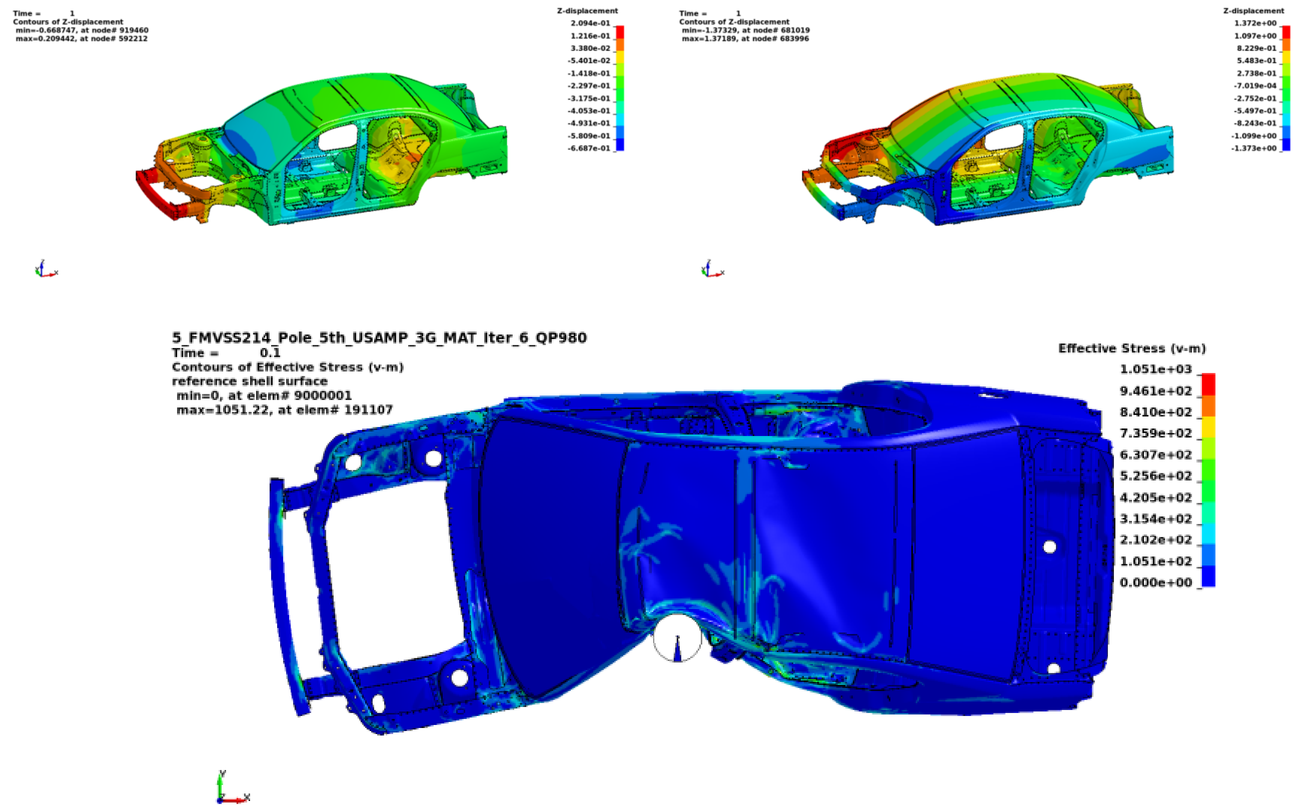


Figure 10-16: Displacement fringe plots for bending (top left) and torsion (top right), and effective stress (MPa) fringe plot for pole crash.

11. Acknowledgement and Disclaimer

The authors wish to thank the many ICME participants from the various universities, research institutions and industries for their collaboration as well as the following colleagues for their development work, especially of the graphical user interface: Katharina Witowski (DYNAMore GmbH, Stuttgart, Germany) and Åke Svedin and Christoffer Belestam (DYNAMore AB, Linköping, Sweden).

This material is based upon work supported by the Department of Energy under Cooperative Agreement Number DE-EE0005976 with United States Automotive Materials Partnership LLC (USAMP).