2021 R1
Ansys
©2021 ANSYS, Inc.
All Rights Reserved.
Unauthorized use, distribution or duplication is prohibited.

Statistics on Structures User's Guide

ANSYS, Inc.
Southpointe
2600 ANSYS Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
http://www.ansys.com
(T) 724-746-3304
(F) 724-514-9494

Release 2021 R1
January 2021

Table of Contents

Chapter 1: Getting Started with SoS

ANSYS Statistics on Structures (SoS) is an ANSYS optiSLang extension for multi-dimensional data analysis. These topics provide basic information about SoS:

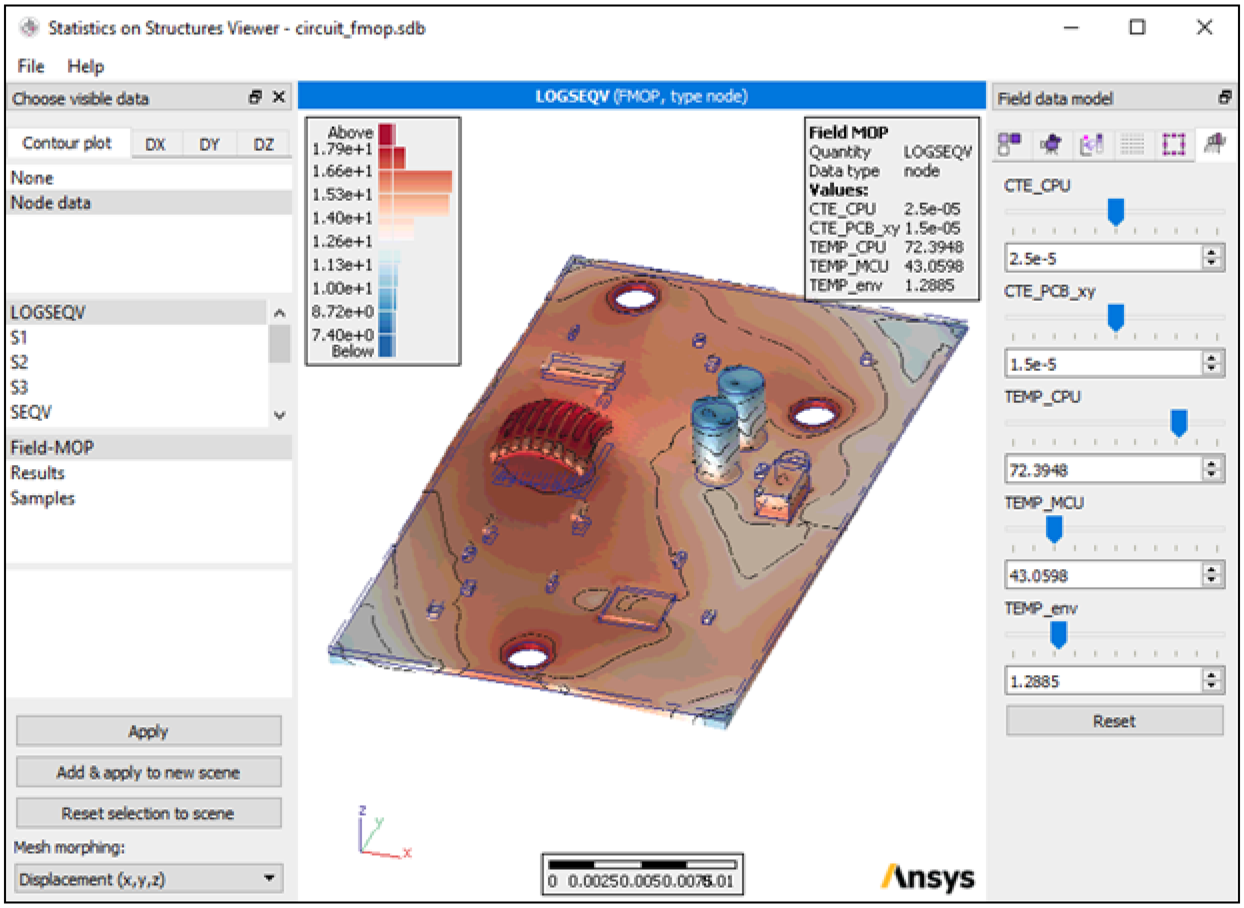
1.1. SoS Overview

The primary objective of SoS is to extend optiSLang's capabilities to distributed data like FEM mesh data (3D) or signals (1D and 2D). With SoS, you can visualize various statistical properties, analyze and expand random fields, simulate random structural designs, and much more.

SoS Algorithms

SoS provides algorithms for:

- Real-time approximations of 3D, 2D, and 1D field data
- Free-form shape optimization
- Automatic parametrization of spatial variations based on virtual or real experiments
- Mesh morphing for the creation of new random designs
- Analysis of random properties of structures by inspecting statistics on the structure
- Detection of hot spots at potential failure locations, sensor positions, and so on
- Random field modeling
- Analysis of measurements such as STL and photos



Field MOP

Quantity	LOGSEQV
Data type	node
Values:	
CTE_CPU	2.5e-05
CTE_PCB_xy	1.5e-05
TEMP_CPU	72.3948
TEMP_MCU	43.0598
TEMP_env	1.2885

CTE_CPU: 2.5e-5
CTE_PCB_xy: 1.5e-5
TEMP_CPU: 72.3948
TEMP_MCU: 43.0598
TEMP_env: 1.2885

SoS algorithms can be grouped as follows:

- Statistical hot spot detection
- For data-based ROMs, sensitivity analysis and approximation with field-MOPs
- Random fields based on measurements
- Random fields based on autocorrelation models

SoS Software Features

The following software features support your multi-dimensional data analysis:

- ANSYS Mechanical plugin for geometric imperfections and result export
- GUI for import, export, analysis, and visualization
- Standalone SoS 3D viewer
- Interactive visualization with sliders for meta models
- Display changes of geometry and their effect on component behavior
- Embedded scripting (Lua) for very fast loops on large data sets

- Binary interfaces to field-MOP (DLL)
- Several CAE interfaces (ANSYS, ABAQUS, LS-DYNA, NASTRAN, STL, PNG, and more)

For your convenience, the next few topics define SoS terms, abbreviations, and acronyms and provide a proven path for learning how to use SoS.

1.1.1. SoS Terms

SoS uses these terms:

Term	Definition
Activity	Active samples are typically input for all statistical algorithms. Inactive samples are ignored.
Component	Subset or selection of a group of finite element nodes or finite elements.
Data object	Single field data object or scalar number. A data object represents a specific field realization and can be visualized.
Design identifier	Name of the sample (number) or name of a specific result.
Design number	Unique number of a sample.
Element data	Field data type where a single discrete value is stored at each finite element.
Field amplitude	Scalar parameter that weighs the influence of a scatter shape onto the variation of a field quantity.
Generated data	Data objects created by SoS. Generated data includes scalar samples and result objects of hot spots.
Hot spot	Denotes the location of potential failure. This depends on the used robustness definition. It is typically the point with largest scatter (standard deviation, value range) or extremal quantile value.
Missing item	Denotes a single or multiple component in a field data object vector for which no data exists. Missing items are used to indicate eroded elements, errors in result computations, and errors in projection algorithms.
Neighbor	Neighbors are data objects that belong to the respective data object.
Node data	Field data type where a single discrete value is stored at each finite element node.
Part	Namespace (="scope") within the finite element structure. Each part defines its own node and element numbering space.
Quantity	Set of data objects with equal quantity identifier.
Quantity identifier	Name of the physical quantity associated with a set of data objects.
Random field components	Descriptive parameters of a random field model, such as field amplitudes.
Random field decomposition	Methodology for building a random field model, such as a decomposition of field samples.

Term	Definition
Random field model	Numerical representation of variations on a FEM mesh with respect to parameters (scaling coefficients).
Reference components	Named selections or groups of nodes or elements that define a substructure of the FEM mesh. The reference node set and reference element set are components that are used to store all statistical data. The fixed node set defines the part of the surface that is not changed in case of geometric deviations.
Reference dictionary	Template defining the contents of any design directory used for import and export.
Reference mesh	FEM mesh on which all statistical data are stored, visualized, and computed.
Result object	Data object with a design identifier that is not a number (latter case would be a sample object).
Scalar data	Scalar number.
Scatter shape	Field data object that associates a variation of some field quantity to the respective field amplitude with value 1.
Spatially global sensitivity analysis	Sensitivity analysis based on a field-MOP (metamodel of optimal prognosis), which is a plotting of the field COP (coefficient of prognosis) on the mesh.
Spatially local sensitivity analysis	Sensitivity analysis at a hot spot (MOP of scalar parameters).
Superset	In mathematics, especially in set theory, a superset is the opposite of a subset.
Update state	If a data object is marked as 'needs update' at least one of its masters or neighbors was deleted or recomputed since its creation.
Variation	Amount of explainable variation by the random field model with respect to the true scatter. • Global variation is a single (average) value describing the variation for the whole field. • Local variation is a field quantity that defines the variation for each position on the mesh.

1.1.2. SoS Abbreviations and Acronyms

SoS uses these abbreviations and acronyms:

Abbreviation	Definition
CDF	Cumulative distribution function
CoD	Coefficient of determination
CoP	Coefficient of prognosis
CoV	Coefficient of variation

Abbreviation	Definition
DOE	Design of experiments
F-CoP	Field coefficient of prognosis
F-MOP	Field metamodel of optimal prognosis
Field-MOP	Field metamodel of optimal prognosis
MOP	Metamodel of optimal prognosis
RBF	Radial basis function
RPCA	Robust Principal Component Analysis
SDB	SoS database file
SoS	Statistics on Structures
SSC	SoS script file
SMOP	Signal metamodel of optimal prognosis
stddev	Standard deviation

1.1.3. Learning How to Use SoS

To learn how to use SoS, consider following this proven path:

1. Become familiar with the SoS interface (p. 15).

2. Review the key capabilities (p. 35).

3. Work your way through tutorials, using precomputed examples and the guidance provided for their analysis.

4. Look at supplied examples (p. 45), which include pre-analyzed SoS database files, raw data to import and analyze, and scripting examples for advanced usage.

5. Review information on core functions and supported file format specifications:

   - SoS File and Data Management (p. 63)
   - SoS Data Objects (p. 81)
   - SoS Field Data Models (p. 95)
   - SoS File Format Specifications (p. 109)

6. Read the theory (p. 167) behind SoS algorithms.

7. See the ANSI C API documentation and examples for the FMOP Solver. Documentation in HTML format is in `C:\Program Files\Dynardo\Statistics on Structures\`*`version`*`\FMOPSolver\share\doc`.

8. See the scripting API documentation for SoS, which is provided in HTML format in `C:\Users\Public\Documents\Dynardo\Statistics on Structures\`*`version`*`\doc`.

1.2. SoS Installation

On the ANSYS customer site (http://www.ansys.com/customercommunity), the SoS package is available on the **Downloads** page under **Add-On Packages**.

These topics provide requirements and installation procedures:

Note:

Release notes are available in `release_notes.rtf`. This file is in the root SoS installation directory.

1.2.1. SoS System Requirements

SoS can be installed on either a Microsoft Windows or Linux operating system.

Supported Windows Versions

- Microsoft Windows 7, 64 Bit
- Microsoft Windows 10, 64 Bit

Supported Linux Versions

- CentOS/Red Hat Linux 6 (6.0 and higher), 64 Bit
- CentOS/Red Hat Linux 7 (7.0 and higher) relying upon Red Hat's compatibility insurance, 64 Bit

  For more Linux distributions, contact ANSYS Support (p. 12).

1.2.2. License Manager

SoS uses the FlexLM license manager because it is available for both Microsoft Windows and Linux hosts. A quick start guide for installing FlexLM is available in PDF format (`licensing_guide_common.pdf`) in `C:\Users\Public\Documents\Dynardo\Statistics on Structures\`*`version`*`\doc`.

1.2.3. Installation on Microsoft Windows

To install SoS on Microsoft Windows, you extract files from the ZIP file in the downloaded SoS package and then run a setup executable that guides you through the installation.

Note:

If during the installation you choose to install SoS so that it is accessible to anyone using this computer, you must have administrator privileges.

SoS Directories

An SoS installation on Windows consists of these directories:

`%PROGRAMFILES%\Dynardo\Statistics on Structures\`*`version`*

Contains SoS program files and libraries, including license files of third-party software.

`%COMMON_DOCS%\Dynardo\Statistics on Structures\`*`version`*

Contains licensing and scripting API documentation and examples. `%COMMON_DOCS%` typically points to `C:\Users\Public\Documents`.

`%USERHOME%\AppData\Local\Dynardo\Statistics on Structures`

Contains autosave files and files that are created at runtime for debugging purposes.

SoS Environment Variables

When SoS is installed on Windows, the following system environment variables are created automatically for all system users:

`SoS_exe`

Points to the `sos.exe` file for the latest SoS installation.

`SoS_debug_exe`

Points to the `sos_console.exe` file for the latest SoS installation.

`SoS_install_dir`

Points to the latest SoS installation directory so that you can generate the `SoS_exe` environment variable by entering this command, including the quotation marks:

```
"%SoS_install_dir%\\sos.exe"
```

`SOS_PUBLIC_DIR`

Points to the directory containing all of the tutorials and examples. On Windows, this is typically `C:\Users\Public\Dynardo\Statistics on Structures\`*`x.y.z`*. optiSLang uses the variable *`x.y.z`* to find the custom integration nodes of SoS.

Note:

Although environment variables are generally case insensitive, SoS and optiSLang interpret them as case sensitive.

SoS Startup

You can start SoS on Windows using any of these methods:

- Click the **Statistics on Structures** icon on your desktop (if the option for this icon is selected during SoS installation.
- From the Windows start menu, select **Statistics on Structures**.
- In the command line, to start SoS in batch mode, enter the following command, including the quotations marks:

  `"%SoS_exe%"`

SignalMOP in optiSLang 8.1 (ANSYS 2020 R2)

optiSLang 8.1.0 is shipped with the `FMOPSolver.dll` from SoS 7.1.4. This means that the **MOP Solver** node can handle SignalMOP models created with SoS 7.1.4 or earlier versions. However, the **MOP Solver** node can handle SoS 8.0.0 SignalMOP models if you replace the DLLs in the optiSLang installation path manually.

Note:

You may need administrator privileges.

1. Remove all DLL files located in `%PROGRAMFILES%\ANSYS Inc\v202\optiSLang\oop\fmopsolver`.
2. Copy the following files from your SoS installation path (`%PROGRAMFILES%\Dynardo\Statistics on Structures\`*`%VERSION_NUMBER%`*) to your optiSLang installation path (`%PROGRAMFILES%\ANSYS Inc\v202\optiSLang\oop\fmopsolver`):
   - `libgcc*.dll`
   - `libgfortran*.dll`
   - `libgomp*.dll`
   - `libquadrmath*.dll`

- `libstdc++*.dll`
- `libwinpthread*.dll`
- `libz.dll`
- `mop_api.dll`
- `zipper.dll`

3. Copy `%PROGRAMFILES%\Dynardo\Statistics on Structures\`*`%VERSION_NUMBER%`*`\FMOPSolver\lib\FMOPSolver.dll` to `%PROGRAMFILES%\ANSYS Inc\v202\optiSLang\oop\fmopsolver` and rename it to `FMOPSolver7.dll`.

4. Restart optiSLang.

Uninstalling SoS

To uninstall SoS on Windows, in your list of installed apps and features, right-click **Statistics on Structures** and then click **Uninstall**. The SoS folder and all other installed files are removed from the installation directory.

Note:

In your list of installed programs and features, only the latest SoS installation is available. If you have several SoS installations in parallel, only the latest one is removed automatically. For all other SoS installations, you must run their uninstall executables from within their installation directories.

1.2.4. Installation on Linux

To install SoS on Linux:

1. Unpack the compressed TAR archive in the downloaded SoS package using one of these methods:

   - Copy the archive to the destination path, extract it, and rename it:

     ```
     cp sos_$VERSION_$LINUX.tar.gz $DEST
     cd $DEST
     tar -xjf sos_$VERSION_$LINUX.tar.gz
     mv sos_$VERSION_$LINUX sos
     ```

   - Extract the archive directly to the destination path:

     ```
     tar -C $DEST -xjf sos_$VERSION_$LINUX.tar.gz
     ```

2. Add SoS environment variables (p. 10) as needed.

Caution:

If installing SoS in a multi-user environment (or in a directory other than a specific user's `home` directory), the `sos` directory within the destination path must be user-writeable. If this is not possible due to IT security requirements, type the following in a shell prompt to create this path: **`mkdir $DEST/sos/.sos_help`**. Otherwise, SoS will crash immediately after starting.

SoS Directories

The destination path of your SoS installation on Linux contains these file system items:

`tutorials/`

Contains guided workflows for using SoS.

`Field-MOPSolver/`

Contains the entry point of the **`Field-MOPSolver`** dynamic link library integration. For more information, see FMOPSolver Dynamic Linked Library (p. 162).

`examples/`

Contains input files for tutorials, file format specification examples, and more.

`sos/`

Contains SoS program files and libraries.

`3rd_party/`

Contains the license files of third-party software.

Additionally, after SoS's first execution, `$HOME/.local/share/data/Dynardo/Statistics on Structures` contains autosave files and files that are created at runtime for debugging purposes.

SoS Environment Variables

Especially for interaction with optiSLang, the following environment variables are assumed to exist:

`SoS_exe`

Points to the `sos.sh` file of the latest SoS installation.

`SoS_install_dir`

Points to the latest SoS installation directory so that you can generate the `SoS_exe` environment variable by typing the following command, including the quotation marks:

`“$SoS_install_dir/sos.sh”`

`SOS_PUBLIC_DIR`

Points to the directory containing all of the tutorials and examples. This variable is used by optiSLang to find the custom integration nodes of SoS. You must edit this variable so that the optiSLang custom integration nodes can be found in `$SOS_PUBLIC_DIR/examples/optislang/custom_integration_node/x.y`.

For example, to create these environment variables in a Red Hat Distribution or one of its forks, edit the following two lines in your system configuration and add them to the `.bash_profile` file located at `"$HOME/.bash_profile"`. If you want to propagate these variables to all users system-wide, add them to the file `/etc/profile.d/sos.sh`:

```
export SoS_exe="$HOME/bin/Statistics on Structures 21R1/sos/sos.sh"
export SoS_install_dir="$HOME/bin/Statistics on Structures 21R1/sos"
```

If the file does not already exist, you must create it. Before starting optiSLang the next time it is necessary, reload the configuration file by doing one of the following:

- In the console, type the following command:

  `. "$HOME/.bash_profile`

- Open a new configuration file.

To create these environment variables on other supported operating systems, you should refer to the manuals for these operating systems.

SoS Startup

You start SoS on Linux in batch mode by entering the following in the command line, including the quotation marks:

`$SoS_exe`

SignalMOP in optiSLang 8.1 (ANSYS 2020 R2)

If you want the **MOP Solver** node in optiSLang 8.1 to handle SoS 8.0.0 SignalMOP models, replace the DLLs in the optiSLang installation path manually as indicated in this section (p. 8) of the Windows installation.

Uninstalling SoS

To uninstall SoS on Linux:

1. Remove the directory: `rm -rf [installation path]`.
2. Remove the manually added SoS environment variables (p. 10).

1.3. SoS Command Line Parameters

SoS supports these command line parameters:

`sos -b filename`

Runs the given SoS script in batch mode.

Intended usage: Linux servers.

In batch mode, the graphical user interface of SoS does not appear. Hence, SoS can be used without the graphical environment.

`sos -s filename`

Runs the given SoS script. After running the script, the graphical user interface of SoS appears.

Intended usage: Re-evaluation with changed data, manually edited SoS scripts, and debugging.

`sos -o filename`

Opens the given SoS database file. After running the script, the graphical user interface of SoS appears.

`sos -h filename`

Lists all available command line arguments.

On Microsoft Windows, use these environment variables:

`"%SoS_exe%"`

Opens SoS in default mode.

`"%SoS_debug_exe%"`

Opens SoS in terminal mode for debugging. Potential log output is transferred to the current terminal. If no terminal is opened, one is opened automatically.

1.4. ANSYS Support

If you are evaluating SoS for purchase or have an active support and maintenance contract, contact ANSYS Support with any questions or issues that you might have:

Email: `<support@ansys.com>`

Online: ANSYS Customer Site

Support Hotline (USA): 1-800-711-7199

Support Fax (USA): 724-514-5096

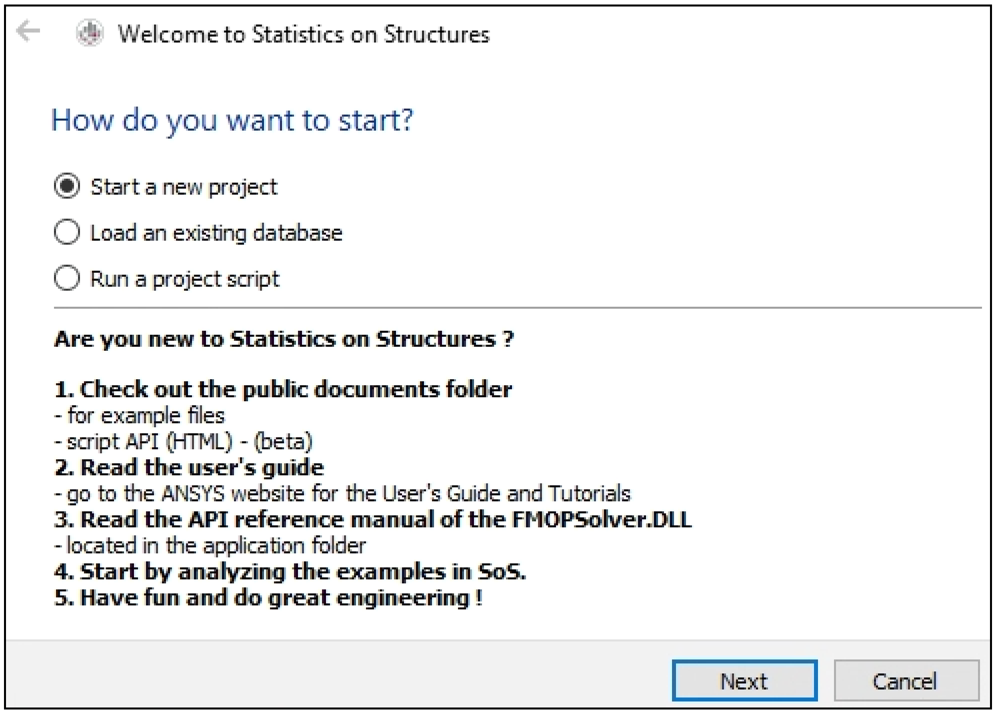
Note:

- Before emailing ANSYS Support about a problem, read through SoS Troubleshooting (p. 173) to see if this section provides a solution or workaround. Next try searching for a solution on the ANSYS customer site. This site provides solutions to thousands of problems that you might encounter when using ANSYS products.

- When emailing ANSYS Support about a problem, send detailed information in an *incident report*. For more information, see Filing Incident Reports (p. 177).
- When telephoning the ANSYS Support hotline, call Monday through Friday from 8:00 AM to 5:00 PM U.S. Eastern time, excluding holidays. If you are outside the U.S., contact your regional ANSYS office or ANSYS partner for support.

Chapter 2: SoS Interface

When you start SoS, the **Welcome to Statistics on Structures** wizard provides options for starting a project, loading an existing database, or running a project script. Listed below these options are suggestions in case you are new to SoS.


Welcome to Statistics on Structures
How do you want to start?
Next
Cancel

[✓ Start a new project]
Load an existing database
Run a project script

Are you new to Statistics on Structures ?

1. Check out the public documents folder
- for example files
- script API (HTML) - (beta)

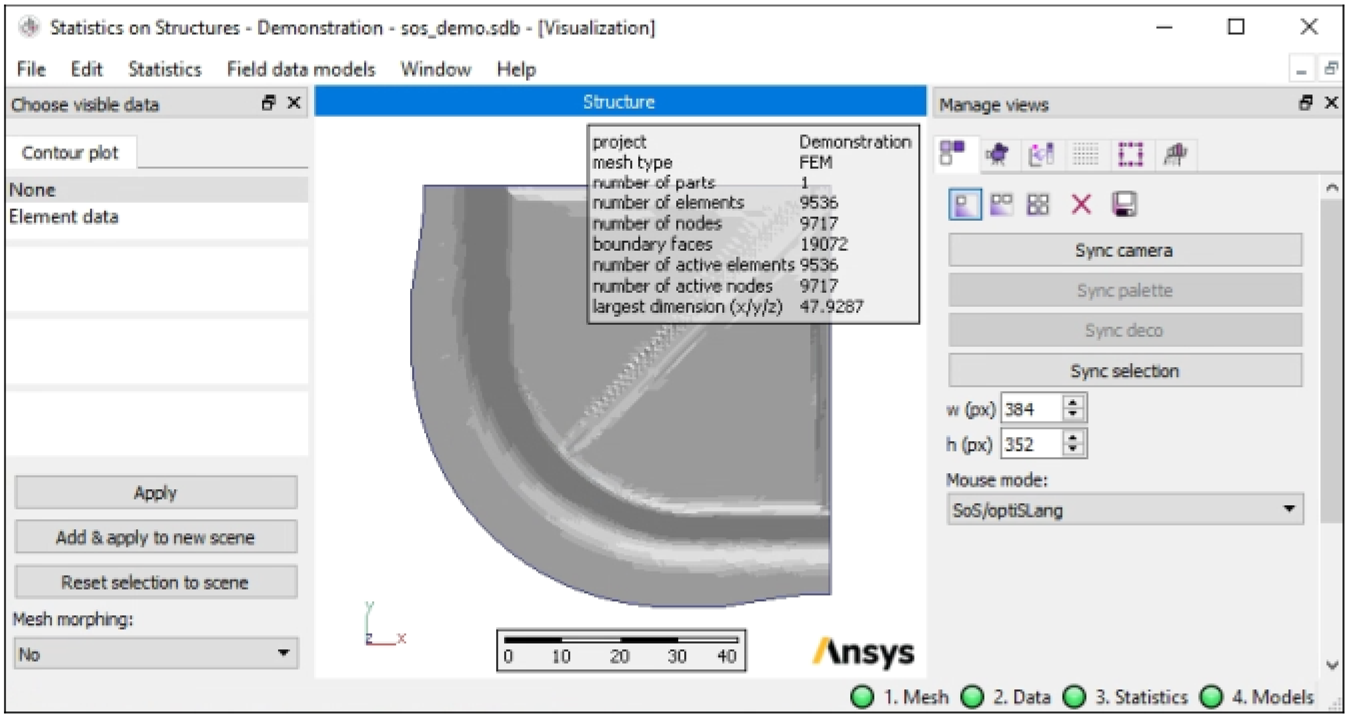
2. Read the user's guide
- go to the ANSYS website for the User's Guide and Tutorials

3. Read the API reference manual of the FMOPSolver.DLL
- located in the application folder

4. Start by analyzing the examples in SoS.
5. Have fun and do great engineering !

This section assumes that you have loaded the supplied database file `sos_demo.sdb` in `C:\Users\Public\Documents\Dynardo\Statistics on Structures\`*`version`*`\examples`. You can quickly open this folder by selecting **Help** → **Open examples folder**.

SoS is initially configured to show the 3D visualization in the main pane and a docked window on either side:



project	Demonstration
mesh type	FEM
number of parts	1
number of elements	9536
number of nodes	9717
boundary faces	19072
number of active elements	9536
number of active nodes	9717
largest dimension (x/y/z)	47.9287

Mesh morphing: No

w (px) 384

h (px) 352

Mouse mode: SoS/optiSLang

In this simplified configuration, you select an object in the **Choose visible data** window on the left and then select options from the **Statistics** menu to calculate statistical measures. You use the options in the docked window on the right to manipulate rendering of the 3D visualization.

However, you can easily customize how SoS is configured. For example, from options on the **Window** menu, you choose whether to display the 3D visualization, data table, or script log in the main pane. You can also choose to tile or cascade all three in separate windows. Similarly, from options on the**Docked windows** submenu, you choose to show or hide additional windows, which are docked by default, but you can easily move, resize, and floated all docked windows. In the SoS tutorials, many included images show customized layouts.

These topics describe key components of the SoS interface:

Note:

To restore SoS to the default layout, click the Windows menu and type `regedit` to open the Registry Editor. Under **Software** → **Dynardo** → **Statistics on Structures**, select and delete all values. Because the **(Default)** value specifies the default layout, acknowledge the message indicating that it cannot be deleted.

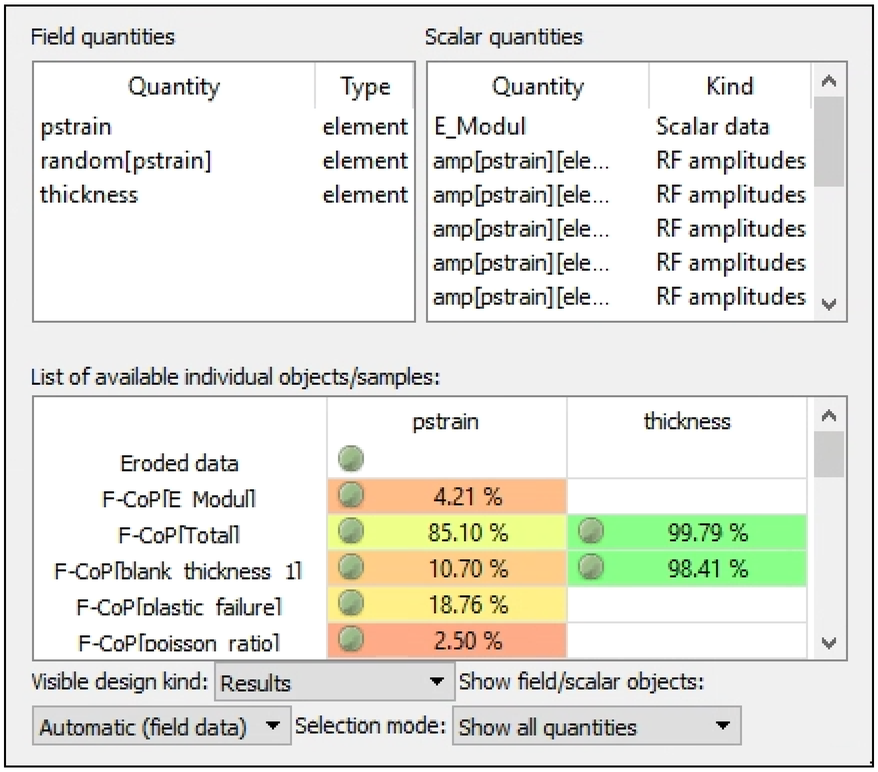
2.1. Main Pane

The **Window** menu provides these options for choosing what is shown in the main SoS pane:

- **Data table**
- **Visualization**
- **Script log**

Data Table

The data table lists the field quantities, scalar quantities, and individual examples from which you can select. Fields near the bottom of the window provide for filtering the lists.



Field quantities

Quantity	Type
pstrain	element
random[pstrain]	element
thickness	element

Scalar quantities

Quantity	Kind
E_Modul	Scalar data
amp[pstrain][ele...	RF amplitudes
amp[pstrain][ele...	RF amplitudes
amp[pstrain][ele...	RF amplitudes
amp[pstrain][ele...	RF amplitudes
amp[pstrain][ele...	RF amplitudes

List of available individual objects/samples:

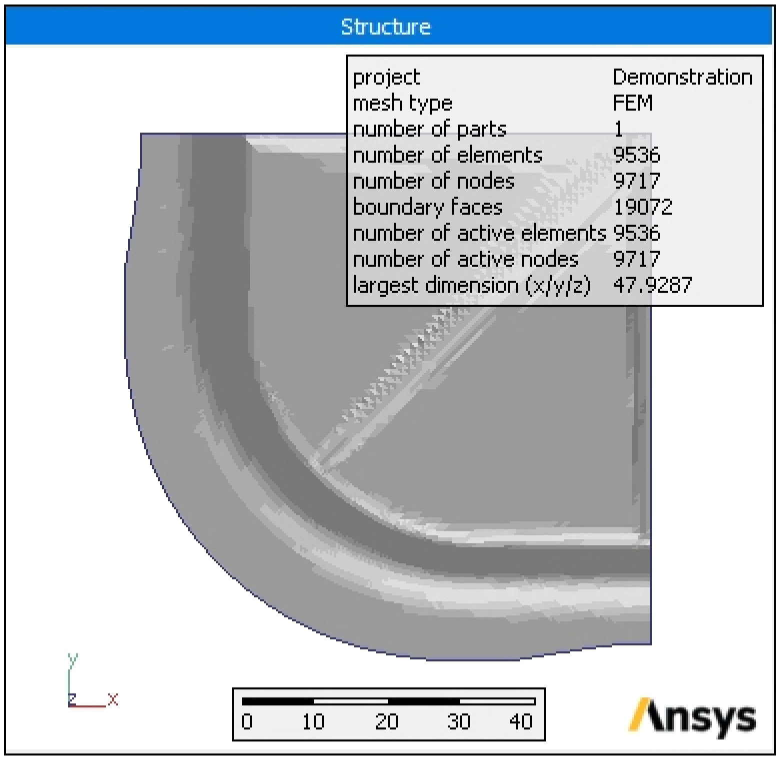
	pstrain	thickness
Eroded data	●	
F-CoP[E_Modul]	● 4.21 %	
F-CoP[Total]	● 85.10 %	● 99.79 %
F-CoP[blank_thickness_1]	● 10.70 %	● 98.41 %
F-CoP[plastic_failure]	● 18.76 %	
F-CoP[poisson_ratio]	● 2.50 %	

Visible design kind: Results
Show field/scalar objects: Automatic (field data)
Selection mode: Show all quantities

To visualize results and samples, you either double-click them or select one or more and then press the **Enter** key. You can repeat this to render the same result or sample several times.

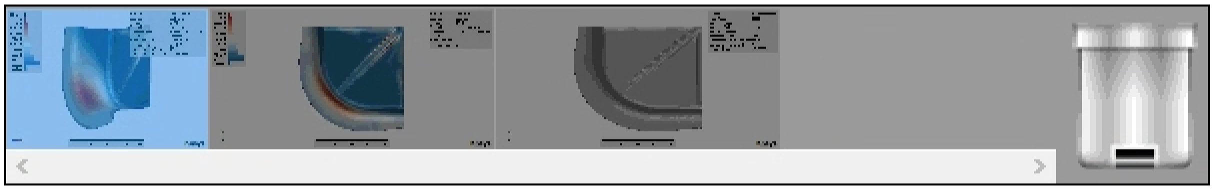
Visualization

The 3D visualization is shown in the main pane by default.



project	Demonstration
mesh type	FEM
number of parts	1
number of elements	9536
number of nodes	9717
boundary faces	19072
number of active elements	9536
number of active nodes	9717
largest dimension (x/y/z)	47.9287

When you render a scene for one or more results and samples, after creating the first *scene*, SoS automatically switches to the 3D visualization. In the gray area beneath the main pane, you can select a *stamp* for a scene to show this scene in the main pane. The scene created last is shown first.

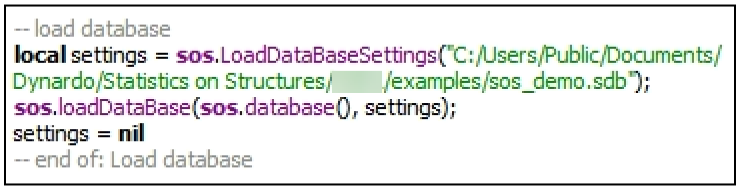


You can place the mouse cursor over a stamp to see information about the scene. The reference structure, which is always shown, cannot be deleted.

- To display a different scene in the main pane, drag the stamp for the scene from the gray area and drop it in the main pane. The stamp for the active scene is highlighted in blue.
- To activate a scene, click the image. Because visualizing the same result or sample in two different scenes is not supported, you cannot drag an already visualized stamp from the gray area and drop it in the main pane.
- To hide an active scene, drag it to the trash icon next to the stamp list.
- To remove a scene, drag the stamp to the trash icon.

Script Log

The script log shows the actions recorded in your current SoS session:



```
-- load database
local settings = sos.LoadDataBaseSettings("C:/Users/Public/Documents/
Dynardo/Statistics on Structures/[redacted]/examples/sos_demo.sdb");
sos.loadDataBase(sos.database(), settings);
settings = nil
-- end of: Load database
```

Window Manipulations

The **Window** menu provides **Tile** and **Cascade** options for displaying the data table, 3D visualization, and script log in individual windows within the main pane. When these windows are cascaded, you might need to rearrange them so that all three are visible.

- When only one of these windows is shown, selecting **Data table**, **Visualization**, or **Script log** from the **Window** menu displays the selected window.
- When multiple windows are tiled or cascaded, selecting **Data table**, **Visualization**, or **Script log** from the **Window** menu makes the selected window active.

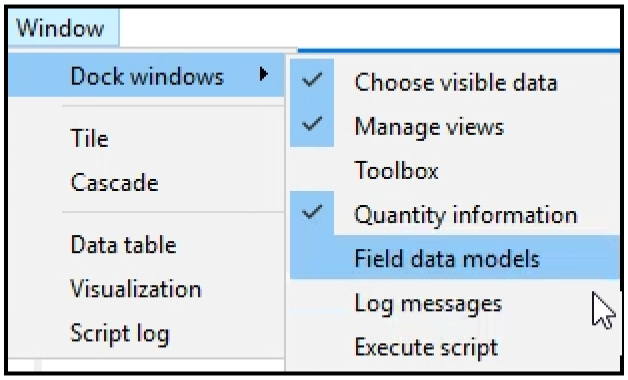
Multiple Item Selection

From the data table and the 3D visualization, you can execute menu options for multiple field quantities, scalar parameters, and individual data objects at once.

- To select a group of consecutive items, left-click and drag.
- To select or deselect a group of consecutive items, press and hold down the **Shift** key while left-clicking items.
- To select non-consecutive items, press and hold down the **Ctrl** key while left-clicking items.

2.2. Docked Windows

The initial setup of SoS shows two docked windows: **Choose visible data** window and **Manage views**. However, if you select **Window** → **Docked windows**, you can see options for showing and hiding many windows:



Note:

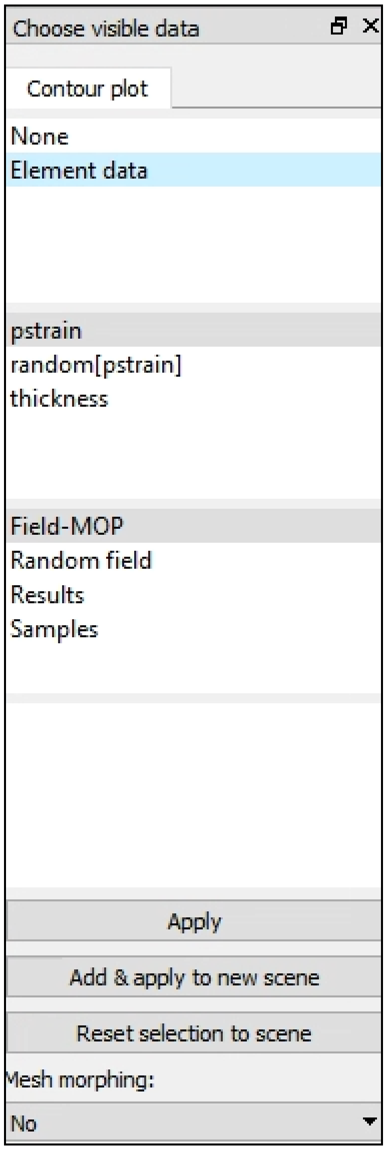
The **Manage views** option is shown when this particular *inspector* is active in the window that contains all of the inspectors. If another inspector is active, such as the **Camera** inspector, its name (**Camera**) appears instead. For more information, see Active "Inspector" Window (p. 21).

To show or hide a window, you select the option for it. A check mark to the left of an option indicates whether the window is shown. While these windows are docked by default, you can move, resize, and float them.

The following topics describe these windows:

2.2.1. Choose Visible Data Window

The **Choose visible data** window is initially docked in the top left corner of the SoS window. You use this window to select the data item on which you want to calculate statistical measures and visualize in 3D. You can visualize existing samples or result objects, or you can generate new data from a random field model or a field-MOP. If both node and element data exist, you can select which data type to visualize.


Choose visible data

Contour plot

None
Element data

pstrain
random[pstrain]
thickness

Field-MOP
Random field
Results
Samples

Mesh morphing:
No

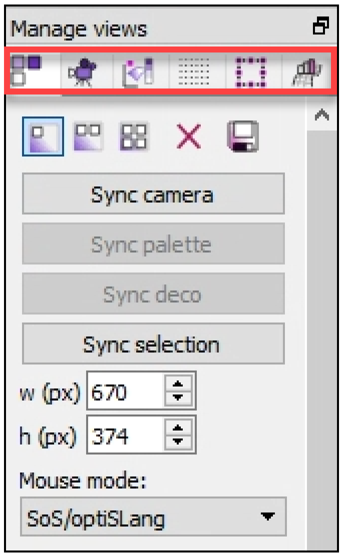
You can display data not only as contour plots but also as geometric deviations. For contour plots, you can select different objects for geometric deviations. For example, you can visualize the stress field as a contour plot that belongs to a specific geometric change.

For **Mesh morphing**, you indicate if the associated data is to be interpreted by selecting **No** or a morphing method: **Normal displacement**, **Coordinates (x,y,z)**, or **Displacement (x,y,z)**.

- Clicking **Apply** registers the selected data sources to the current scene.
- Clicking **Add & apply to new scene** creates a new scene.
- Clicking **Reset selection to scene** clears all setting changes that you have made but not yet applied to the scene, thereby restoring settings to those that were last applied to the scene.

2.2.2. Active "Inspector" Window

You use the docked window that is visible by default on the right to manipulate image rendering in the 3D visualization. This window contains several *inspectors*, which are categorical groupings of rendering options. The window title displays the name of the inspector that is active. The buttons below the window title provide for switching between inspectors.



The following topics describe the inspectors:

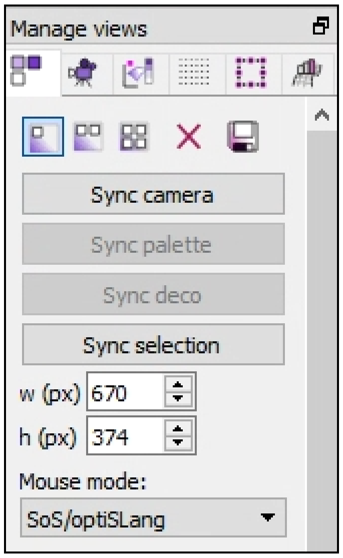
Note:

When selecting **Window** → **Dock windows**, the name of the last active inspector is shown. Selecting this option closes the window containing all of the inspectors.

2.2.2.1. Manage Views Inspector

In the window containing the inspectors, clicking the first icon displays the **Manage views** inspector:

The **Manage views** inspector provides options for choosing the number of visible scenes, saving images, deleting scenes, and syncing scenes.



w (px) 670
h (px) 374
Mouse mode:
SoS/optiSLang

In this table, the options with no text descriptions display their tooltips in parentheses.

Option	**Description**
▢ (Show a single scene)	Shows only one scene at one time.
▢ (Split horizontally)	Shows two scenes at one time, splitting the scenes horizontally.
⊞ (Show 4 scenes in a grid)	Shows four scenes at one time in a grid.
✗ (Deletes all scenes)	Deletes all existing scenes. You are first prompted to confirm their deletion.
💾 (Save active scene as image file with specified width and height)	Saves the active scene to either an image file (such as PNG or JPG) or a high-resolution image file (PDF). The formats supported depend on the operating system on which SoS is running.
Sync camera	Syncs all camera views, including slice settings, according to the active scene.
Sync palette	Syncs all palettes according to the active scene.
Sync deco	Syncs all decoration settings according to the active scene.
Sync selection	Syncs the selection according to the active scene.
w (px)	Sets the width in pixels of the image that is to be saved.
h (px)	Sets the height in pixels of the image that is to be saved.
Mouse mode	Sets the mode of the mouse to either **SoS/optiSLang** or **LS-PrePost**. Additional information about object rotation follows this table.

Note:

The defaults for the width and height of the image that is to be saved are based on the size of the scene.

Object Rotation

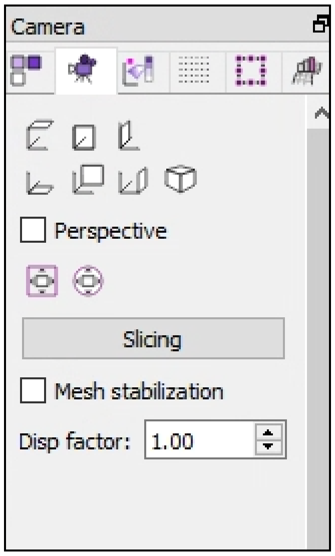
Using the mouse, you can manually rotate objects in the active scene. The default mouse mode is the same as in LS-PrePost. In the mouse mode combo box, tooltips explain the usage of the mouse and keyboard. Defaults follow:

- Rotate: Left mouse button
- Translate: **Shift** + left mouse button
- Zoom: **Alt** + left mouse button
- Fast rendering: Right mouse button
- Area selection: **Ctrl** + hold left mouse button and move mouse
- Pick one element or node: **Ctrl** + left mouse click
- Toggle selection: **Ctrl** + **Alt** + ...
- Translate slicing plane: In active slicing mode, press and hold the **C** key and left mouse button.

2.2.2.2. Camera Inspector

In the window containing the inspectors, clicking the second icon displays the **Camera** inspector:

The **Camera** inspector provides options for rotating, scaling, and slicing an object in the active scene.



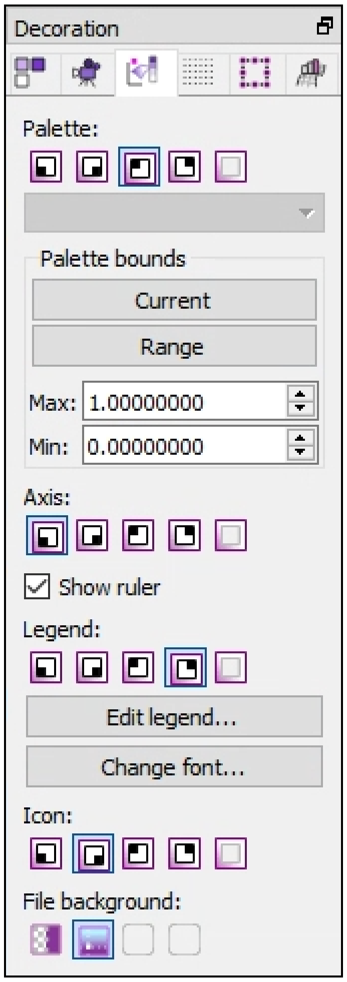
In the following table, for options with no text descriptions, tooltips are shown in parentheses.

Option	Description
(Top view)	Sets the camera to a plain view onto the xy coordinate plane, with the z axis pointing into the screen.
(Front view)	Sets the camera to a plain view onto the xz coordinate plane, with the y axis pointing into the screen.
(Left view)	Sets the camera to a plain view onto the yz coordinate plane, with the x axis pointing into the screen.
(Bottom view)	Sets the camera to a plain view onto the xy coordinate plane, with the z axis pointing out of the screen.
(Back view)	Sets the camera to a plain view onto the xz coordinate plane, with the y axis pointing out of the screen.
(Right view)	Sets the camera to a plain view onto the yz coordinate plane, with the x axis pointing out of the screen.
(Isometric view)	Sets the camera to a standard isometric view with two axis rotated.
Perspective	If this check box is selected, perspective is on. If it is cleared, isometry is used.
(Fit)	Scales and translates the view to fit the entire model for the current rotation.
(Scale)	Scales and translates the view to fit the entire model for any rotation. This option generates a view that is slightly smaller than the previous option.
Slicing	Clicking this button switches between showing and hiding a cutting plane. Initially, the cutting plane is always defined normal to the screen, y-axis pointing into the screen, slicing the structure in the middle of the active scene. The z-axis points to the left, hiding the structure on this side.
Mesh stabilization	After selecting a data object of type **node** that is to be interpreted as a geometric deviation in the **Chose visible data** window, you can select this check box and provide a scaling factor in the next option to visualize displacements. If this check box is selected, the mesh is stabilized when geometric changes are visualized. The iterative algorithm that is activated tries to create proper meshes after application of the geometric deviation. You should use this with care because the iterations can take a long time for large meshes and large deviations.
Disp factor	Sets the scale factor for the applied deviation. A factor of **1** is the default. A value greater than 1 exaggerates the geometric deviation from the reference mesh. **Note:** This scaling is applied after the mesh stabilization algorithm.

2.2.2.3. Decoration Inspector

In the window containing the inspectors, clicking the third icon displays the **Decoration** inspector:

The **Decoration** inspector provides options for customizing all decorations, which include the palette, axis, legend, ruler, icon, and background.


Decoration

Palette:

Palette bounds

Current

Range

Max: 1.00000000

Min: 0.00000000

Axis:

Show ruler

Legend:

Edit legend...

Change font...

Icon:

File background:

Under **Palette**, **Axis**, **Legend**, and **Icon**, you see the same five options for moving and hiding the decorator:

The first four buttons position the decorator, respectively: Bottom left, Bottom right, Top left, and Top right. The last button hides the decorator so that it is no longer visible.

Option	Description
Palette	When creating a new scene, SoS tries to associate the proper palette type and settings with it. However, you can customize

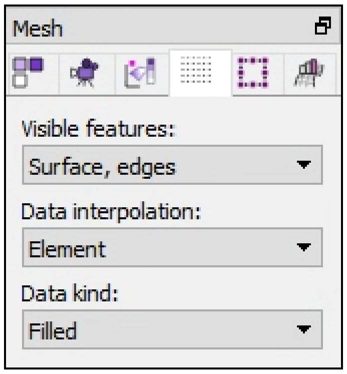
Option	Description
	the palette by using the options provided to modify its position, type, and bounds.
	Use the buttons to position and hide the pallet.
	Use the drop-down box to select the palette type. Choices are: • **F-COP**: Creates a colored plot, using green to indicate high values and red to indicate low values. • **Rainbow (inverse)**: Creates a multicolored plot or its inverse. • **Gray scale (inverse)**: Creates a plot in gray scale or its inverse. • **Traffic light (inverse)**: Creates a plot in either green/yellow/red or red/yellow/green.
Palette bounds	Use these options to specify the palette bounds: • **Current**: Sets bounds automatically from current view. • **Range**: Sets bounds automatically from all samples of the current quantity. • **Max**: Sets upper bound to the manually specified value. • **Min**: Sets lower bound to the manually specified value.
Axis	The coordinate axis is represented by a red ray (x-axis), a green ray (y-axis), and a blue ray (z-axis). Use the buttons provided to position and hide the coordinate axis.
Show ruler	If this check box is selected, the ruler is shown. The scale on the bottom of the screen gives you an idea of the actual structure size. The scale changes as you zoom into or outside the structure.
Legend	You can customize the legend by using the options provided to modify its position, content, and font.
	Use the buttons to position and hide the legend.
	Edit legend: Opens a new window in which you can modify all text that appears in the legend. The legend text is assigned to the respective data object and saved in the SoS database file.
	Change font: Opens a new window in which you can change the font. The font is used in the palette and as the default font in the legend.
Icon	Use the buttons to position the icon or to hide it.
File background	Sets the background of the image files created by SoS.

Option	Description
	The background can be transparent, a single color, or a gradient. The default setting for the background is transparent. While SoS always displays a blue-to-gray gradient background, an exported image file displays whatever background is set when the image is saved to a file. Use the following buttons to change the background: • The first button sets the colors for both the third and fourth buttons to transparent. This is the default. • The second button sets the gradient to the colors selected for the third and fourth buttons. If a color is selected for only the third button, the background is a single color. • The third button opens a window in which to select the background color to use for a single-color background or for the top color in a gradient background. • The fourth button opens a window in which to select the color to use for the bottom color in a gradient background.

2.2.2.4. Mesh Inspector

In the window containing the inspectors, clicking the fourth icon displays the **Mesh** inspector:

The **Mesh** inspector provides options for modifying the mesh.


Mesh

Visible features: Surface, edges

Data interpolation: Element

Data kind: Filled

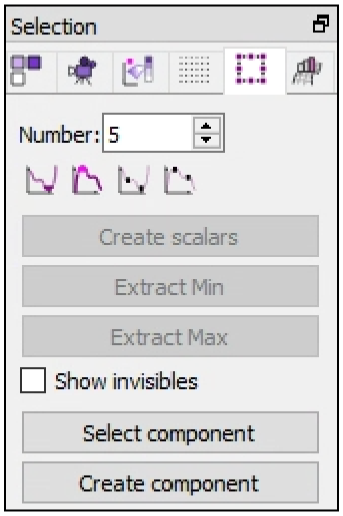
Option	Description
Visible features	Sets the visible features of the mesh according to demands. Choices are: • **Edges**: Shows only topological edges. • **Surface**: Shows only the surface.

Option	Description
	• **Surface, edges**: Shows both the surface and topological edges. • **Wireframe, edges**: Shows the FEM elements as a wireframe with edges.
Data interpolation	Sets how to interpolate data. Choices are: • **Element**: Shows single value per element (default for element data). • **Smooth**: Interpolates between nodes. • **Isolines**: Shows isolines, interpolated between nodes (default for node data). • **Isosurfaces**: Shows isosurfaces in a transparent structure, interpolated between nodes. • **Single Isosurface**: Shows a single isosurface in a transparent structure, interpolated between nodes.
Data kind	Sets what data to display: • **Default**: Shows data values and highlights missing items. • **Filled**: Shows data values and interpolates missing values. • **Missing**: Highlights missing items, thereby creating a two-tone plot.

2.2.2.5. Selection Inspector

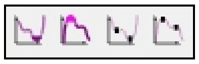
In the window containing the inspectors, clicking the fifth icon displays the **Selection** inspector:

The **Selection** inspector provides options for computing global and local maxima and minima and extracting scalar parameters from these values.



Here are some guidelines for making selections in the active scene:

- To select a single item, click it. To select a group of items, draw a selection window using some key and mouse combination. The mouse and key combination to use depends on the selected **Mouse mode** in the Manage Views Inspector (p. 22). You can draw the selection window from either left to right or from bottom to top.
- To add items to already selected items, press the **Alt** key before making the additional selections.
- To remove all current selections, click somewhere next to the structure.
- To remove only particular items from the current selection, re-select the items to clear them from the selection.
- To replace a current selection, skip pressing the **Alt** key before selecting new items.

Option	**Description**
Number	Limits the number of values to be created. The default is **5**. Changing this limit influences all subsequent selection commands and affects the following operations: • Maximum number of selected data types in (area) selection and therefore the maximum number of extracted (extremal) scalars • Number of computed maxima and minima
	Use the buttons to compute values and label respective elements inside the plot:  You should work with the local maxima and minima. However, problems can occur if the elements found are on a plane or if there is a saddle point. It is therefore advisable to use these

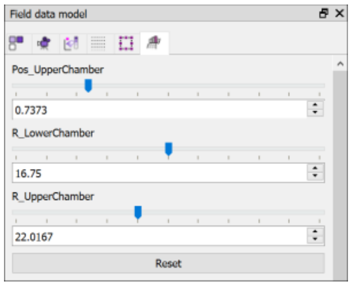
Option	Description
	options to cross check the values obtained with the general smallest and largest values. • The first button selects the *n* smallest values. • The second button selects the *n* largest values. • The third button selects the *n* smallest values of all local minima. • The fourth button selects the *n* larges values of all local maxima.
Create scalars	Extracts the values of the elements and nodes that are selected in the active plot. The new parameters are extracted for all samples of the currently shown field quantity. The function is typically used to export data at hot spots to optiSLang.
Extract Min	Extracts the minimum field value among the selected nodes or elements in the active scene. A scalar quantity is created, containing this minimum value for each sample. You set the name of the scalar quantity when executing the command. **Note:** Select a single scalar data object in the data table's **Sample** view. In the **Quantity Information** window, the **Creating algorithm** line contains the node or element ID of the respective minimum. To open this window, select **Window** → **Docked Windows** → **Quantity Information**.
Extract Max	Extracts the maxima values among the items selected in the active scene for all samples. This command computes the maximum value of the selected data values for each design and extracts this value to a new scalar parameter.
Show invisibles	When this check box is selected, labels that are hidden from view are shown. For example, labels for items that are on a side of the object are not shown unless this check box is selected.
Select component	Opens a window in which to select a node or element component and all active nodes or elements belonging to it. The set type depends on the visible data type.
Create component	Opens a window in which to create a node component or set. You specify the set identifier and set type. Choices for set type are **Node set** and **Element set**.

2.2.2.6. Field Data Model Inspector

In the window containing the inspectors, clicking the sixth icon displays the **Field data model** inspector:

The **Field data model** inspector provides options for computing global and local maxima and minima and extracting scalar parameters from these values.

If the scene represents a random field or a field-MOP, this inspector provides options for changing the values of the random field amplitudes or input parameters for the field-MOP.

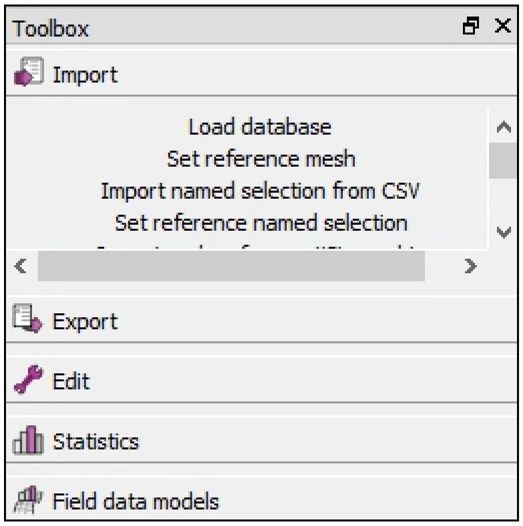


Pos_UpperChamber
0.7373
R_LowerChamber
16.75
R_UpperChamber
22.0167

SoS evaluates the currently displayed quantity for the set values and updates the scene accordingly.

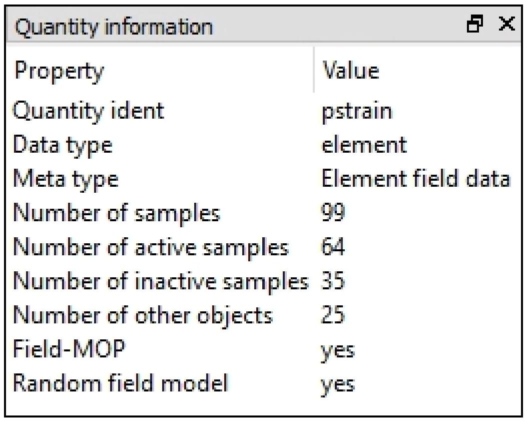
2.2.3. Toolbox Window

The **Toolbox** window provides quick access to SoS menu options under functional category headings:



2.2.4. Quantity Information Window

The **Quantify information** window displays information about quantities:

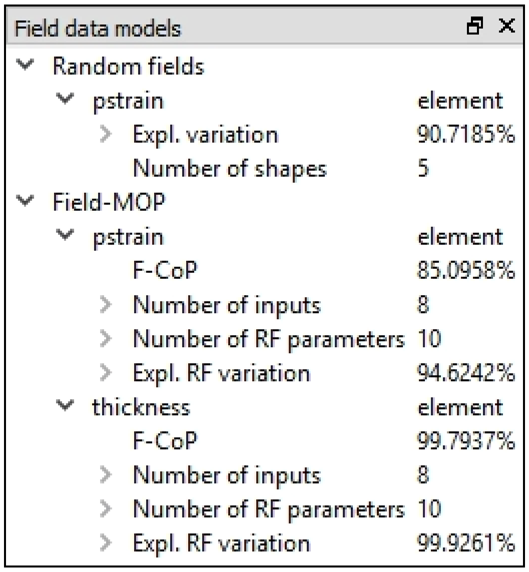


Quantity information

Property	Value
Quantity ident	pstrain
Data type	element
Meta type	Element field data
Number of samples	99
Number of active samples	64
Number of inactive samples	35
Number of other objects	25
Field-MOP	yes
Random field model	yes

2.2.5. Field Data Models Window

The **Field data models** window displays information about random fields and field-MOPs:

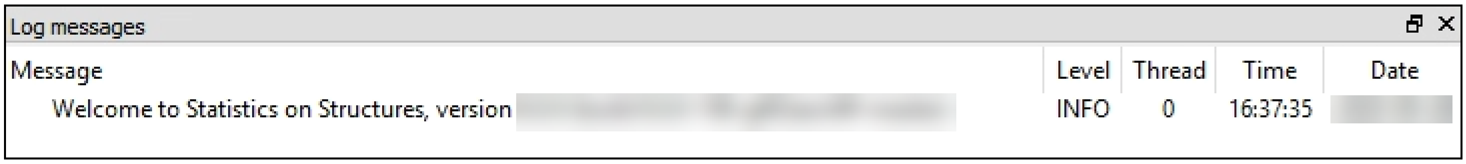


Field data models

Random fields	
pstrain	element
Expl. variation	90.7185%
Number of shapes	5
Field-MOP	
pstrain	element
F-CoP	85.0958%
Number of inputs	8
Number of RF parameters	10
Expl. RF variation	94.6242%
thickness	element
F-CoP	99.7937%
Number of inputs	8
Number of RF parameters	10
Expl. RF variation	99.9261%

2.2.6. Log Messages Window

The **Log messages** window displays the information written to the SoS log file:

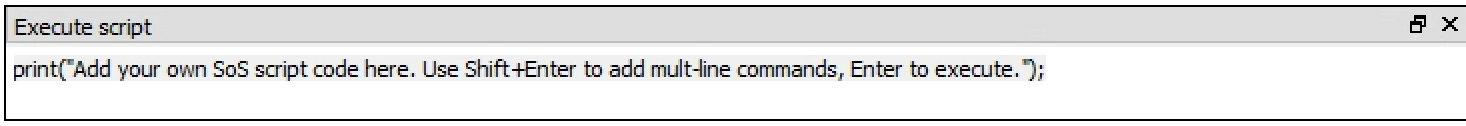


Log messages

Message	Level	Thread	Time	Date
Welcome to Statistics on Structures, version [redacted]	INFO	0	16:37:35	[redacted]

2.2.7. Execute Script Window

The **Execute script** window displays the SoS script code to be executed:



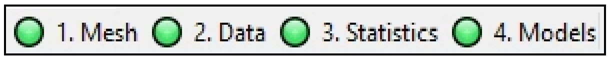
```
print("Add your own SoS script code here. Use Shift+Enter to add mult-line commands, Enter to execute.");
```

2.3. Traffic Light Indicator

The traffic light indicator in the lower right corner of the SoS window guides you through the SoS workflow:

1. Import mesh.
2. Import data.
3. Analyze data.
4. Create field models.

When all four steps are completed, all four lights are green:



In a new project, all lights are red. When you complete a step, the light for this step turns green.

When you place the mouse cursor over a light, a tooltip displays relevant information about the data for this step.

For a hands-on introduction to SoS, see Basic SoS Tutorials in the *Statistics on Structures Tutorials* in the *Statistics on Structures Tutorials*.

2.4. Help Menu

The **Help** menu provides options for:

- Browsing the *Statistics on Structures User's Guide* and *Statistics on Structures Tutorials* on the ANSYS Help site
- Opening the `examples` folder for quick access to sample projects and files
- Browsing SoS scripting and programming library documentation
- Viewing information about this release

Chapter 3: SoS Capabilities

These topics summarize key SoS capabilities:

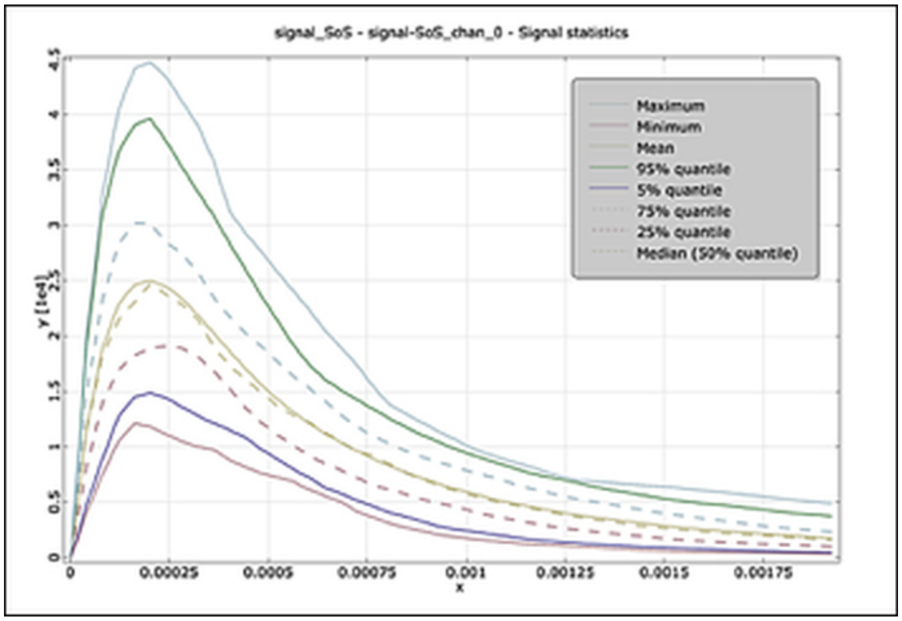
3.1. Spatial Domains

SoS uses these spatial domains:

- 1D domains (curves)
- 2D and 3D grids
- FEM meshes

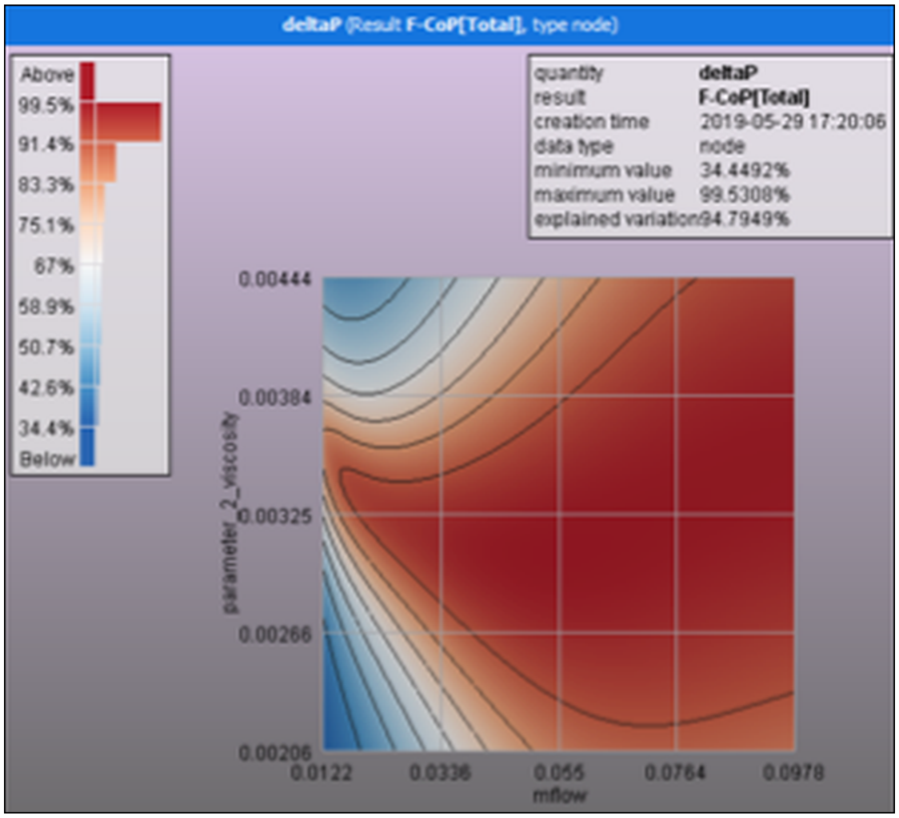
1D Domains (Curves)

The 1D domain is a curve. optiSLang refers to curves as *signal data*, which includes all types of vector data that are assigned to an abscissa axis. Signal data can be visualized in optiSLang postprocessing that uses SoS as a backend.



2D and 3D Grids

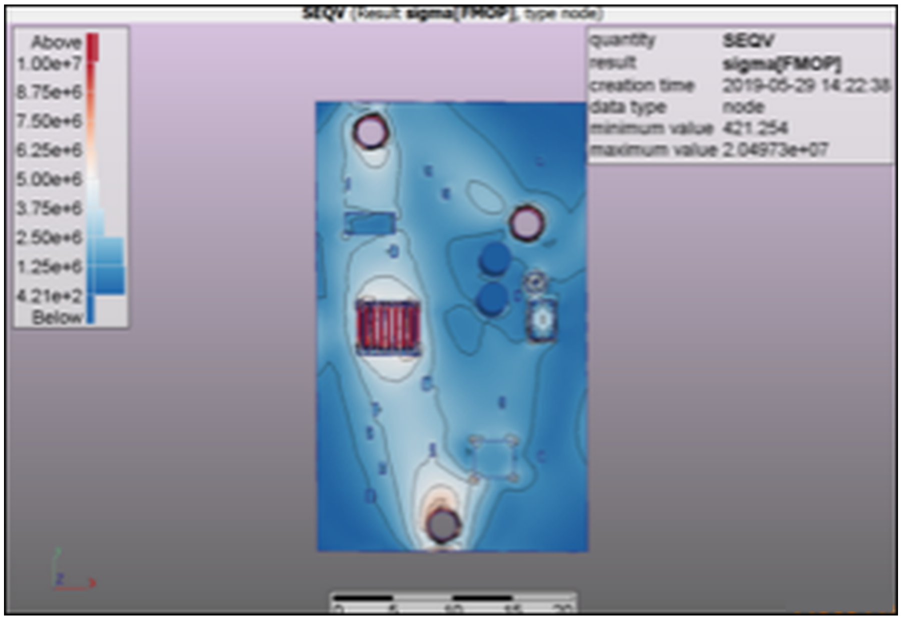
Regular grids are domains where the connectivity is defined by the numbering of the nodes. While signals are 1D grids, there are also 2D grids (matrix data) and 3D grids (voxel data). Examples of 2D grids are performance maps or pixel-based images.



quantity	deltaP
result	F-CoP[Total]
creation time	2019-05-29 17:20:06
data type	node
minimum value	34.4492%
maximum value	99.5308%
explained variation	94.7949%

FEM Meshes

FEM meshes are unstructured grids in 3D.

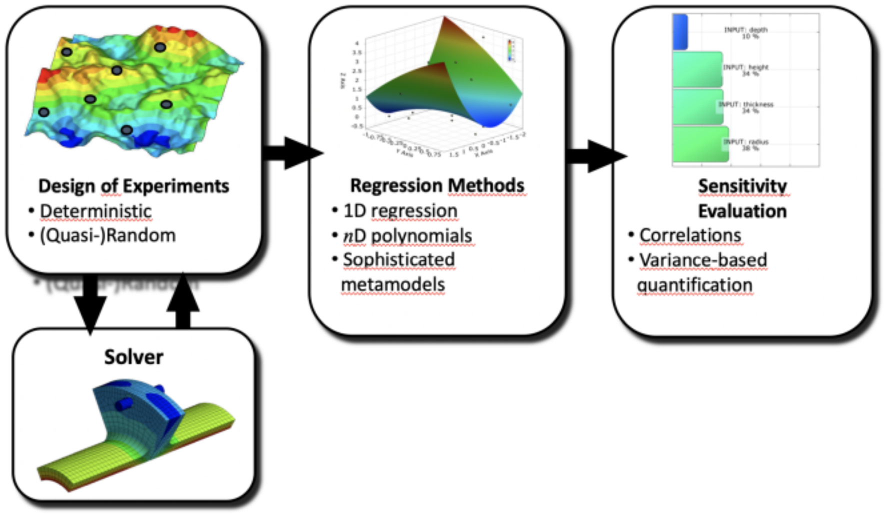


quantity	SEQV
result	sigma[FMOP]
creation time	2019-05-29 14:22:38
data type	node
minimum value	421.254
maximum value	2.04973e+07

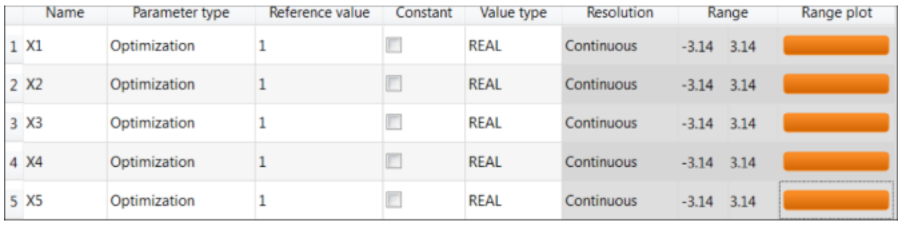
3.2. DOEs and Their Treatment of Field Data

SoS needs to have a definition of the connectivity (for visualization and spatial association), including the type of field data (1D, 2D, 3D, or FEM) and the associated data for each design variation.

Here is the workflow for a DOE:

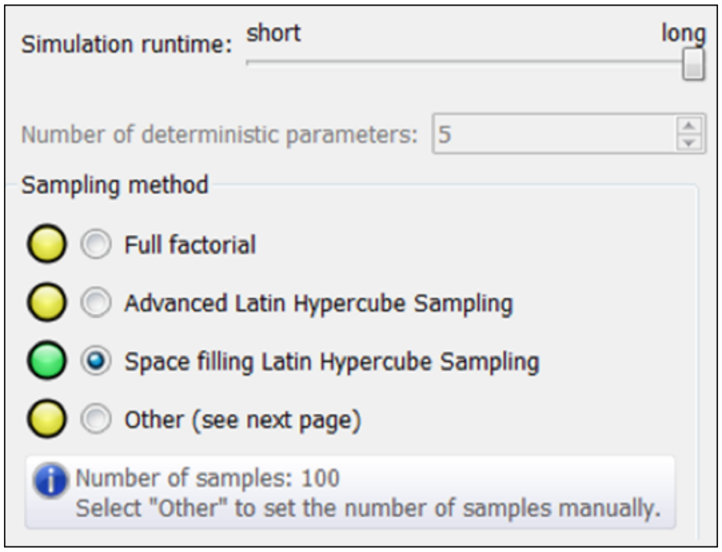


In optiSLang, you start by defining a solver chain and the parameter properties.



	Name	Parameter type	Reference value	Constant	Value type	Resolution	Range		Range plot
1	X1	Optimization	1	☐	REAL	Continuous	-3.14	3.14	🟧
2	X2	Optimization	1	☐	REAL	Continuous	-3.14	3.14	🟧
3	X3	Optimization	1	☐	REAL	Continuous	-3.14	3.14	🟧
4	X4	Optimization	1	☐	REAL	Continuous	-3.14	3.14	🟧
5	X5	Optimization	1	☐	REAL	Continuous	-3.14	3.14	🟧

You then drop the sensitivity wizard on the final solver chain and define the lower and upper bounds of the input variables. optiSLang recommends the sampling method and sample number based on the chosen solver runtime and number of parameters.


Simulation runtime: short long

Number of deterministic parameters: 5

Sampling method

● Full factorial

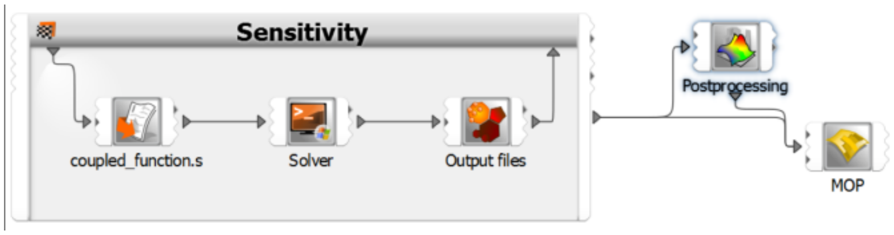
● Advanced Latin Hypercube Sampling

● [✓ Space filling Latin Hypercube Sampling]

● Other (see next page)

Number of samples: 100
Select "Other" to set the number of samples manually.

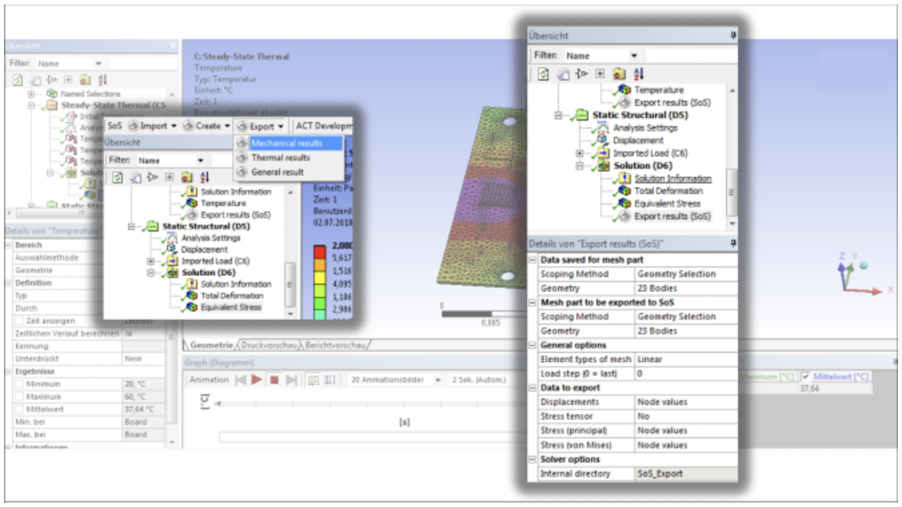
optiSLang then creates a sensitivity system that generates a DOE scheme and executes the corresponding solver runs. The statistical postprocessing calculates correlations for the available designs even while the DOE is running. The MOP is generated within an additional node after the DOE is finished.



When dealing with DOEs, you must define the output files that are produced by the solver:

- ANSYS Mechanical has an ACT extension for SoS that can prepare the 3D output (meshes and data fields).
- ANSYS LS-DYNA has LS-PrePost that can prepare the 3D result data (data fields).
- For signal data, you can use optiSLang's capabilities to import signals from text files into the OMDB database file.
- For all other applications, you must write a converter or use the embedded scripting abilities to produce output that SoS can import.

The following image shows the ACT SoS extension for ANSYS Mechanical:



Details von "Export results (SoS)"

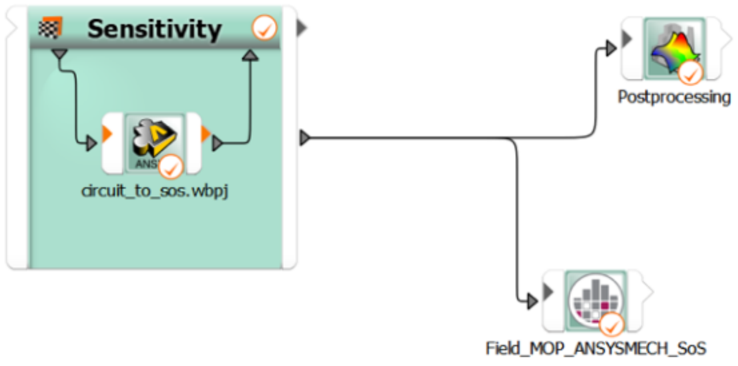
Data saved for mesh part	
Scoping Method	Geometry Selection
Geometry	23 Bodies
Mesh part to be exported to SoS	
Scoping Method	Geometry Selection
Geometry	23 Bodies
General options	
Element types of mesh	Linear
Load step (0 = last)	0
Data to export	
Displacements	Node values
Stress tensor	No
Stress (principal)	Node values
Stress (von Mises)	Node values
Solver options	
Internal directory	SoS_Export

For standard formats, you can use optiSLang integration nodes to import the data and automatically perform a standard analysis. This includes:

- ANSYS Mechanical results if mesh topology was not changed
- 2D performance maps when using encapsulated DOE systems
- Signals in MOP node

For other cases, the SoS standalone GUI must be open and the data must be imported manually.

The following image shows the ANSYS Mechanical SoS integration node in the optiSLang workflow.



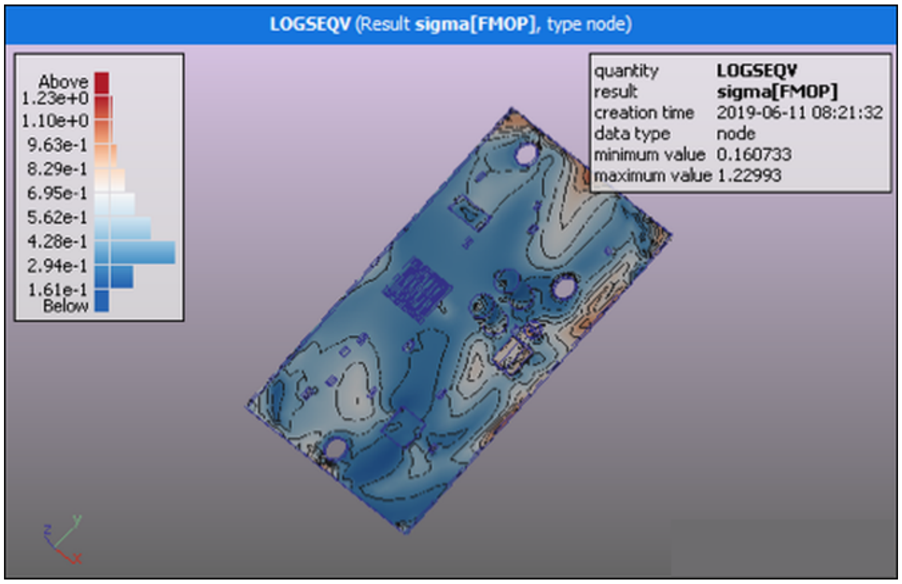
3.3. Statistical Analysis: Quantify Variation in 3D

After importing data, the first step is typically a statistical analysis in 3D to understand the variation magnitudes and patterns of the model performance. SoS provides visualization of properties describing the variation. These statistical quantities include:

- Variation patterns
- Nonlinear sensitivity analysis
- Mean, variance, and standard deviation

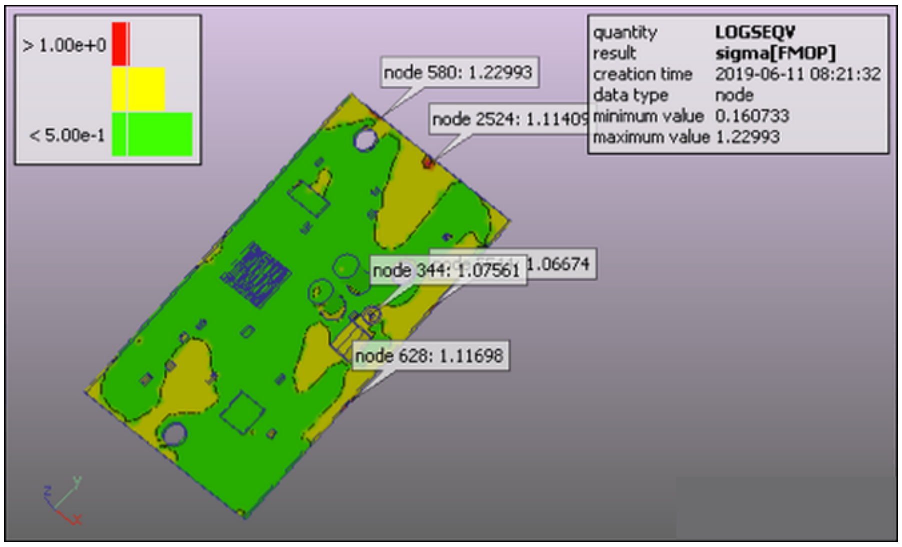
- Linear correlations
- Standard errors
- Quantile values and exceedance probabilities (68-95-99 rule)
- Probability of being the maximum or minimum
- Minimum, maximum, and ranges
- Standard errors of the mean and standard deviation
- Cp and Cpk quality capability values
- Robust Principal Component Analysis (RPCA) modes and outliers

This image shows a 3D plot of a standard deviation of a temperature field:



quantity	LOGSEQV
result	sigma[FMOP]
creation time	2019-06-11 08:21:32
data type	node
minimum value	0.160733
maximum value	1.22993

This image shows a traffic light plot to highlight critical regions:



quantity	LOGSEQV
result	sigma[FMOP]
creation time	2019-06-11 08:21:32
data type	node
minimum value	0.160733
maximum value	1.22993

3.4. Random Fields

Random fields are models that help to find a parametrization for arbitrary data on spatial domains. Typically, these are in:

- 1D: Signal variations (such as time, frequency, and load-displacement curves)
- 2D: Matrix or performance map variations
- 3D: Spatial variations

Random fields can be used as inputs or response to:

- Find automatically a parameter that describes field variations based on measurements or simulation results
- Generate random designs (such a random signals representing the distribution of temperature profiles in time, uncertain load-displacement curves, and random geometries reflecting manufacturing tolerances)
- Predict field responses (meta models for signals and 3D fields)
- Perform sensitivity analysis of distributed quantities (signals and 3D fields)
- Perform statistical smoothening
- Perform data reconstruction

SoS provides a variety of random field models with different properties and application ranges:

Model Type	**Inputs**	**Properties**	**Application**
Empirical random field	Many measurements	Accurate description of reality	Robustness analysis, Quality control, Maintenance
Synthetic random field model	Few measurements	Mean + standard deviation from measurements, spatial correlation from assumption	Robustness analysis: Predict Pf based on true magnitude of variations
Free-form model	Few measurements	Mean + standard deviation from measurements	Needs more parameters but can localize effects on structure
Synthetic random field model	Single/no measurement	Artificial, global	Check if variation has impact
Free-form model	No measurement	Artificial, local	Sensitivity analysis with respect to input region

Example

Objective

Train a random field model based on 60 measured geometries of mandibles

Challenges

Fine meshes and large deflections

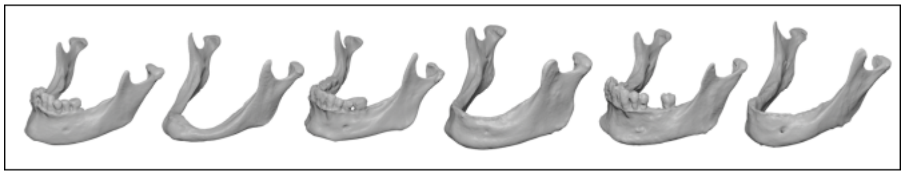
Solution

Delete teeth from model to simplify the geometric variation pattern

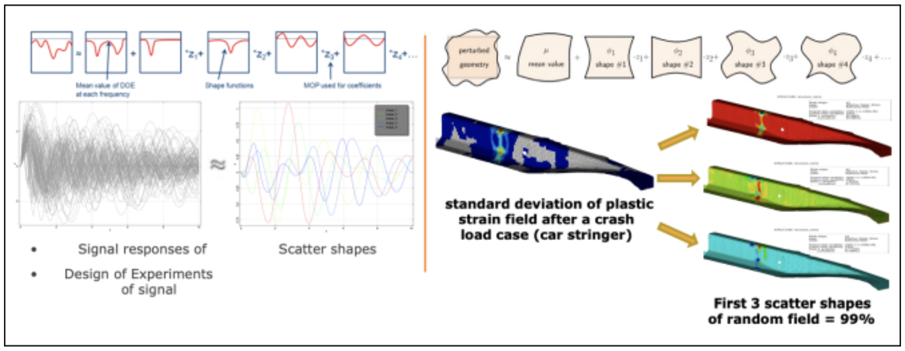
The geometries were prepared using Blender scripting to have the same mesh topology. SoS then compared the x, y, and z coordinates of the surface nodes using a cross-correlated empirical random field model for the x, y, and z coordinates.

The result is a representation of variation patterns by a very small number of parameters. The mesh morphing is supported by mesh stabilization algorithms that reduce the amount of element distortion when applying local deformation vectors. The following images are used with permission from UK Aachen. For more information, see S. Raith, WOST 2015.

This first image shows scans of mandibles as inputs (STL):



This next image shows variation patterns (scatter shapes) and random field schematics. The left side are the identified variation patterns of a DOE of curves. On the right side are simple FEM solutions distributed on a mesh.


Mean value of DOE at each frequency
Shape functions
MOP used for coefficients
Signal responses of
Design of Experiments of signal
Scatter shapes
perturbed geometry
mean value
shape #1
shape #2
shape #3
shape #4
standard deviation of plastic strain field after a crash load case (car stringer)
First 3 scatter shapes of random field = 99%

These variation patterns can be combined with sub-space parameters in a linear combination to represent either the original data (in a *compressed* sub-space) or to generate new field data with a correlation structure based on the training data set. This is known as the Karhunen-Loeve expansion.

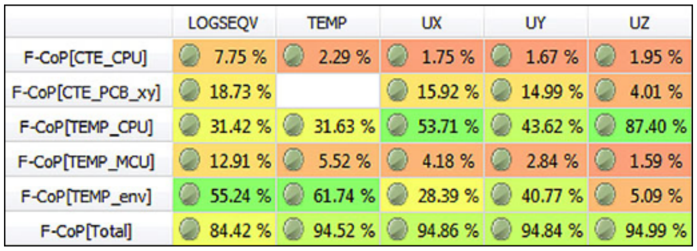
By using random fields, you can use field data as inputs. The random field model translates the field into scalar parameters that can serve as inputs to other algorithms, such as meta models.

3.5. Spatial Sensitivity Analysis and Field-MOP

With a field-MOP, SoS provides a technology for approximating field data by nonlinear approximation functions. The quality of the field-MOP can be accessed easily through the F-CoP, which visualizes in 3D the approximation quality for each respective position on the mesh.

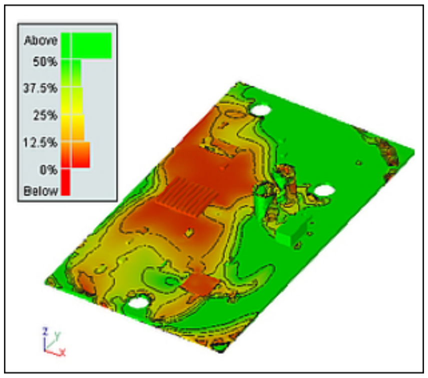
The F-CoP can be represented by:

- A single F-CoP value, which is a *spatially averaged* value of the CoP for the entire field. This image is an F-CoP matrix showing spatially averaged prognosis quality and sensitivity indices:



	LOGSEQV	TEMP	UX	UY	UZ
F-CoP[CTE_CPU]	7.75 %	2.29 %	1.75 %	1.67 %	1.95 %
F-CoP[CTE_PCB_xy]	18.73 %		15.92 %	14.99 %	4.01 %
F-CoP[TEMP_CPU]	31.42 %	31.63 %	53.71 %	43.62 %	87.40 %
F-CoP[TEMP_MCU]	12.91 %	5.52 %	4.18 %	2.84 %	1.59 %
F-CoP[TEMP_env]	55.24 %	61.74 %	28.39 %	40.77 %	5.09 %
F-CoP[Total]	84.42 %	94.52 %	94.86 %	94.84 %	94.99 %

- An F-CoP plot, which is the CoP at the respective position. This image is an F-CoP plot in 3D that highlights the sensitivity of the temperature field with respect to input parameter variations:



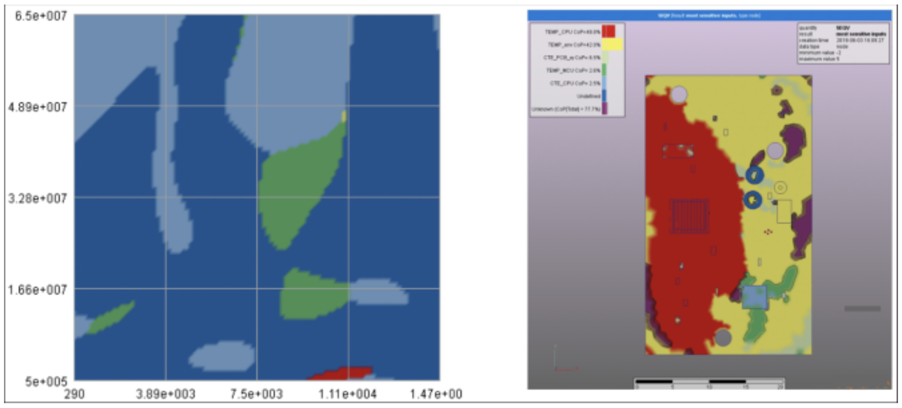
The CoP represents as a percentage the amount of the expected variance that can be explained by the model.

- The F-CoP of the whole model (F-CoP(Total) is the explainable variation at a specific position (prognosis quality).

- The F-CoP for an individual input parameter is the explainable variation at a specific position through the respective parameter.

The field-MOP creates a data-based ROM but also reduces the set of input parameters to the relevant sub-set and orders the remaining inputs according to their importance. The importance (sensitivity) can be measured in terms of an average value or by a spatially variable plot.

To simplify the analysis, you can visualize the most important input for each position. This image shows the most important input plot in 3D, highlighting the input that is most important for a specific region on a FEM mesh (left) and on a performance map (right):


6.5e+007
4.89e+007
3.28e+007
1.66e+007
5e+005
290
3.89e+003
7.5e+003
1.11e+004
1.47e+00
TEMP_CPU
TEMP_env
CTE_PCB_xy
TEMP_MCU
CTE_CPU
Undefined
Unknown

Chapter 4: SoS Examples

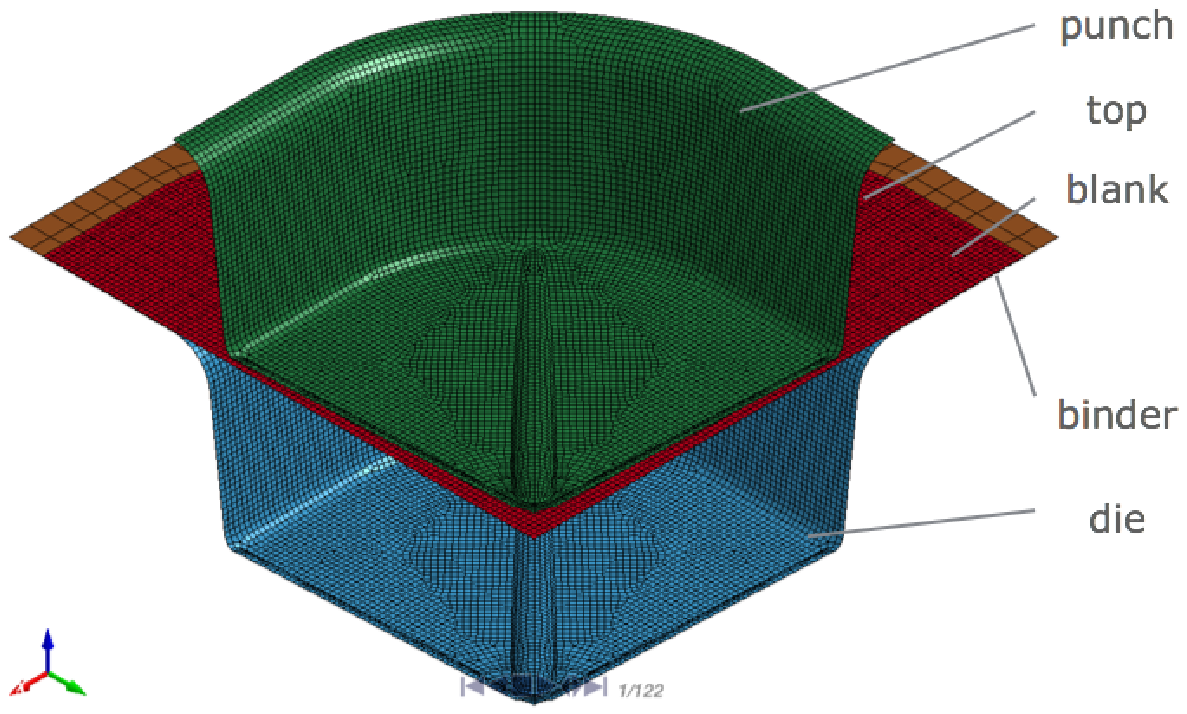
You can gain additional insight into SoS by reviewing these examples:

Note:

The files for these examples are in your public `Documents` folder: `C:\Users\Public\Documents\Dynardo\Statistics on Structures\`*`version`*`\examples`. Paths given herein start at the `examples` folder level. To quickly open the `Examples` folder, you can select **Help** → **Open examples folder**.

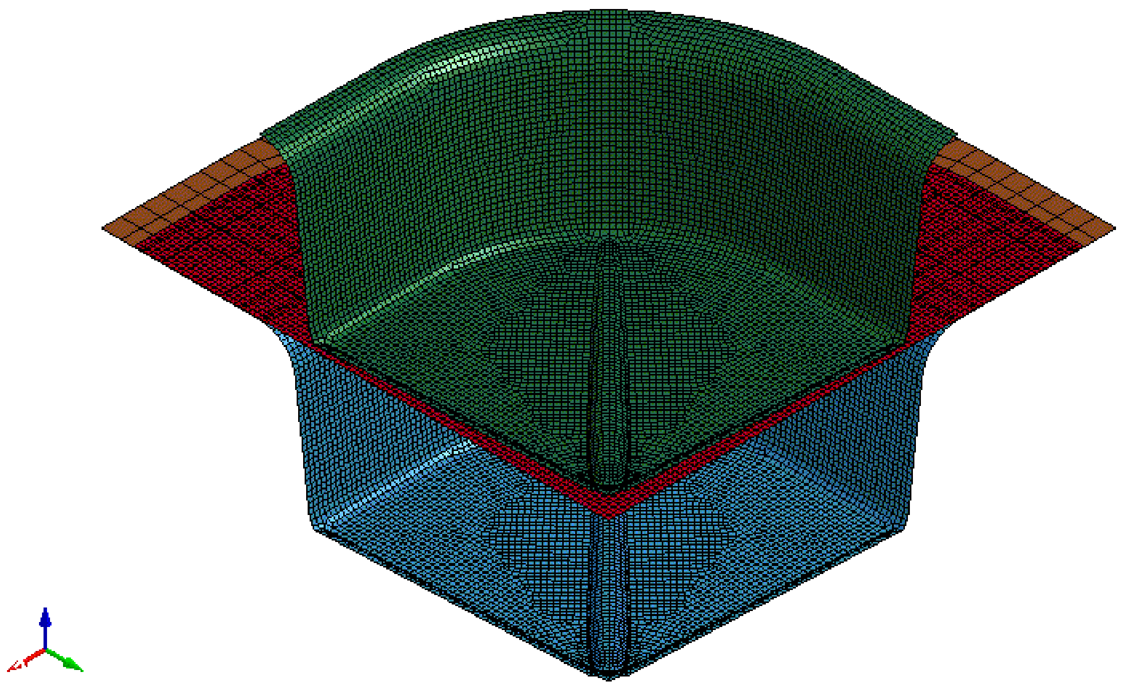
4.1. Deep Drawing

The folder `examples/lsdyna` contains a simple example that simulates the sheet metal forming production process (deep drawing) of a shell structure. This image shows the process setup:



LS-DYNA and LS-PrePost were used to create the numerical model. Due to symmetry of geometry and loading, only a quarter of the structure is modeled. The top is invisible in the figure. During the production process, the punch is subject to vertical motion. Because all bodies except the blank are rigid, the blank is deformed into its desired shape. At the end, the rebound phase is initiated by removing the top and binder and shifting back the punch. The blank rebounds to its final geometry.

The following demo is an animated GIF in the online help. View online if you are reading the PDF version of this guide.



The FEM mesh of the blank contains approximately 10,000 nodes and linear shell elements so that this example can run on even low-end computers. While this simple deep drawing setup is not very realistic, it demonstrates most of the features of SoS within a robustness evaluation.

For the blank material, aluminum is chosen using a bilinear plastic material model. optiSLang was used to evaluate the robustness of the structural design. The simulation of the production process was repeated 100 times, using slightly varied input parameters. For a list of input parameters, see:

- Case A: With Eroded Elements (p. 47)
- Case B: With Varying Punch Velocity (p. 48)

When evaluating the robustness of the process, engineers are interested in random fields that are distributed on the FEM mesh, including the equivalent plastic strain, thickness reduction (thinning), and FLC (forming limit curve) coefficients.

The simulation is not a real-life example. Its main purpose is to demonstrate most of the functionality of SoS as a postprocessor. Simplifying the geometry and process parameters reduces the ability to interpret the data, as shown in Case A: With Eroded Elements (p. 47).

Unless stated otherwise, the directory paths for Case A and Case B are subdirectories in `examples/lsdyna`.

4.1.1. Case A: With Eroded Elements

The software requirements for Case A are:

- Products: SoS, optiSLang (optional), LS-DYNA (optional), and LS-PrePost (optional)
- Platform: Linux or Windows

The folder `metal_forming__eroded_elements` contains the first variation of the previously introduced example. optiSLang is used to evaluate the robustness of the shell structure. For Case A, the following parameters are investigated:

- Material parameters (random numbers)
  - Mass density ρ
  - Elastic modulus E
  - Poisson ratio ν
  - Bilinear σ - ε diagram (0-0.5)
  - Plastic failure (max. plastic strain σ)
- Geometric parameters (random number)
  - Shell thickness t

The results are in the directory `sampling`. This directory contains the respective optiSLang bin file, the reference mesh, and the design directories. The optiSLang bin file contains the design table of the varied input parameters and some scalar output parameters (such as maximum or average strains).

Each design directory contains the subdirectories `included` and `NW`:

- The files in `included` are used to perform the simulation with LS-DYNA.
- The files in `NW` are the alpha-numeric output files being generated by LS-DYNA and LS-PrePost. These files demonstrate the variety of supported LS-DYNA input formats for result files.

To demonstrate most of the functionality of SoS as a postprocessor, eroded data are enforced in this example. All elements with a plastic strain of greater than 0.42 are eliminated during calculation. In total, only 16 designs do not have eroded data. In the tutorials, designs with too many eroded data are marked as statistical outliers and removed from the analysis. Thus, only 64 designs remain for further analysis. When analyzing meta models of optimal prognosis in optiSLang, you should take the upper bound of the plastic strain into account.

The path `export` prepares a consecutive simulation. To demonstrate some of the export features of SoS, a robustness analysis was prepared, where each design directory contains a mesh definition. SoS can then be used to export generated (simulated) or smoothened (from a previous analysis) random geometries into these design directories.

4.1.2. Case B: With Varying Punch Velocity

The folder `metal_forming` contains a slight variation of the previously described example. In practice, eroding has been replaced by varying punch velocity and friction. For Case B, the following parameters are investigated:

- Material parameters (random numbers)
  - Mass density ρ
  - Elastic modulus E
  - Poisson ratio ν
  - Bilinear σ - ε diagram (0-0.5)
  - Friction μ between blank and binder
- Geometric parameters (random numbers)
  - Shell thickness t
- Simulation parameters (random numbers)
  - Punch velocity

For Case B, all setup files and some of the output files are delivered.

To demonstrate the *non-matching meshes* feature, two additional meshes are found next to the reference mesh `reference_mesh.k` used for the simulation process. The file `reference_mesh_coarse.k` contains a reduced FEM mesh with only 2304 elements, while the file `reference_mesh_fine.k` contains a finer FEM mesh with 38139 elements.

Note:

Using a different reference mesh significantly influences the time needed to import all designs.

Unless stated otherwise, directory paths for Case B are subdirectories in `examples\lsdyna\metal_forming`.

Robustness Analysis

The software requirements for the robustness analysis are:

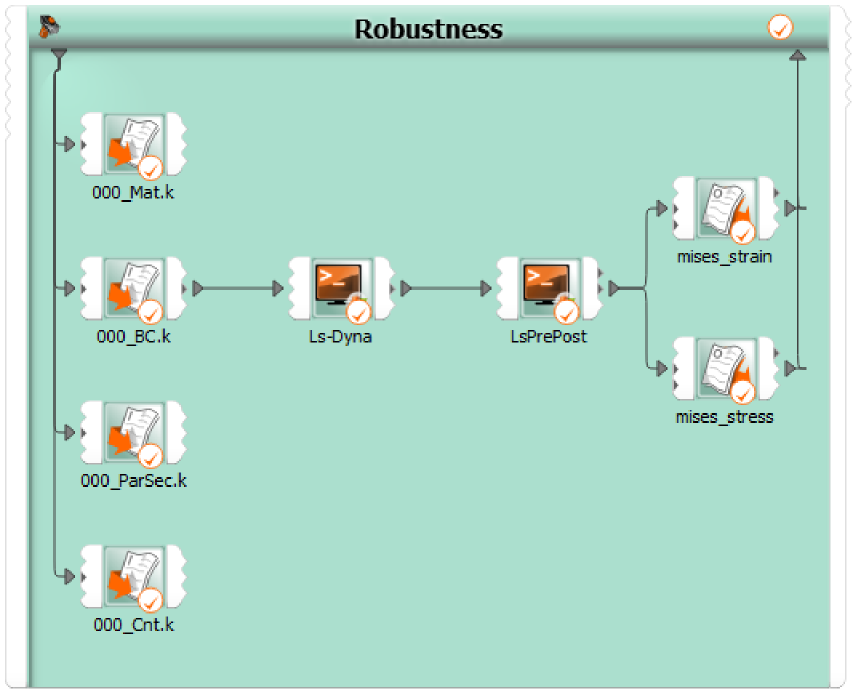
- Products: SoS, optiSLang (optional), LS-DYNA (optional), and LS-PrePost (optional)

- Platform: Windows

The optiSLang project file `deep_drawing.opf` and its project working directory `deep_drawing.opd` contain the predefined Robustness workflow used to generate this example. It has been created and successfully tested on Windows with LS-LS-DYNA and LS-PrePost. All alpha-numerical output files generated by LS-PrePost are in the project working directory in `Robustness/Design0*/out/*.k`. The file `plastic_strain.k` for Design0001 is in:

```
examples/lsdyna/meta_forming/deep_drawing.opd/Robustness/Design0001/out/plastic_st
```

A diagram follows of the optiSLang workflow for a robustness analysis:



When re-executing or even changing this optiSLang project, keep the following information in mind:

- All input files used for this project are in the directory `deep_drawing_template`. Any file mentioned in subsequent bullets can be found in this directory or one of its subdirectories.
- The text input parser definition works as long as it is not modified. Otherwise, the file path location must be updated.
- The batch script actor named LS-DYNA executes a predefined script building on the following prerequisites:
  - The actor copies several files before executing the script. The file path locations must be updated.
  - The actor relies on an environment variable named LSTC_EXECUTABLE defined within this actor. This environment variable must point to the LS-DYNA executable. It is used to generate the following command call, written in line number 16 in the predefined script, assuming four available CPUs:

    ```
    "%LSTC_EXECUTABLE%" I=000_Main.k NCPU= 4 > console.out
    ```

This behavior can be changed by modifying the definition set **`LSTC_NUM_CORES=4`** in line number 7, omitting any space between the equality sign and the number of CPUs.

Managing the LS-DYNA license is your responsibility. For more information, see the ANSYS LS-DYNA User's Guide or the LSTC home page.

- The batch script actor named `Ls-PrePost` executes a predefined Windows batch script, building on the following prerequisites:
  - The actor copies the file `deep_drawing_template\generate_output.cfile` before executing the predefined script. The file path must be updated.
  - The actor relies on an environment variable named `LSPP_EXE_DIR` that is defined within this actor. This environment variable must point to the directory of the LS-PrePost executable. It is used to generate the following command call, written in line number 17 in the predefined script:

```
"%LPP_EXE_DIR%\\lsprepost.exe" c=generate_output.cfile
```

Field-MOP Analysis

The software requirements for a field-MOP analysis are:

- Product: SoS
- Platform: Linux or Windows

To perform field-MOP analysis:

1. Start a new SoS project and import the file `DeepDrawing_fineRef.sdb` from `examples root directoy`.

   This database was generated using the finest reference mesh in `deep_drawing.opd/Robustness/reference_mesh_fine.k` with an *Incompatible (assuming a smooth boundary)* mesh mapper, a search distance given by 5, and a *double-sided, node-to-segment* projection.

2. Load the simulation input parameters into this SoS instance.

   Using the option Import scalars from optiSLang .bin (p. 65), import the optiSLang postprocessor database file in `deep_drawing.opd/Robustness/Robustness_osl3.bin`

3. To complete the field-MOP setup:

   a. Select all scalar input parameters or a subset of interest, although the latter is not generally recommended.

   b. Select one of the random field quantities, such as **`pstrain`**.

4. Use the option Create Field-MOP (p. 96) to create the field-MOP.

The field-MOP is automatically generated in the background and results are shown:

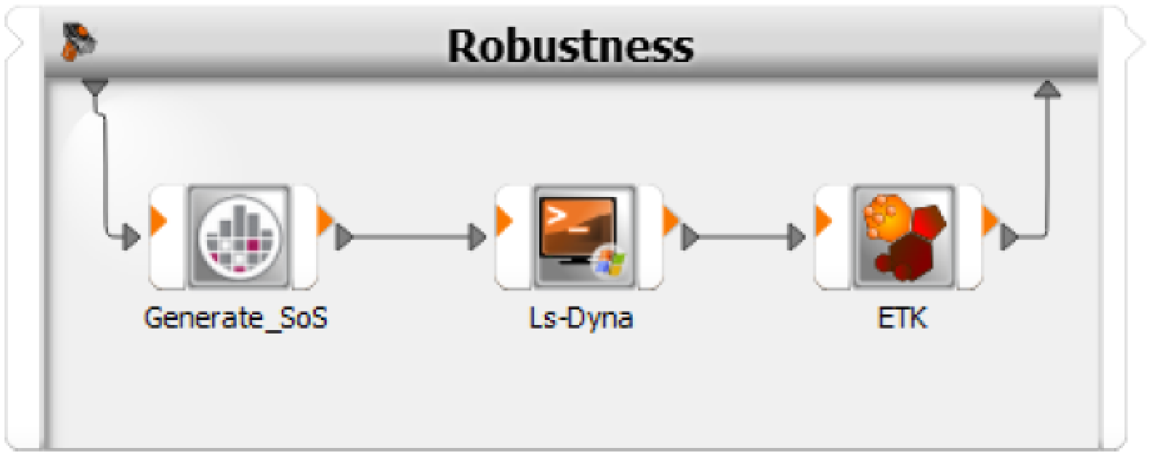
To visualize these F-CoP values only, you can open the file `DeepDrawing_fineRef__FMOP.sdb`, which has already been prepared, following the given procedure. For an illustrated step-by-step guide and more information, see the appropriate SoS tutorial and the description in Creating Field-MOPs (p. 96).

Random Field Generation

The software requirements for random field generation are:

- Product: SoS, optiSLang (optional), LS-DYNA (optional), and LS-PrePost (optional)
- Platform: Linux or Windows

The following diagram shows the optiSLang workflow for generating random fields:



This optiSLang project prepares a consecutive simulation as described in Case A: With Eroded Elements (p. 47). When using this optiSLang, keep in mind the following:

- For information on using the **`Generate_SoS`** node for random field simulation, in the *Statistics on Structures Tutorials*, see Using the "Generate_SoS" Node in optiSLang.
- The batch script actor named `LS-DYNA` executes a predefined script. This actor defines and relies on an environment variable named LSTC_EXECUTABLE. This environment variable must point to the LS-DYNA executable. It is used to generate the following command call, written in line number 16 in the predefined script assuming four available CPUs:

  ```
  "%LSTC_EXECUTABLE%" I=000_Main.k NCPU= 4 > console.out
  ```

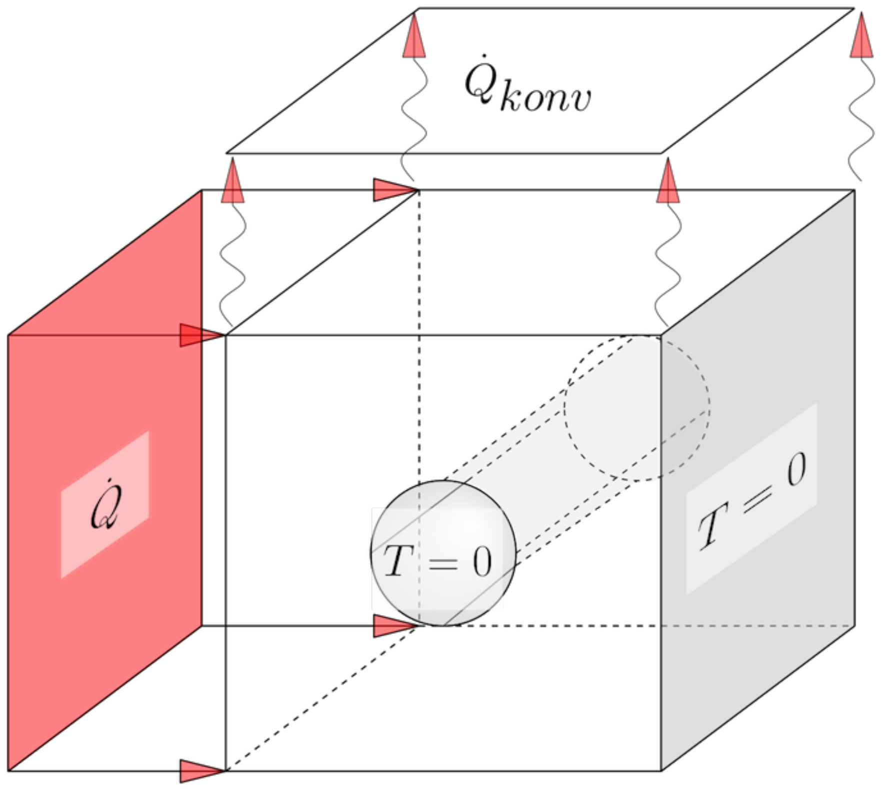
  You can change this behavior by modifying the definition set **`LSTC_NUM_CORES=4`** in line number 7, omitting any space between the equal sign and the number of CPUs.

  Managing the LS-DYNA license is your responsibility. For more information, see the ANSYS LS-DYNA User's Guide or the LSTC home page.

4.2. Heat Conduction

The folder `examples/slangtng/heat_conduction` contains a simple thermal example. Unless stated otherwise, the directory paths for this example are all subdirectories in this folder.

This example simulates the steady-state heat conduction process of a solid structure. Here is the simulation setup:



The numerical model was created using gmsh as the grid generator and slangTNG as the solver and mesh format converter. The simulation model is subject to a given temperature T=0 on the right surface as well as the hole. On the left surface, a rate of heat flow is given with a determined convection coefficient on the top. All other sides are free from boundary conditions, or in terms of heat conduction, perfectly isolated.

The FEM mesh of the cube contains 55.653 linear tetrahedra, given in the file `heat_conduction_template/block2.msh`. The thermal analysis given in the file `heat_conduction_template/thermal.tng` is readily solved by slangTNG.

While this example is not sophisticated, it demonstrates most of the 3D features of SoS.

Sensitivity Analysis

The software requirements for the sensitivity analysis are:

- Product: SoS, optiSLang (optional), slangTNG (optional), and gmsh (optional)
- Platform: Windows

optiSLang was used for the sensitivity study of the heat conduction. The simulation of the heat flaw was repeated 100 times using slightly varied input parameters. The following thermal parameters (random numbers) were subject to investigations:

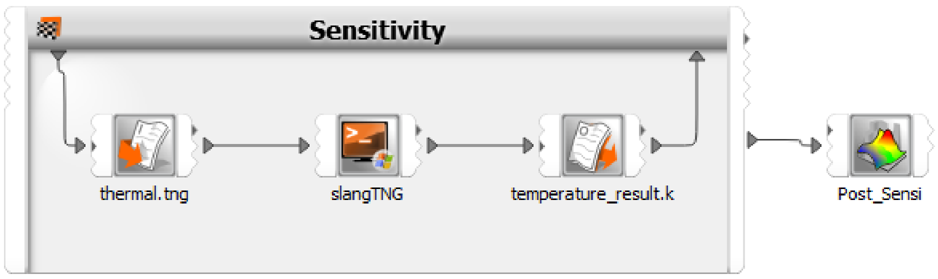
- Rate of heat flow **`heat`**
- Convection coefficient **`conv`**

The optiSLang project file `heat_conduction.opf` contains the predefined Sensitivity workflow used to generate this example. It has been created and successfully tested on Windows 7 SP1 with slangTNG. Its working directory `heat_conduction.opd/Sensitivity` contains the respective optiSLang bin file, the reference mesh, and the design directories. The optiSLang bin file contains the design table of the varied input parameters. Each design directory contains the slangTNG problem definition file `thermal.tng` and the spatially distributed temperatures result file `temperature_result.k` in LS-PrePost syntax generated by slangTNG.

For this example, the file `temperature_result.k` for Design0001 is in:

```
examples/slangtng/heat_conduction/heat_conduction.opd/Sensitivity/Design0001/tempera
```

A diagram follows of the optiSLang workflow for a sensitivity analysis:



When re-executing or even changing this optiSLang project, keep the following information in mind:

- All input files used for this project are in the directory `heat_conduction_template`. Any file mentioned in subsequent bullets is also in this directory.
- The text input parser definition works as long as it is not modified. Otherwise, the file path location for `thermal.tng` must be updated.
- The Batch Script actor named slangTNG executes a predefined script building on the following prerequisites:
  - The actor copies the file `block2.msh` before executing the script. The file path location must be updated.
  - The actor relies on an environment variable named `SLANGTNG_EXECUTABLE` defined within this actor. This environment variable must point to the slangTNG executable.

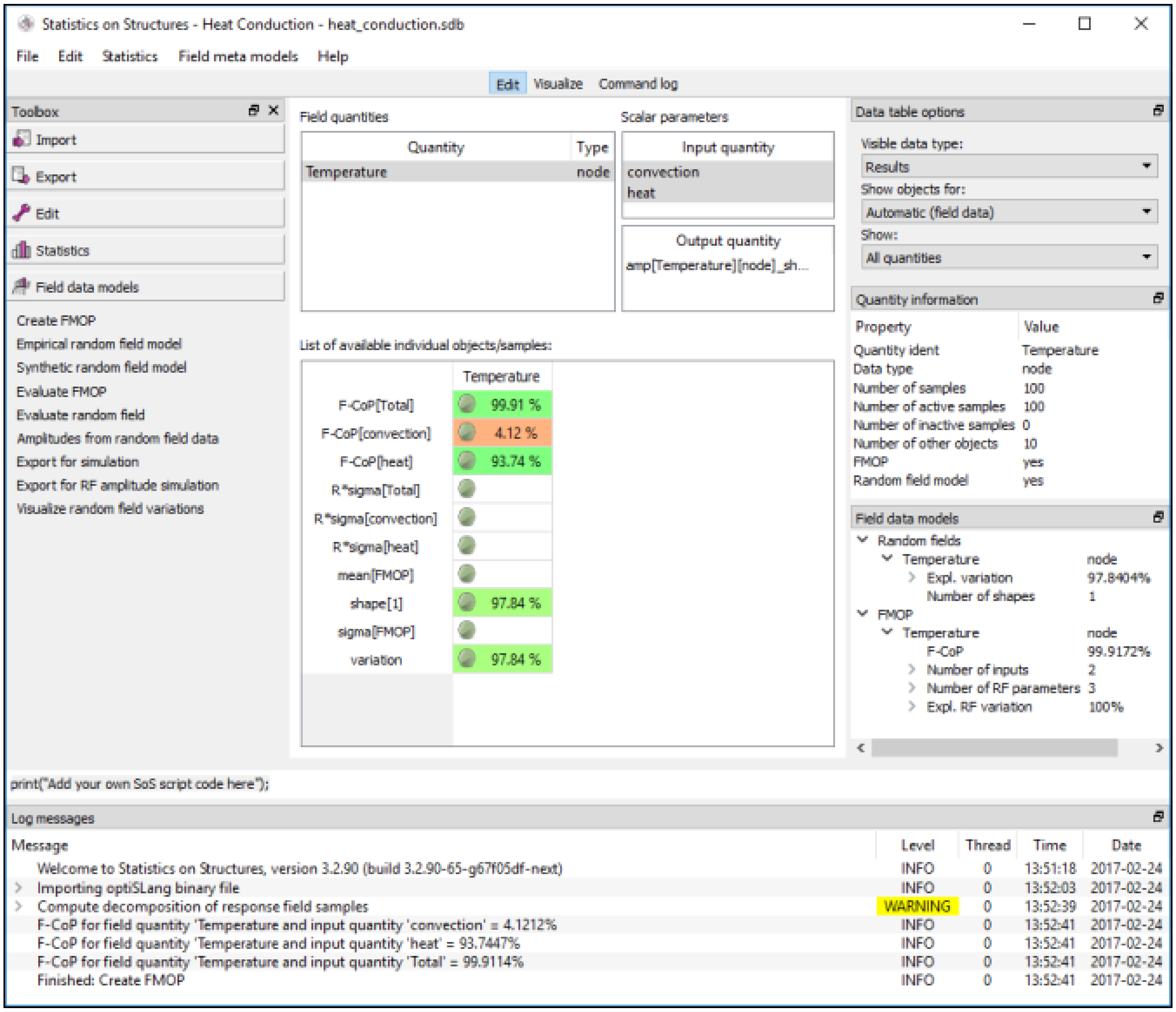
Field-MOP Analysis

The software requirements for a field-MOP analysis are:

- Product: SoS

- Platform: Linux or Windows

To compute the field-MOP for this example, you follow the same basic procedure given in Case B: With Varying Punch Velocity (p. 48). To start the field-MOP analysis, you open the `heat_conduction.sdb` file, where you find the prepared random field for the single quantity `Temperature`. The optiSLang postprocessor database file needed as input to the field-MOP is in `heat_conduction.opd/Sensitivity/Sensitivity_osl3.bin`


Statistics on Structures - Heat Conduction - heat_conduction.sdb
File
Edit
Statistics
Field meta models
Help
Edit
Visualize
Command log
Toolbox
Import
Export
Edit
Statistics
Field data models
Create FMOP
Empirical random field model
Synthetic random field model
Evaluate FMOP
Evaluate random field
Amplitudes from random field data
Export for simulation
Export for RF amplitude simulation
Visualize random field variations

Field quantities

Quantity	Type
Temperature	node

Scalar parameters

Input quantity
convection
heat

Output quantity
amp[Temperature][node]_sh...

List of available individual objects/samples:

	Temperature
F-CoP[Total]	● 99.91 %
F-CoP[convection]	● 4.12 %
F-CoP[heat]	● 93.74 %
R^2sigma[Total]	●
R^2sigma[convection]	●
R^2sigma[heat]	●
mean[FMOP]	●
shape[1]	● 97.84 %
sigma[FMOP]	●
variation	● 97.84 %

Data table options

Visible data type: Results

Show objects for: Automatic (field data)

Show: All quantities

Quantity information

Property	Value
Quantity ident	Temperature
Data type	node
Number of samples	100
Number of active samples	100
Number of inactive samples	0
Number of other objects	10
FMOP	yes
Random field model	yes

Field data models

Random fields	
Temperature	node
Expl. variation	97.8404%
Number of shapes	1
FMOP	
Temperature	node
F-CoP	99.9172%
Number of inputs	2
Number of RF parameters	3
Expl. RF variation	100%

```
print("Add your own SoS script code here");
```

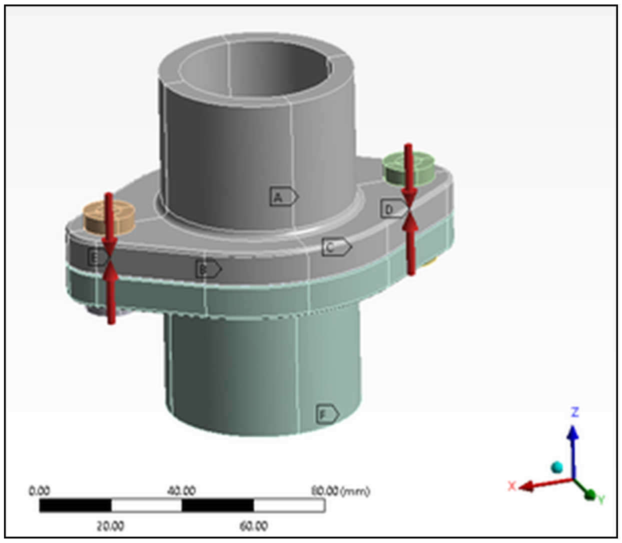
Log messages

```
Message                                                                          Level    Thread  Time      Date
Welcome to Statistics on Structures, version 3.2.90 (build 3.2.90-65-g67f05df-next)  INFO     0       13:51:18  2017-02-24
Importing optiSLang binary file                                                  INFO     0       13:52:03  2017-02-24
Compute decomposition of response field samples                                  WARNING  0       13:52:39  2017-02-24
F-CoP for field quantity 'Temperature and input quantity 'convection' = 4.1212%  INFO     0       13:52:41  2017-02-24
F-CoP for field quantity 'Temperature and input quantity 'heat' = 93.7447%       INFO     0       13:52:41  2017-02-24
F-CoP for field quantity 'Temperature and input quantity 'Total' = 99.9114%      INFO     0       13:52:41  2017-02-24
Finished: Create FMOP                                                            INFO     0       13:52:41  2017-02-24
```

If you want to visualize the F-CoP values only, you can open the file `heat_conduction__FMOP.sdb`, which has been already prepared. For an illustrated step-by-step guide and more information, see the appropriate SoS tutorial and the description in Creating Field-MOPs (p. 96).

4.3. Gasket Tightness

The folder `examples/ansys/gasket` contains an example for evaluating whether a gasket is tight given a certain bolt pretension and pressure of the flowing medium.

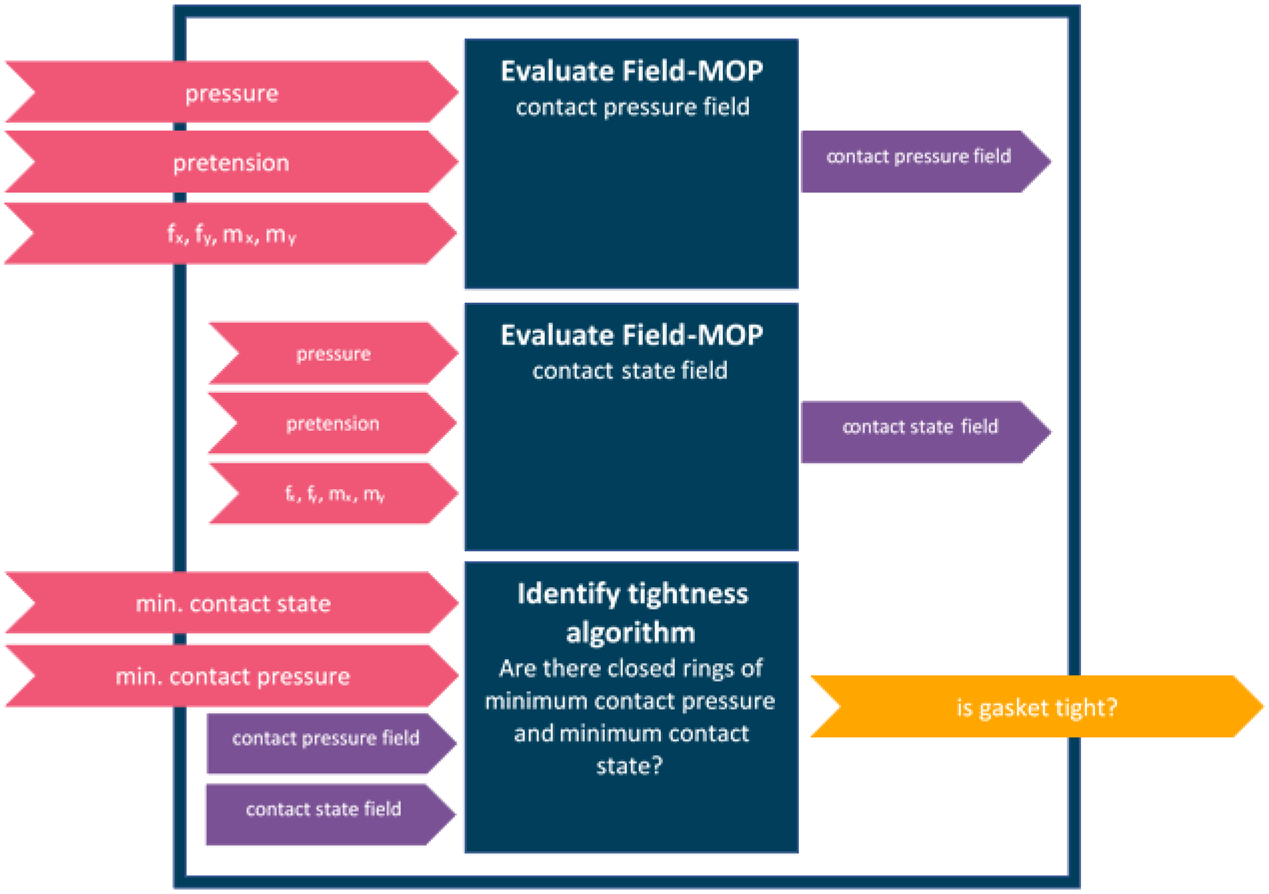


The SoS analysis is based on simulation data from ANSYS Mechanical. It demonstrates how multiple field-MOPs and custom algorithms can be combined in a single SoS analysis to return a meaningful scalar output.

The ultimate goal is to quickly determine the gasket's tightness when given new input parameters, without running a computationally expensive simulation. The SoS analysis consists of the evaluation of two field-MOPs and a sophisticated algorithm for assessing whether the flowing medium remains contained.

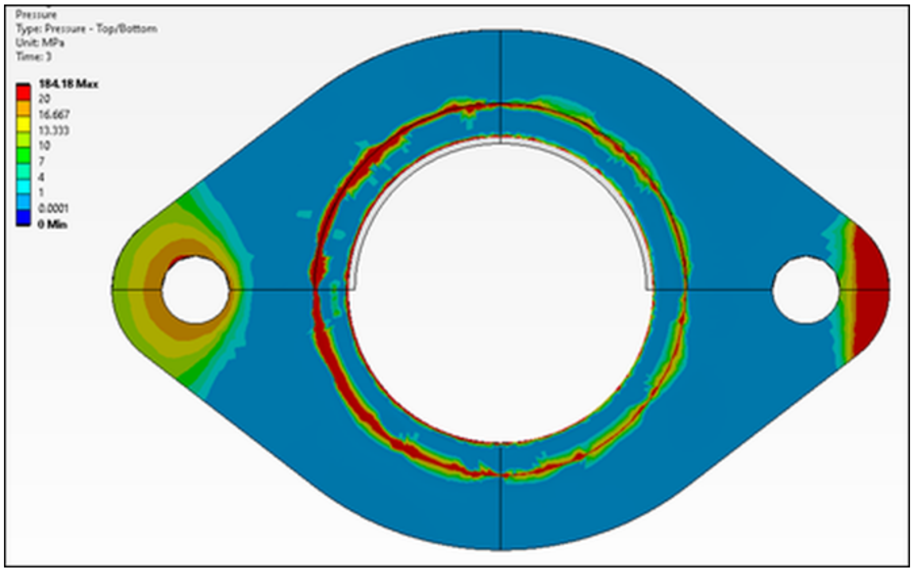
The SoS analysis is then exported to an FMU 2.0 file. This file takes the scalar input parameter and determines whether the gasket remains tight. The exported FMU2.0 file can then be consumed in supporting tools, such as ANSYS optiSLang and ANSYS Twin Builder.

The following image shows the three steps in the SoS analysis. The inputs and outputs that enter and leave the frame are scalar values exposed by the FMU 2.0 file.



The flowing medium's pressure acts against the contact pressure field between the lower and upper gasket part. The medium can escape only if it finds a continuous path where either or both of these conditions are met:

- The two gasket parts are not in close contact.
- The contact pressure field's magnitude is below a certain threshold.



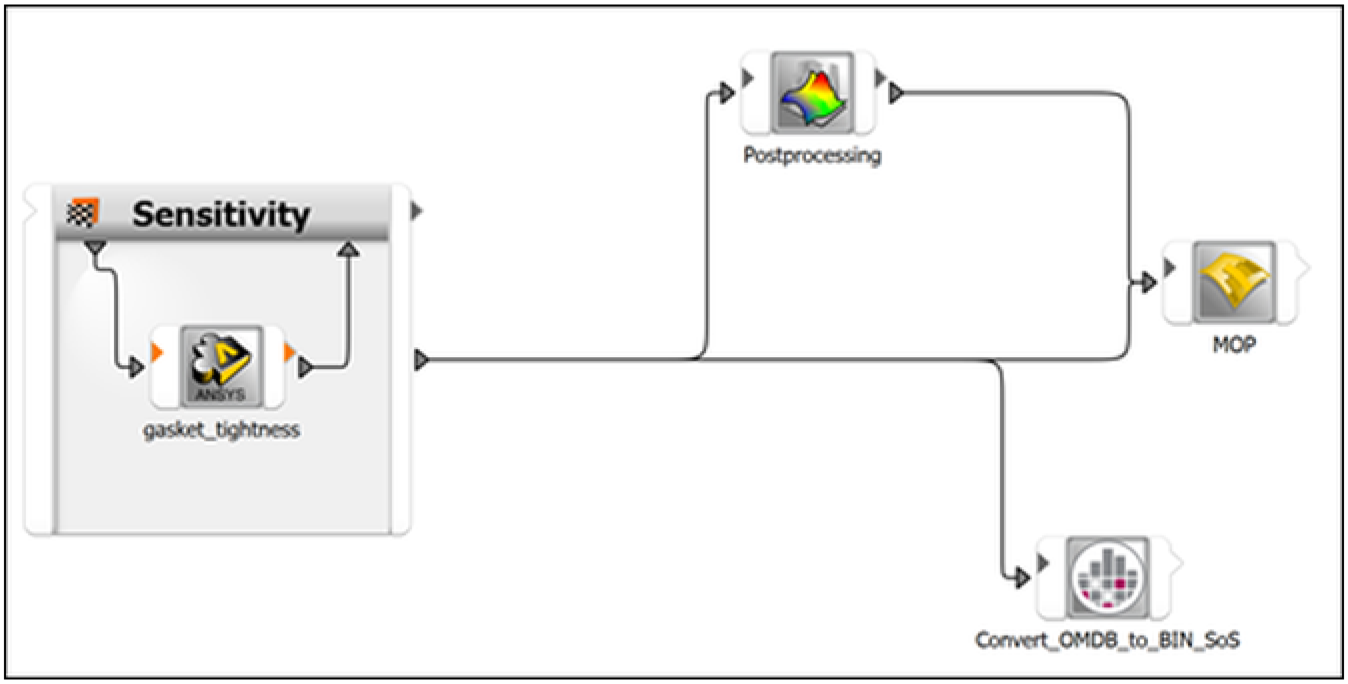
The contact pressure field forms a closed ring with a magnitude of at least 6 MPa. The flowing medium cannot escape, if additionally, the upper and lower gasket parts form a similar closed ring of close contact.

Thus, to calculate the desired Boolean output variable (whether the gasket is tight), two fields must be considered:

- Contact state field (close contact/open contact)
- Contact pressure field

ANSYS Mechanical is used to simulate these two fields for a given set of input parameters.

First, ANSYS optiSLang runs a design of experiments (DOE), varying the input parameters and exporting the simulated field data. This image shows the DOE workflow in optiSLang.



Subsequently, in SoS, field-MOPs are created for both fields.

4.3.1. Gasket Model Description

The gasket's simulation is set up in ANSYS Workbench, with ANSYS Mechanical as a solver. The output fields calculated with Mechanical's contact tool are exported using the SoS ACT plugin (p. 149).

The gasket's lower part is assumed to be fixed in place. Force and angular momentum vectors act on the gasket's upper part. Think of wind, snow, or ice loads and vibrations acting on an installation.

Scalar input parameters:

- Pressure of medium flowing through the gasket (8 - 12 MPa)
- Gasket bolt pretension (4.5 – 5.5 kN)

- f_x, f_y: forces acting on top part (0.3 – 0.6 kN)
- m_x, m_y: angular momenta acting on top part (0.036 – 0.044 kNm)

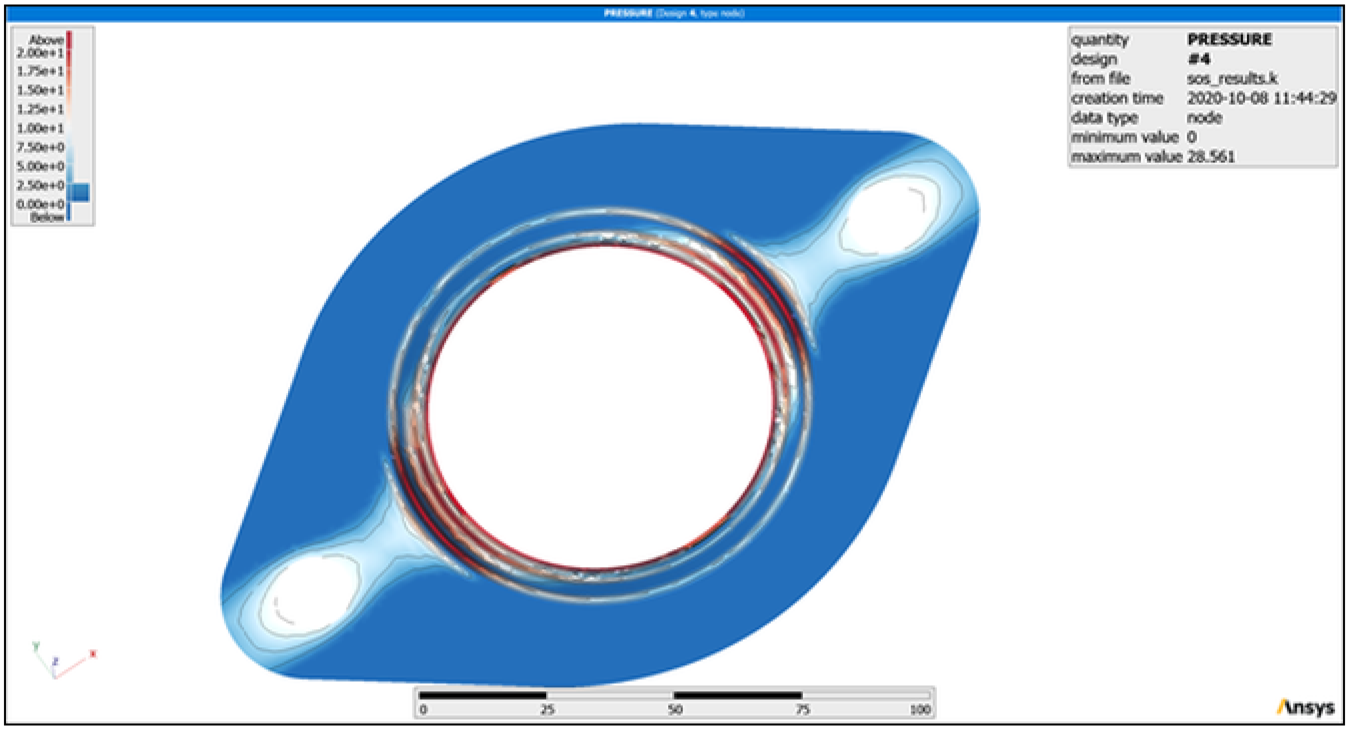
4.3.2. Gasket SoS Analysis and FMU 2.0 File Export

These sections describe the SoS workflow:

4.3.2.1. Importing DOE Simulation Data

Running a DOE with Ansys optiSLang obtains 45 designs, each consisting of a set of scalar input parameters, the contact pressure field, and contact state field.

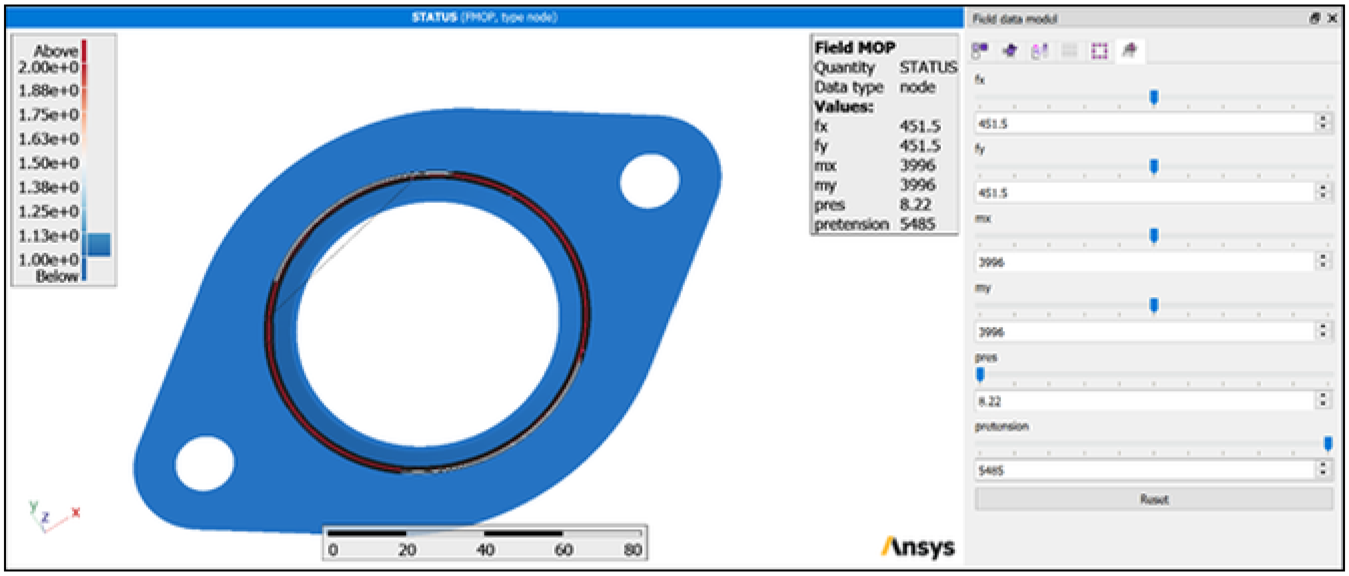
The following image shows the simulated contact pressure field for design 4.



4.3.2.2. Creating Field-MOPs

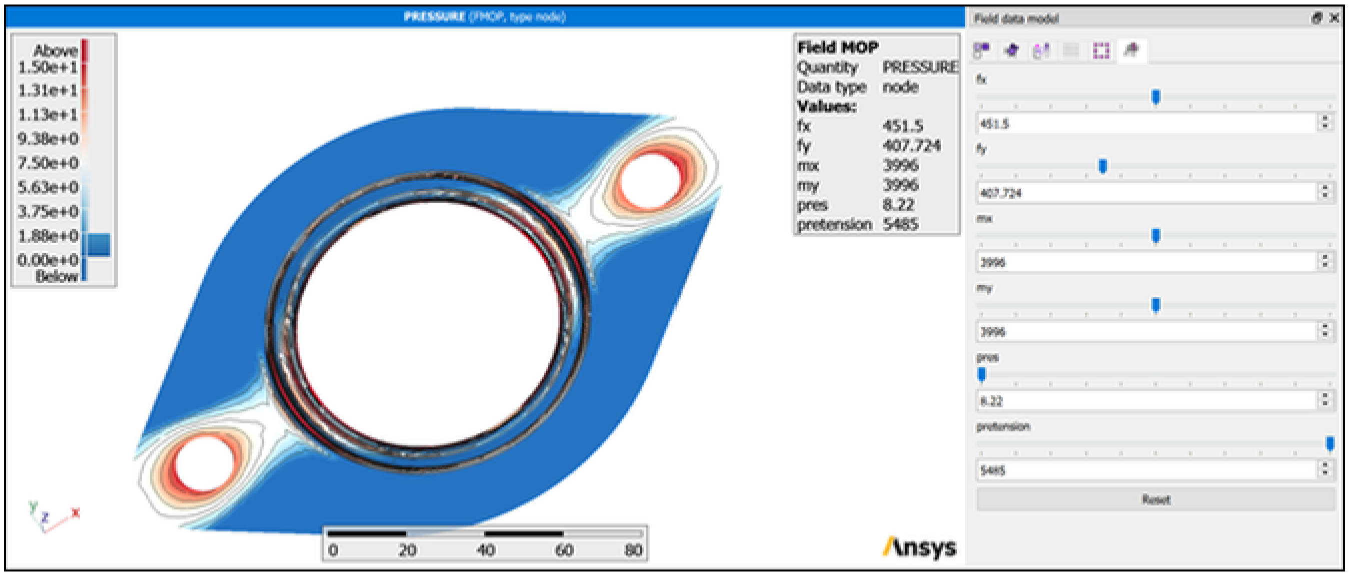
Using the imported scalar input parameters and field designs as training data, two field-MOPs are created within seconds.

This image shows the field-MOP with the contact state field as the output:



Field MOP	
Quantity	STATUS
Data type	node
Values:	
fx	451.5
fy	451.5
mx	3996
my	3996
pres	8.22
pretension	5485

This next image shows the field-MOP with the contact pressure field as the output:



Field MOP	
Quantity	PRESSURE
Data type	node
Values:	
fx	451.5
fy	407.724
mx	3996
my	3996
pres	8.22
pretension	5485

Note:

Using the sliders to change the scalar input parameters provides instant results.

4.3.2.3. Building the Custom SoS Analysis and Exporting it to an FMU 2.0 File

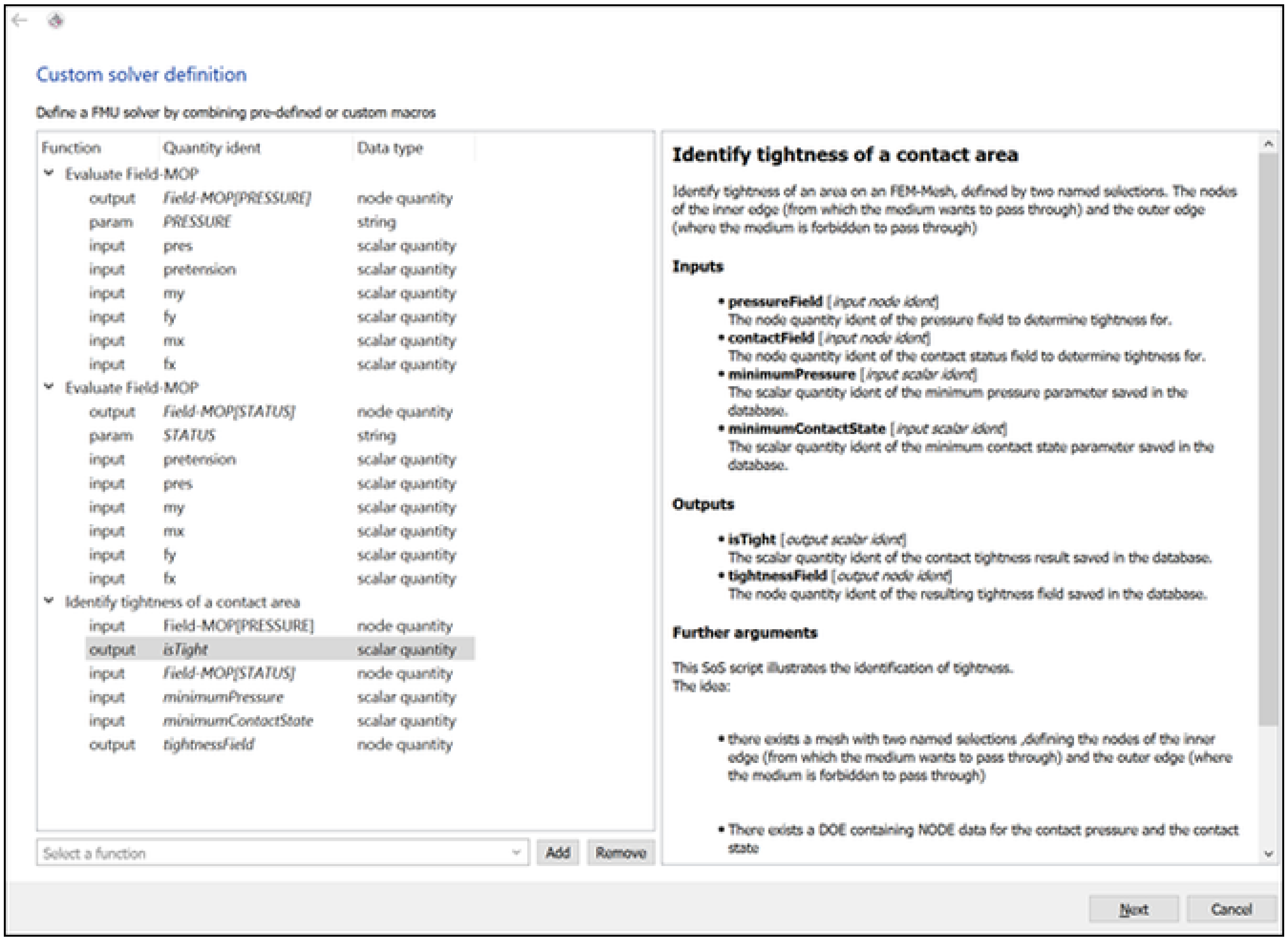
A custom solver is created and exported to an FMU2.0 file.

SoS allows you to interactively combine the three analysis steps:

1. Evaluate the contact pressure field-MOP.
2. Evaluate the contact state field-MOP.

3. Run the algorithm for identifying tightness.

SoS tracks input and output parameters. This image shows the definition for the custom solver:



Custom solver definition

Define a FMU solver by combining pre-defined or custom macros

Function	Quantity ident	Data type
Evaluate Field-MOP		
output	*Field-MOP[PRESSURE]*	node quantity
param	*PRESSURE*	string
input	pres	scalar quantity
input	pretension	scalar quantity
input	my	scalar quantity
input	fy	scalar quantity
input	mx	scalar quantity
input	fx	scalar quantity
Evaluate Field-MOP		
output	*Field-MOP[STATUS]*	node quantity
param	*STATUS*	string
input	pretension	scalar quantity
input	pres	scalar quantity
input	my	scalar quantity
input	mx	scalar quantity
input	fy	scalar quantity
input	fx	scalar quantity
Identify tightness of a contact area		
input	Field-MOP[PRESSURE]	node quantity
output	*isTight*	scalar quantity
input	*Field-MOP[STATUS]*	node quantity
input	*minimumPressure*	scalar quantity
input	*minimumContactState*	scalar quantity
output	*tightnessField*	node quantity

Identify tightness of a contact area

Identify tightness of an area on an FEM-Mesh, defined by two named selections. The nodes of the inner edge (from which the medium wants to pass through) and the outer edge (where the medium is forbidden to pass through)

Inputs

- **pressureField** [*input node ident*]
  The node quantity ident of the pressure field to determine tightness for.
- **contactField** [*input node ident*]
  The node quantity ident of the contact status field to determine tightness for.
- **minimumPressure** [*input scalar ident*]
  The scalar quantity ident of the minimum pressure parameter saved in the database.
- **minimumContactState** [*input scalar ident*]
  The scalar quantity ident of the minimum contact state parameter saved in the database.

Outputs

- **isTight** [*output scalar ident*]
  The scalar quantity ident of the contact tightness result saved in the database.
- **tightnessField** [*output node ident*]
  The node quantity ident of the resulting tightness field saved in the database.

Further arguments

This SoS script illustrates the identification of tightness.
The idea:

- there exists a mesh with two named selections ,defining the nodes of the inner edge (from which the medium wants to pass through) and the outer edge (where the medium is forbidden to pass through)

- There exists a DOE containing NODE data for the contact pressure and the contact state

Chapter 5: SoS File and Data Management

Using options on the **File** menu, you can manage SoS files and data:

Note:

The **Toolbox** window provides quick access to **File** menu options under these functional category headings: **Import** and **Export**. The one exception is the option for managing macros (p. 76), which is under the functional category heading **Field data models**. For more information on this docked window, see Toolbox Window (p. 32).

5.1. SoS Database Files

From the **File** menu, you can load and save an SoS database:

5.1.1. Loading a Database

The **File** menu option **Load database** overwrites the current contents of the SoS database with the contents from a specified single binary file (`*.sdb`). It loads and eventually overwrites the reference mesh and all data objects, including samples and results. Furthermore, all scenes in the 3D visualization are erased, leaving only the new reference mesh.

5.1.2. Saving a Database

The **File** menu option **Save database** saves the current contents of the SoS database to a single binary file (`*.sdb`). This portable file contains the reference mesh and all data objects, including samples and results.

5.1.3. Saving a Database to a New File

The **File** menu option **Save database as** saves the current contents of the SoS database to a new single binary file (`*.sdb`) in the specified location. This portable file contains the reference mesh and all data objects, including samples and results.

5.2. Mesh Files

From the **File** → **Mesh** menu, you can perform mesh-related tasks:

5.2.1. Setting the Reference Mesh

The **File** menu option **Mesh > Set reference mesh** defines the reference mesh, which is read from a single file. For a list of supported mesh file formats, see File Formats Overview (p. 109).

Generally, a reference mesh must be set before further commands can be called.

5.2.2. Importing a Named Selection from a CSV File

A *named selection* describes a subset of FEM nodes or elements. The **File** menu option **Mesh > Import named selection from CSV** imports a named selection from a CSV file. You set a name for your selection, the selection type, and the file from which to import.

Note:

The CSV file must consist of a single column of node or element identifiers. Tabs must be avoided.

Alternatively, you can import a named selection through the reference mesh file `sec_file_mesh_import_reference_mesh`.

.

5.2.3. Setting a Reference Named Selection

Field quantities are defined on a unique reference name selection. You can use the entire reference mesh or only parts of the reference mesh to form a reference named selection.

The **File** menu option **Mesh > Set reference named selection** is used to set the reference named selection. You can change the reference named selection only if no data objects exist in the database, which is directly after loading the reference mesh.

- **Nodes used for statistics/models**: All field data objects of type **node** that are imported or created in SoS are defined when this option is selected.

- **Elements used for statistics/models** or **reference element set**: All field data objects of type **element** are imported or created in SoS when this option is selected.
- **Fixed nodes for mesh morphing**: Fixed nodes are not modified when SoS performs geometry variations. You should define the fixed nodes named selection from nodes not contained within the reference node or element named selection used for statistics and models.

Note:

If you do not plan to modify geometries, you do not need to define fixed nodes.

5.2.4. Exporting a Geometry

The **File** menu option **Mesh > Export geometry** exports a deformed geometry into a simplified FEM mesh file. Depending on the format, SoS might not create an executable FEM mesh file. For example, SoS does not know material laws.

The output file contains at least finite element nodes and mostly finite elements (at least their respective analogy in SoS). If supported by the target format, node and element sets are exported. The mesh topology is based on the reference mesh, but you can change the x, y, and z coordinates.

When exporting, you can choose:

- The output file name and file format. Only STL, LS-DYNA, and Nastran mesh file formats are fully supported. ANSYS Mechanical mesh file formats are partially supported. For more information, see the appropriate file format specifications (p. 109).
- Whether the x, y, and z coordinates are modified.
- Whether the x, y, and z coordinates are defined by user-specified coordinates or coordinate deviations (with respect to the reference mesh).
- The data object that is to define the x, y, and z coordinates. The data object must be of type **node**. It is defined by its quantity identifier (from a drop-down list) and its design identifier (from the input line or a drop-down list). The design identifier can specify a sample number or a result name.

5.3. Data Imports

From the **File** → **Import Data** menu, you can perform import tasks:

5.3.1. Importing Scalar Data Objects from an optiSLang BIN File

The **File** menu option **Import data > Import scalars from optiSLang .bin** imports scalar data objects from a BIN file created with optiSLang. You use this option to get the scalar parameters associated with each field design, such as the scalar input parameters used during a field's simulation.

Subsequently, you can create a field-MOP or compute correlations between field quantities and scalar quantities.

Scalar quantities, including all design numbers, are stored in a single optiSLang bin file. optiSLang further defines failed designs and inactive designs as follows:

- Failed designs are not imported.
- Inactive designs are marked as inactive samples.

Only input parameters are imported by default. However, you can select the check box to also import scalar responses.

5.3.2. Importing Scalar Data Objects from a CSV File

The **File** menu option **Import data > Import scalar from CSV** imports scalar data objects from a CSV file. You use this option to get the scalar parameters associated with each field design, such as the scalar input parameters used during a field's simulation. Subsequently,you can create a Field-MOP or compute correlations between field quantities and scalar quantities

The scalar quantities, including all design numbers, are stored in a single CSV file. SoS tries to identify the delimiter automatically. However, you can define the delimiter before importing the file.

Here is an example of a CSV file:

```
#,scalar1,scalar2
1,0.0,1.0
2,99,3.14e100
```

5.3.3. Importing from a Static ROM Builder Project

The **File** menu option **Import data > Import SRB project** imports data objects from a static ROM builder project that was created in ANSYS Twin Builder. You use this option to get the following data objects from the static ROM builder project:

- Point cloud as a reference FEM mesh, including named selections
- Scalar input parameters for each field design
- Node field quantity samples

After importing a static ROM builder project, you have all scalar input parameters and their associated field designs for creating a field-MOP or an empirical random field model.

When importing a static ROM builder project, you select the folder containing the project files (`points.bin`, `doe.csv`, the snapshots folder, and more).

The option **First design identifier** allows you to append the imported data to existing designs in the database, starting with the design identifier set. SoS does not overwrite already existing designs but displays an error message instead.

The action that SoS takes when importing a static ROM builder project with an already existing reference mesh depends on whether this mesh is compatible with the project's point cloud:

- If the existing reference mesh is compatible with the project's point cloud, SoS uses its compatible mesh mapper for field data import.
- If the existing reference mesh is incompatible with the project's point cloud and the existing reference mesh is an FEM mesh with shell elements, SoS uses its closest point projection mesh mapper for field data import.

With incompatible mesh mapping, SoS reports mapping accuracy through log messages. Carefully analyze mapping results visually and compare minimum and maximum field values.

5.4. Data Exports

From the **File** → **Export Data** menu, you can perform export tasks:

5.4.1. Exporting Scalar Quantities to optiSLang

The optiSLang BIN file contains features of which SoS is unaware. Hence, SoS modifies an existing optiSLang BIN file by importing it, adding its own scalar parameters as optiSLang responses, and saving the modified data to a new file.

The **File** menu option **Export data > Export scalars to optiSLang** provides for selecting the subset of scalar quantities to export. If you do not select a subset, all scalar quantities are exported.

- The **Export generated scalars only** option excludes samples previously imported from a file.
- The **Export only active samples** option, selected by default, excludes inactive samples. Other designs are marked as `failed` in the output file.

Any scalars imported previously may be checked for compatibility with the chosen input bin file. The output optiSLang BIN file can be used directly as input for MOP in optiSLang.

5.4.2. Exporting Selected Scalar Quantities to a CSV File

The **File** menu option **Export data > Export selected scalar quantities to CSV** writes selected scalar quantities to a CSV file.

To control the export:

- Choose the delimiter for separating field entries
- Select the **optiSLang compatible idents** check box to generate strings on export with respect to naming conventions in optiSLang

To refine the selection to export:

- Select a subset of input, output, or both input and output scalar quantities. If you do not select a subset, no scalar quantities are exported and a warning is issued.

- Select the **Export generated scalars only** check box if you want to export only scalar quantities generated within SoS. For example, you might want to export only generated amplitudes and hot spots. Scalar quantities imported from an optiSLang BIN file are omitted.
- Select the **Export result objects** check box if you want to export only scalar quantities generated by SoS algorithms for display in the data table.
- Select the **Export sample objects** check box if you want to export all imported data except result objects.
- Select the **Export inactive objects** check box if you want to add inactive designs to the exported scalar data object.

For information on syntax rules used in exports, see CSV Files (p. 117), paying particular attention to Exporting Scalar Data from CSV Files (p. 119).

5.4.3. Exporting Selected Field Data Objects to a CSV File

The **File** menu option **Export data > Export selected field data objects to CSV** writes preselected individual node and element field data objects and samples to a CSV file.

To control the export:

- Choose the delimiter for separating field entries
- Select the **optiSLang compatible idents** check box to generate strings on export with respect to naming conventions in optiSLang
- Check additional check boxes to export field indices, such as the element and part number if multiple parts exist, and the Cartesian coordinates respectively.

To refine the selection to export:

- Select the **Export generated field data only** check box if you want to export only field data generated within SoS. For example, you might want to export only generated mean and random field shape objects. Imported field data is omitted.
- Select the **Export result objects** check box if you want to export only field data generated by SoS algorithms and displayed in the data table.
- Select the **Export sample objects** check box if you want to export all imported field data except result objects.
- Select the **Export inactive objects** check box if you want to add inactive designs to the exported field data object.

For information on syntax rules used in exports, see CSV Files (p. 117), paying particular attention to Exporting Field Data Objects (p. 122).

5.5. Multiple-Design Processing

From the **File** → **Process multiple designs** menu, you can import, modify, and export multiple designs in one action:

5.5.1. Importing Multiple Field Designs

The **File** menu option **Process multiple designs > Import field designs** imports multiple field designs at once. SoS expects the following directory structure:

```
.
+-- Design0001  # reference design directory
|   +-- file1   # file containing field or mesh data
|   +-- file2
|   +-- ...
+-- Design0002
|   +-- file1
|   +-- file2
|   +-- ...
+-- Design0003
...
```

If the reference mesh and field data mesh diverge from each other, SoS provides mesh morphing and mapping algorithms to adjust the FEM mesh position and map field data onto the reference mesh. For more information, see Compatible Mesh Mapper (p. 168).

SoS supports scalar fields only. If a file contains vector field data (such as stress tensors), each component is imported individually. An index is appended to the quantity identifier for each tensor component.

To import field designs:

1. Prepare the data objects to import.

   You can import field and mesh data objects.

2. On the first wizard page, add files located in your reference design directory.

   As you select appropriate file types, SoS parses them for field and mesh dash. SoS automatically detects the reference directory's path.

3. If you want SoS to create unique identifiers for quantities already imported, select the **Create unique idents** check box.

4. If you are using a mesh mapper other than `Compatible`, add the mesh file to the list of imported files.

5. Review the list of data objects found during file parsing:

   a. Ensure that only the identifiers to be imported are selected.

   b. Set a custom identifier for each data object.

6. On the second wizard page, ensure that **Base path** points to the directory containing all design directories.

   You can adjust the name of the individual design directories, defined by **Design directory format string**, using Perl Regular Expression syntax. In most cases, SoS automatically identifies the correct expression, especially when importing data from optiSLang or ANSYS projects. If this auto-detection fails, you can either select one of the predefined expressions from the list or enter a valid Perl expression. This expression can include directory separators, instructing SoS to descend several directory levels during its file system scan.

   Caution:

   While this procedure is convenient, it is prone to errors, such as missing file system permissions. Additionally, if numerous directories or numerous directory separators are used in the regular expression, long scan durations can occur.

   The file system scan does not follow any symbolic links. If the design number is to be extracted through **Design directory format string**, rather than generating one automatically in the range **`[1-n]`**, you must define a sub-expression.

   SoS detects the value for **Design number range** based on **Design directory format string**. You can edit it manually, such as **`1-10;12;14-20`**, or, if an optiSLang BIN file is available, you can use it to exclude failed designs form the import.

7. When applying (parallelized) mesh morphing and mapping algorithms, limit the number of designs imported in parallel by setting a non-zero value.

When importing mesh files, you can:

- Import node coordinates for each design
- Import mesh data from different files, such as node information in one file and element information in another file

Note:

- If an error occurs during the file import, the current design is excluded and import of the remaining designs continues.
- When data items (such as eroded elements) are missing, SoS marks the element or node as `missing` and calculates an interpolated value based on the topological neighbors.

5.5.2. Modifying and Exporting Multiple Field Designs

The **File** menu option **Process multiple designs** > **Modify/Export field designs** modifies and exports existing design directories adhering to the optiSLang design directory structure:

```
.
+-- Design0001  # reference design directory
|   +-- file1
|   +-- file2
|   +-- ...
+-- Design0002
|   +-- file1
|   +-- file2
|   +-- ...
+-- Design0003
...
```

SoS exports field data objects contained in the current database and sorts them into existing design directories according to their design identifier.

To modify and export field data:

1. Define the reference design directory template.

   All design directories are modified following this template.

2. On the first wizard page, set the path of the reference design.

   Files can be located in a subdirectory of the reference design path. The list of parsed files is exported relative to this path.

3. Determine your use case and perform the necessary steps:

   - Creating New Field Design Output Files (p. 71)
   - Modifying Existing CAE Solver Input Files (p. 72)

4. After completing the steps for your use case, on the second wizard page, set **Base path** to the directory containing all design directories.

5. Set **Design directory format string** to detect all design directories contained in the base path.

   If necessary, you can use Perl Regular Expression syntax to adjust the expression. For more information, refer to step 6 (p. 70) in the previous topic.

   SoS detects the value for **Design number range** based on **Design directory format string**. You can edit it manually, such as **`1-10;12;14-20`**.

Note:

- If an error occurs during the file export, the current design is excluded and export of the remaining designs continues.
- Eroded data cannot be exported. If eroded data exists, an interpolated value is taken.
- The modified files must contain meshes and data that are compatible with the reference mesh.

5.5.2.1. Creating New Field Design Output Files

To create new field design output files:

1. Select **Create new file** to export the resulting field design in a file format of your choice.

   This appends a new row to the list of data to export.

2. In the list of data to export, select the desired quantity identifier.

3. Select a design identifier for export.

4. If you are modifying and exporting field designs:

   a. Select **Use current design number** to sort the output file in the existing design directories according to the exported field object's design identifier.

      Otherwise, any field result object present in the database may be exported with the newly created file.

   b. Return to step 5 (p. 71) in Modifying and Exporting Multiple Field Designs (p. 70).

5. If you are exporting a reference design directory template and the files needed to generate new field designs in an optiSLang workflow:

   a. Select **Evaluate Field-MOP** or **Generate random field** to evaluate previously created field models for the selected quantity identifier.

      Otherwise, any field result object present in the database may be exported with the newly created file.

   b. Return to step 4 (p. 74) in Generating Field Objects in optiSLang for Use in a Simulation (p. 74).

5.5.2.2. Modifying Existing CAE Solver Input Files

During processing with optiSLang, the CAE solver input file selected for modification must exist in each design directory. SoS is unable to generate CAE solver input files with complete settings.

To modify field data in an existing CAE solver input file:

1. Select **Modify file** to edit field data in the file but leave it otherwise unchanged.

   For each modifiable field data object detected in the file, a new row is added to the list of data to export.

2. In the list of data to export, select the desired quantity identifier.

3. Select a design identifier for export.

4. If you are modifying and exporting field designs:

   a. Select **Use current design number** to modify files in the existing design directories that match the exported field object's design identifier.

      Otherwise, any field result object present in the database may be exported with the newly created file.

   b. Return to step 5 (p. 71) in Modifying and Exporting Multiple Field Designs (p. 70).

5. If you are exporting a reference design directory template and the files needed to generate new field designs in an optiSLang workflow:

   a. Select **Evaluate Field-MOP** or **Generate random field** to evaluate previously created field models for the selected quantity identifier.

      Otherwise, any field result object present in the database may be exported with the newly created file.

   b. Return to step 4 (p. 74) in Generating Field Objects in optiSLang for Use in a Simulation (p. 74).

Note:

You can specify node coordinate modification either by absolute coordinate values or by deviations from the reference mesh. If you specify both absolute coordinate values and deviations, the behavior is undefined.

5.6. Exports for Simulation

From the **File** → **Export for simulation** menu, you can export FMU 2.0 archive files, field objects, and amplitudes for use in simulations:

5.6.1. Exporting a Macro-Based FMU 2.0 Archive File

The **File** menu option **Export for simulation > Macro-based FMU** combines existing macros to automate a specific task and export an FMU 2.0 archive file for use with an external program. The FMU 2.0 archive file supports co-simulation and model exchange.

This option expects a set of scalar input parameters and returns a set of scalar output parameters:

- Evaluate an existing field-MOP with its set of input parameters, thereby creating a new field design.
- Return the maximum value or other scalar metrics of the newly created field design.

To build your custom task:

1. Select and add macros from the context-sensitive drop-down list.
2. Edit quantity identifiers that are shown in a cursive font.
3. On the wizard's second page, to export an FMU 2.0 archive file:

   a. Inspect the final input and output parameters.

b. Select a file type and file name.

   When you select an SDB databaseas the file type, your analysis is stored as a custom macro in the macro manager. For more information, see Managing Macros (p. 76). Via the SoS Python Package for optiSLang (p. 141), you can use this SDB database file in a custom optiSLang workflow.

You can use the `FMU_SoS` custom integration node to evaluate the exported FMU in optiSLang. A minimal copy of the currently loaded database is exported with the FMU. Field-MOPs and random field models are included. Field designs, scalar data objects, and result data objects are excluded.

The exported FMU operates on the level of a single design. When setting new scalar input parameters and executing the task, a new design identifier is created, storing input and output parameters in the database. On FMU termination, an SDB database file is written, containing the minimal database and the newly created designs. To set the file name (absolute path) of the FMU's input/output SDB database file, you use the exported FMU's string parameter.

5.6.2. Generating Field Objects in optiSLang for Use in a Simulation

The **File** menu option **Export for simulation > Generate field objects in optiSLang** exports a reference design directory template and the files needed to generate new field designs in an optiSLang workflow. New field designs are created by evaluating an existing field-MOP or random field model for a set of scalar input parameters or random field amplitudes.

To export field objects:

1. Define the reference design directory template.

   In optiSLang, all design directories are treated based on this template.

2. On the first wizard page, set the path of the reference design.

   Files can be located in a subdirectory of the reference design path. The list of parsed files is exported relative to this path.

3. Determine your use case, performing the necessary steps:

   - Creating New Field Design Output Files (p. 71)
   - Modifying Existing CAE Solver Input Files (p. 72)

4. After completing the steps for your use case, on the second wizard page, choose an output format:

   - Export as a single archive (.sim) file. optiSLang can import this file.
   - Export all individual files and the reference design directory template to the output path.

5. Set the output path.

SoS does not modify any files. It only creates the reference design directory template containing the reference files.

SoS exports files that are needed to evaluate field models:

- CSV file containing scalar field-MOP input parameters and random field amplitudes
- SoS database file containing the random field models
- SoS script file (SSC) to evaluate the random field models.

The SoS database file `randomfield_simulation_data.sdb` contains the reference mesh and used scatter shapes.

5.6.3. Generating Amplitudes from Fields in optiSLang for Use in a Simulation

The **File** menu option **Export for simulation > Generate amplitudes from fields in optiSLang** exports the files that are required by optiSLang to calculate shape amplitudes from field data. You use the `AmplitudesFromField_SoS` custom integration node to return the shape amplitudes as scalar parameters in optiSLang. You can provide field data in any format that SoS handles:

- Node data from measurement in CSV format
- File output data from your favorite CAE solver

To generate amplitudes from fields in optiSLang:

1. Add at least one file containing your field data.
2. On the first wizard page, set the path of the reference design.

   The reference design is a single design directory serving as a template for all design directories.
3. Select the field quantities to be imported for shape amplitude calculation by selecting the appropriate check boxes.
4. Optionally, select a mesh-mapping algorithm.
5. On the second wizard page, select an output path for template folder.

The template folder contains all files needed to run the `AmplitudesFromField_SoS` custom integration node in optiSLang.

5.7. Macros

From the **File** → **Macros** menu, you can import, export, and manage macros:

5.7.1. Importing Macros from an SoS Database

The **File** menu option **Macros > Import macros from SDB** imports user-defined macros contained in an SoS database file. Macros with identifiers already existing in the database are not replaced.

5.7.2. Exporting Macros to an SoS Database

The **File** menu option **Macros > Export macros to SDB** exports user-defined macros to an SoS database file.

5.7.3. Saving the Session Script

The **File** menu option **Macros > Save session script as** saves all processed script commands. You can then run this script to recreate all actions processed in the GUI exactly. You can adapt the script to your needs and create a workflow for automated batch analysis.

5.7.4. Managing Macros

The **File** menu option **Macros > Manage macros** provides for viewing and managing both SoS-supplied macros and your user-defined macros. The **Manage macro** window supports:

Note:

In the Toolbox Window (p. 32), the **Manage macros** option is under the **Field data models** functional category heading.

Guidelines

- Macros with duplicate identifiers fail to work.
- User-defined macros are saved to the SoS database file. For more information, see Adding or Editing a User-Defined Macro (p. 77).
- Macros compatible with the SoS FMU exporter must take a `database` type as the first argument and a `design ident` as the second argument. For more information, see Writing Macros Compatible with the FMU Exporter (p. 78).
- To protect your intellectual property, you should obscure your macros. For more information, see Obscuring a User-defined Macro (p. 79)

5.7.4.1. Adding or Editing a User-Defined Macro

You can add a new or edit an existing user-defined macro for custom analysis and data processing and then test its validity.

This table describes the fields in the **Edit chunks / macros** window:

Field	Description
Ident	The macro is added to the Lua table `sos_chunks`. You call the macro from the script with **`sos_chunks.ident()`**.
Name	A human-readable version of the identifier that appears in the FMU exporter.
Brief help	A brief function description of the macro that is displayed by the FMU exporter.
Help	An extended function description of the macro that is displayed by the FMU exporter.
Input arguments	Provides for defining the variable name, type, and help text for input arguments.
Return values	Provides for defining the variable name, type, and help text for return values.
Code	The SoS script code or Lua code. For more information, see SoS-Embedded Scripting Using Lua (p. 141).
Compatibility	Defines the macro's mesh compatibility.
Requires Mesh	Indicates whether the mesh is required.
Test code	Provides for testing your macro. You can write your own test code or click **Generate test template** to have SoS attempt to generate test code with sensible values. Clicking **Test** runs the macro tests for syntax errors and asserts the correct return values. For more information, see Testing User-defined Macros (p. 78).

This next table describes the data types for input arguments and return values:

Data Type	Description
string	A basic string type.
integer number	A basic integer type.
floating point number	A basic floating point type.
bool	A basic Boolean type.
matrix	A tmath.Matrix.
input scalar ident	A string type for data object identifiers. Use this in combination with the **`database`** and **`design ident`** types. The macro expects that the key **`sos.DataObjectKey(designIdent, objectIdent)`** exists in the database.
input node ident	
input element ident	
output scalar ident	A string type for data object identifiers. Use this in combination with the **`database`** and **`design ident`** types. The macro writes a data object with key **`sos.DataObjectKey(designIdent, objectIdent)`** to the database.
output node ident	
output element ident	

Data Type	Description
remove scalar ident	A string type for data object identifiers. Use this in combination with the `database` and `design ident` types. The macro removes a data object with key `sos.DataObjectKey(designIdent, objectIdent)` from the database.
remove node ident	
remove element ident	
database	An `sos.Structure` object that lets the macro work on a specific database. Pass `sos.database()` to use the current global database.
design ident	A string type for design identifiers that let macros work on a specific design identifier.
unknown	Any other type.

5.7.4.2. Writing Macros Compatible with the FMU Exporter

The SoS FMU exporter writes FMU input variables to the global database (`sos.database()`) and reads FMU output variables from the global database.

- For macros to be compatible with the FMU exporter, they must take a `database` type as the first argument and a `design ident` type as the second argument.
- If the macro is to read a scalar FMU 2.0 input variable, it takes an `input scalar ident` type as an argument and reads the scalar input's value by calling `database:scalarData():find(designIdent, scalarIdent)`.
- If the macro is to provide a scalar FMU 2.0 output variable, it takes an `output scalar ident` type as an argument, and creates a scalar data object with key `sos.DataObjectKey(designIdent, scalarIdent)` in the database that is passed with the first argument.

5.7.4.3. Testing User-defined Macros

You should test your macros for syntax errors and output validity. You can let SoS attempt to generate test code with sensible argument values or write your own test code for more elaborate testing. When the test is executed, the macro is saved.

Caution:

When you execute a test, you are using the global SoS scripting environment, which means that your tests can change the global database.

5.7.4.4. Copying a Macro

You can copy either a supplied or user-defined macro and then edit it to create a new user-defined macro. When SoS creates the copy, it adds the suffix `_copy` to the identifier. However, you can change the identifier to any unique value when you edit the new user-defined macro.

5.7.4.5. Deleting a User-defined Macro

You can delete a user-defined macro from the SoS database. It is excluded from the database when the database is saved. You cannot delete a standard macro.

5.7.4.6. Obscuring a User-defined Macro

To protect your intellectual property, you can obscure a user-defined macro. Once the macro is obscured, you cannot display or edit the macro's code. However, ANSYS is able to restore the code.

Chapter 6: SoS Data Objects

Using options on the **Edit** menu, you can edit a project name and activate, deactivate, and delete SoS data objects:

Note:

The **Toolbox** window provides quick access to **Edit** menu options under the functional category heading **Edit**. For more information on this docked window, see Toolbox Window (p. 32).

6.1. Project Name

The **Edit** menu option **Edit project name** allows you to change the project name. The project name is an optional identifier that appears in the description strings of all data objects and in the legend in the 3D visualization. When you change the project name, all manually edited legend strings are lost.

6.2. Data Object Activation

From the **Edit** menu, you can activate data objects:

6.2.1. Activating All Objects

The **Edit** menu option **Activate all objects** makes all samples (field and scalar data) active.

6.2.2. Activating Selected Objects

The **Edit** menu option **Activate selected objects** makes all samples selected in the data object table active.

6.3. Data Object Deactivation

From the **Edit** menu, you can deactivate data objects:

6.3.1. Deactivate selected objects

The **Edit** menu option **Deactivate selected objects** deactivates all samples selected in the data object table.

6.3.2. Deactivate designs with eroded elements

The **Edit** menu option **Deactivate designs with eroded elements** excludes all samples with eroded data from the statistical analysis. It deactivates the samples of all field and scalar quantities if the number is missing in at least one field quantity of the respective samples.

6.3.3. Deactivate incomplete designs

The **Edit** menu option **Deactivate incomplete designs** helps you to keep the database consistent. For any sample, if at least one scalar or field quantity is inactive or not imported (deleted or not existing due to any reason), the whole sample is made inactive for all scalar and field quantities.

6.4. Data Object Deletion

From the **Edit** menu, you can delete data objects:

6.4.1. Delete selected objects

The **Edit** menu option **Delete selected objects** removes all data objects selected in the data object table. When the data object table has keyboard focus, you can also press the **Backspace** key to perform this action.

6.4.2. Delete selected field quantities

The **Edit** menu option **Delete selected field quantities** removes all data objects (samples and results) belonging to the selected field quantities in the quantity list. When the data field quantity list has keyboard focus, you can press the **Backspace** key to perform this action.

6.4.3. Delete selected scalar quantities

The **Edit** menu option **Delete selected scalar quantities** removes all data objects (samples and results) belonging to the selected scalar quantities in the quantity list. When the input or output scalar quantity list has keyboard focus, you can also press the **Backspace** key to perform this action.

Chapter 7: SoS Statistical Measures

Using options on the **Statistics** menu, you can calculate statistical measures.

Note:

The Toolbox Window (p. 32)**Toolbox** window provides quick access to all **Statistics** menu options under the **Statistics** functional category heading.

On the **Statistics** menu, submenus organize the measures that can be calculated. For more information, see these topics:

Guidelines

- Most functions are applied to all active designs of a selected quantity.
- Only active samples are used as inputs.
- Missing data is removed from the set of input samples.
- If errors occur during computation of a statistical measure, where applicable, the respective error locations are highlighted as missing data in the result object.

7.1. Standard Estimators

From the **Statics** → **Standard estimators** menu, you can calculate standard estimators:

7.1.1. Calculating the Mean

The **Statistics** menu option **Standard estimators > Mean** computes the mean value for all selected field quantities. For each selected quantity, a single result object is created. Each result object is a field vector that contains the mean at each mesh position.

For quantity **Y**, number of active and non-missing sample values **N**, sample index **i**, and mesh position **k**, the mean value is:

$$\mu[Y_k]=E[Y_k]=\frac{1}{N_k}\sum_{i=1}^{N_k} Y_k^i$$

7.1.2. Calculating the Standard Deviation

The **Statistics** menu option **Standard estimators > Standard deviation** computes the standard deviation σ for all selected field quantities. For each selected quantity, a single result object is created. Each result object is a field vector that contains the standard deviation at each mesh position.

For quantity **Y**, number of active and non-missing sample values **N**, sample index **i**, and mesh position **k**, the standard deviation is:

$$\sigma[Y_k]=\sqrt{E[(Y_k-E[Y_k])^2]}=\sqrt{\frac{1}{N_k-1}\sum_{i=1}^{N_k}\left(Y_k^i-E(Y_k)\right)^2}$$

7.1.3. Calculating the Variance

The **Statistics** menu option **Standard estimators > Variance** computes the variance `Var` for all selected field quantities. For each selected quantity, a single result object is created. Each result object is a field vector that contains the variance at each mesh position.

For quantity **Y**, number of active and non-missing sample values **N**, sample index **i**, and mesh position **k**, the variance is:

$$\mathrm{Var}[Y_k]=E[(Y_k-E[Y_k])^2]=\frac{1}{N_k-1}\sum_{i=1}^{N_k}\left(Y_k^i-E(Y_k)\right)^2=\left(\sigma[Y_k]\right)^2$$

7.1.4. Calculating the Coefficient of Variation (CoV)

The **Statistics** menu option **Standard estimators > Coefficient of Variation (CoV)** computes the CoV for all selected field quantities. For each selected quantity, a single result object is created. Each result object is a field vector that contains the CoV at each mesh position.

For quantity **Y** and mesh position **k**, the CoV is:

$$CoV[Y_k]=\frac{\sigma[Y_k]}{\mu[Y_k]}$$

7.2. Value Ranges

From the **Statistics** → **Value Ranges** menu, you can calculate value ranges:

7.2.1. Calculating a Range of Minimum Sample Values

The **Statistics** menu option **Value ranges > Minimum sample values** computes the minimum sample values for all selected field quantities. For each selected quantity, a single result object is created. Each result object is a field vector that contains the smallest value among all samples at this mesh position.

For quantity **Y**, number of active and non-missing sample values **N**, sample index **i**, and mesh position **k**, the minimum sample value is:

$$\min[Y_k]=\min_i Y_k^i$$

7.2.2. Calculating a Range of Maximum Sample Values

The **Statistics** menu option **Value ranges > Maximum sample values** computes the maximum sample values for all selected field quantities. For each selected quantity, a single result object is created. Each result object is a field vector that contains the largest value among all samples at this mesh position.

For quantity **Y**, number of active and non-missing sample values **N**, sample index **i**, and mesh position **k**, the maximum sample value is:

$$\max[Y_k]=\min_i Y_k^i$$

7.2.3. Calculating a Range of Sample Values

The **Statistics** menu option **Value ranges > Range of sample values** computes the value ranges for all selected field quantities. For each selected quantity, a single result object is created. Each result object is a field vector that contains the difference between the largest and smallest values among all samples at this mesh position.

For quantity **Y** and mesh position **k**, the sample range is:

$$range[Y_k]=\max[Y_k]-\min[Y_k]$$

7.3. Quantiles and Probabilities

From the **Statistics** → **Quantiles and probabilities** menu, you can calculate quantile and probability values:

7.3.1. Calculating the Probability of Being the Minimum Value

The **Statistics** menu option **Quantiles and probabilities > Probability being minimum** computes the relative probability for each node or element of being the absolute minimum value. This is done for all selected field quantities. For each selected quantity, the algorithm counts how often a node or element contains the minimum value for each sample. For any sample, if several nodes or elements have the same minimum value, all of them are counted.

7.3.2. Calculating the Probability of Being the Maximum Value

The **Statistics** menu option **Quantiles and probabilities > Probability being maximum** computes the relative probability for each node or element of being the absolute maximum value. This is done for all selected field quantities. For each selected quantity, the algorithm counts how often a node or element contains the maximum value for each sample. For any sample, if several nodes or elements have the same maximum value, all of them are counted.

7.3.3. Calculating Quantile Values

The **Statistics** menu option **Quantiles and probabilities > Quantile** computes the quantile values for all selected field quantities and a user-specified probability value. For each selected quantity, a single result object is created. Each result object is a field vector containing the quantile at each mesh position.

Note:

Quantiles need to have at least 20 input samples.

Given a cumulative distribution function **`F(Y)`**, the quantile value **q** for a specific probability **p** is defined as:

$$q(Y_k,p)=F_k^{-1}(p)$$

where **Y** denotes the quantity and **k** denotes the mesh position.

The CDF is approximated from the given sample values. Generally, SoS uses the CDF of a histogram. At its "tails" (for small and large probabilities), SoS approximates the CDF by a quadratic regression model for an assumed Gaussian distribution. This strategy ensures that quantile values beyond the actual value bounds $\mathtt{min[Y_k]}$ and $\mathtt{max[Y_k]}$ can be obtained.

Usage

- Compute the quantile value of **p** if you are interested in the value that is not exceeded by the probability **p**.

- Compute the quantile value of **`1-p`** if you are interested in the value that is exceeded by the probability **p**.
- Typical quantile probabilities are 0.135% (**`p=0.00135`**) and 99.865% (**`p=0.99865`**).

7.3.4. Calculating the Non-exceedance Probability

The **Statistics** menu option **Quantiles and probabilities > Non-exceedance probability** computes the non-exceedance probabilities for all selected field quantities and a user-specified threshold value. For each selected quantity, a single result object is created. Each result object is a field vector that contains the probability at each mesh position.

Note:

Quantiles need to have at least 20 input samples.

Given a cumulative distribution function **`F(Y)`**, the probability **p** for a quantile value **q** is defined as:

$$p(Y_k, q) = \mathrm{F}_k(q)$$

where **Y** denotes the quantity and **k** denotes the mesh position.

The CDF is approximated from the given sample values as described in Calculating Quantile Values (p. 86).

Usage

- Compute the probability **p_k** for which a certain threshold **q** is not exceeded.
- If you are interested in the exceedance probability, you need to interpret the values using **$1-p_k$**.

7.3.5. Calculating the Mean + k*sigma Value

The **Statistics** menu option **Quantiles and probabilities > Mean + k*sigma** computes the sum of the mean and a multiple of the standard deviation for all selected field quantities. For each selected quantity, a single result object is created. Each result object is a field vector containing the desired value at each mesh position.

Note:

The scaling factor is user-specified.

The resulting limit value **q** is defined by:

$$q_k = \mu[Y_k] + k\sigma[Y_k]$$

where **Y** denotes the quantity, **`j`** denotes the mesh position, and **k** denotes the user-specified scaling factor.

Usage

- Area of application: Robustness evaluation based on safety margins (such as distances with respect to the mean value).
- Use negative multiples to get lower threshold values.

7.3.6. Calculating the Probability of the k*sigma Interval

The **Statistics** menu option **Quantiles and probabilities > Probability of k*sigma interval** computes the probabilities that the values of the selected field quantity are within the interval:

$$[\mu+k_i\sigma,\mu+k_u\sigma]$$

If the field quantity's values are distributed normally, the resulting probabilities follow the 68–95–99.7 rule. This measure can be used as a simple test for outliers if the probability distribution is assumed normal and as a normality test if the probability distribution is potentially not normal.

Given a cumulative distribution function **F(Y)**, the probability **p** for the k-sigma-interval is defined as:

$$[p(Y_k,k_l,k_u)=F_k(\mu_k+k_u\sigma_k)-F_k(\mu_k+k_l\sigma_k)$$

where **Y** denotes the quantity, **j** denotes the mesh position, and $\mathbf{k_l}$ and $\mathbf{k_u}$ denote the lower and upper scaling factor of the standard deviation σ.

The CDF is approximated from the given sample values as described in Calculating Quantile Values (p. 86).

Note:

- The computation of the CDF requires at least 20 input samples.
- If you do not have a lower bound, use a lower factor of -8.
- If you do not have an upper bound, use a lower factor of +8.

7.4. Quality Capability Values

From the **Statistics** → **Process** menu, you can calculate quality capability values:

Quality Capability Statistics (QCS), also called Process Capability Statistics (PCS), covers the calculation of values according to DIN 55319-3:2002 Method M4. A distinction is made between variance-based (C_p) and variance location-based (C_{pk}) values.

Note:

QCS is typically used in conjunction with traffic light plots, with red values for $c_{pk} < 1.0$ and green values for $c_{pk} > 1.33$.

7.4.1. Calculating the Quality Capability Cp Value

The **Statistics** menu option **Process > Quality capability Cp** computes the C_p for all selected field quantities and a user-specified threshold value. For each selected quantity, a single result object is created. Each result object is a field vector that contains the C_p value at each mesh position.

Note:

- Internally, the command computes quantiles, which requires at least 20 input samples.
- The lower and upper bounds are user-specified values.

For C_p (scattering-oriented QCS):

$$C_p(Y_k)=\frac{U-L}{q(Y_k,0.99865)-q(Y_k,0.00135)}$$

where **U** and **L** denote the upper and lower bounds, respectively, and $\mathbf{q(Y_{k,p})}$ denotes the p-th quantile of quantity $\mathbf{Y_k}$.

The quantile probabilities correspond to the plus and minus **3** -quantiles of a Gaussian distribution. **Y** denotes the quantity and **k** denotes the mesh position.

7.4.2. Calculating the Quality Capability Cpk Value

As indicated in the previous topic, a distinction is made between variance-based (C_p) and variance location-based (C_{pk}) values.

The **Statistics** menu option **Process > Quality capability Cpk** computes the C_{pk} values for all selected field quantities and a user-specified threshold value. For each selected quantity, a single result object is created. Each result object is a field vector that contains the C_{pk} value at each mesh position.

The input and error information for calculating C_{pk} is the same as that given in the previous topic for C_p.

For C_{pk} (scattering- and location-oriented QCS):

$$C_{pk}(Y_k)=\min(C_{pk,lower},C_{pk,upper})=\min\left(\frac{\mu(Y_k)-L}{\mu(Y_k)-q(Y_k,0.00135)},\frac{U-\mu(Y_k)}{q(Y_k,0.99865)-\mu(Y_k)}\right)$$

with estimated mean $\mu(Y_k)$.

7.5. Correlations

From the **Statistics** → **Correlations** menu, you can calculate correlation values:

7.5.1. Calculating the Linear Correlation

The **Statistics** menu option **Correlations > Linear correlation** computes the linear correlation for all selected field and scalar quantities. For each selected quantity pair (all combinations of a single field quantity and a scalar quantity), a single result object is created. Each result object is a field vector that contains the linear correlation at each mesh position for the specific field quantity with the respective scalar quantity.

Note:

Design identifiers of the field and scalar quantities must be compatible.

For field quantity **Y**, scalar quantity **Z**, number of active and non-missing samples **N**, sample index **i**, and mesh position **k**, the linear correlation is:

$$\rho[Y_k,Z]=\frac{E[(Y_k-E[Y_k])(Z-E[Z])]}{\sigma[Y_k]\sigma[Z]}=\frac{1}{\sigma[Y_k]\sigma[Z]}\frac{1}{N_k-1}\sum_{i=1}^{N_k}(Y_k^i-E[Y_k])(Z^i-E[Z]$$

7.5.2. Calculating the Coefficient of Determination (CoD)

The **Statistics** menu option **Correlations > Coefficient of Determination** computes the adjusted CoD for a linear regression model for all selected field and scalar quantities. For each selected field quantity, a single result object is created. Each result object is a field vector that contains the CoD of the specific field quantity with the selected scalar quantities.

Note:

Design identifiers of the field and scalar quantities must be compatible.

7.6. Errors and Differences

From the **Statistics** → **Errors and differences** menu, you can calculate both errors and differences:

7.6.1. Calculating the Eroded Data Fraction

The **Statistics** menu option **Errors and differences > Eroded data fraction** computes the eroded data fraction value μ_e for all selected field quantities. For each selected quantity, a single result object is created. Each result object is a field vector that contains the eroded data fraction at each mesh position. Only active samples are used as inputs.

For quantity **Y**, number of active samples **N**, sample index **i**, and mesh position **k**, the eroded data fraction is the mean value of the missing data items:

$$\mu_e[Y_k]=E[M(Y)_k]=\frac{1}{N_k}\sum_{i=1}^{N_k}M(Y^i)_k$$

where the function $M(Y^i)$ is an indicator function for missing data.

It returns a field vector that is **1** for all missing positions in the mesh and **0** for all existing mesh values.

Usage

The eroded data fraction denotes the probability of element erosion at a specific mesh position.

7.6.2. Calculating the Relative Error

The **Statistics** menu option **Errors and differences > Compute relative error** computes the relative error ε between selected objects and results of one or typically two selected field quantities of the same data type. Acceptable field quantities are of type node or element only. For each selected design, a single result object with the same design identifiers as its sources are created. Each result object is a field vector ε that contains $_k$ at each mesh position **k**:

$$\varepsilon_k=\frac{\varepsilon_{\text{other},k}}{\varepsilon_{\text{ref},k}}$$

- You must define a quantity identifier for the new field quantity ε.
- You can define the reference quantity identifier.
- If you select the **Use interpolated missing items** check box, SoS uses interpolated values based on topological neighbors wherever **other** or **ref** have missing values. Otherwise, ε has missing values on the same field indices as **other** and **ref**.
- If the reference values at mesh position **k** equals **0**, the division would not be defined. In such cases, $_k$ is either **1** if $\mathbf{other_k}$ equals **ref** and **0** otherwise.

7.6.3. Calculating the Standard Error of Mean

The **Statistics** menu option **Errors and differences > Standard error of mean** computes the standard error, which is the standard deviation of the mean estimator based on all selected field quantities. For each selected quantity, a single result object is created. Each result object is a field vector that contains the standard error at each mesh position.

For quantity **Y**, number of active and non-missing samples **N**, sample index **i**, and mesh position **k**, the standard error is:

$$\sigma_\mu[Y_k] = \sqrt{\mathrm{Var}\big(\mu[Y_k]\big)} = \frac{\sigma[Y_k]}{\sqrt{N_k}}$$

7.6.4. Calculating the Standard Error of Variance

The **Statistics** menu option **Errors and differences > Standard error of variance** computes the standard error of variance, which is the standard deviation of the variance estimator based on all selected field quantities. For each selected quantity a single result object is created. Each result object is a field vector that contains the standard error at each mesh position.

For quantity **Y**, number of active and non-missing samples **N**, sample index **i**, and mesh position **k**, the standard error is:

$$\sigma_{\mathrm{Var}}[Y_k] = \sqrt{\mathrm{Var}\big(\mathrm{Var}[Y_k]\big)} = \mathrm{Var}[Y_k]\sqrt{\frac{2}{N_k - 1}}$$

7.7. Filters

From the **Statistics** → **Filters** menu, you can select **Robust Principal Component Analysis** to perform Robust Principal Component Analysis (RPCA). RPCA is an algorithm for statistical pre-filtering of outliers in measurements. The objective is to separate correlations between field design from outliers. While traditional Principal Component Analysis (PCA) is very sensitive to data corruption or outliers, RPCA is robust to data corruption under surprisingly broad conditions.

RPCA attempts to split a given matrix (**M**) into two matrices (**S** and **L**):

M = **L** + **S**

where **L** is a low-rank matrix and **S** is a sparse matrix of random errors (of arbitrary magnitude and random sign). In the context of SoS, each column vector of **M** might be a particular field design. Without any prior knowledge about outliers, RPCA is then able to separate correlations between field designs (**L**) from outliers (**S**).

Security camera footage is a good example. Following the **M** = **L** + **S** data model, **L** represents the slowly changing background, while **S** represents movement (the people who are walking). For more examples and a precise definition of the conditions for RPCA to deliver good results, see https://arxiv.org/abs/0912.3599.

By default, **ComputeRPCA** creates two new quantity identifiers named **RPCA[quantityIdent]** (**L**) and **RPCAError[quantityIdent]** (**S**). The algorithm attempts to recover **L** and **S** by running the Principal Component Pursuit bi-objective optimization program:

$$\text{minimize } \|L\|_* + \lambda \|S\|_1$$

where $\|L\|$ denotes the nuclear norm of Y (the sum of the singular values of **L**) and $\|L\|_1 = \sum_{ij} |S_{ij}|$ denotes the l_1 -norm of **S** seen as a long vector.

The recommended value is $\lambda = 1 / \sqrt{\max(\dim(M))}$.

Chapter 8: SoS Field Data Models

Using options on the **Field data models** menu, you can create and evaluate field-MOPs, random field models, and field samples:

Note:

The Toolbox Window (p. 32) provides quick access to **Field data models** menu options under the **Field data models** functional category heading. This functional category heading also displays the option for managing macros (p. 76).

Guidelines

- Most functions are applied to all active designs of a selected quantity.
- Only active samples are used as inputs.
- Missing data is removed from the set of input samples.
- If errors occur during computation of a statistical metric, where applicable, the respective error locations are highlighted as missing data in the result object.

8.1. Field-MOPs

From the **Field data models** → **Field-MOP** menu, you can create field-MOPs and show field-MOP subspace interpolations in optiSLang:

8.1.1. Creating Field-MOPs

The **Field data models** menu option **Field-MOP > Create Field-MOP** creates a field meta model of optimal prognosis (field-MOP) based on previously imported field response and input parameter samples. Only active samples are considered. The field meta model is stored internally.

Note:

You can view the current list of created field meta models in the **Field data models** window. For more information, see Field Data Models Window (p. 33).

The **Create Field Metamodel of Optimal Prognosis (Field-MOP)** window has two tabs: **Select training data** and **Settings**.

- On the **Select training data** tab, you select the inputs and outputs for the field-MOP from the list of field and scalar quantities. SoS does not allow you to drag and drop a quantity to a location where it is not supported. Inputs that you drop into the **Input (mandatory)** location are included in the field-MOP, even if their sensitivity indicates a negligible influence on the field-MOP.
- On the **Settings** tab, you specify the settings to use. Descriptions of all possible settings follow. However, as you specify a setting, the visibility and available of other settings can change.

Settings

Maximum number of shapes

The upper bound for the number of scatter shapes to be computed in the internally used random field model. The default is **10**. This option is used to limit hardware resource usage.

Use a single cross-correlated model, ident

Clear this check box to create an individual field meta model for each selected quantity. Select this check box to collect all selected quantities into a single field meta model. In the text box on to the right, specify the identifier for the new data object to create. The number of active designs must be equal for all selected quantities.

Use interpolated missing items

Clear this check box to use the mean value for missing items or eroded elements. Select this check box to use the interpolated (filled) value for missing items. If the missing item originated from an incompatible mesh projection, selecting the check box is recommended.

Compute sensitivities

Available only when you are not using the MOP backend, select this check box to calculate the sensitivity of the field-MOP's output with respect to each input parameter. A `F-CoP[input parameter]` field data object is created for every relevant or mandatory input parameter.

Compute R*sigma

Select this check box when not using the MOP-backend to generate data objects that contain the F-CoP values in absolute numbers, which is the square root of the F-CoP scaled by the individual standard deviation for each spatial point.

Default meta model settings

Use default settings, balancing accuracy and performance. SoS has built-in checks to improve performance for a large number of designs:

- Number of RBF localization points is limited to 200.
- Number of reshuffles is limited to 200.
- Anisotropic function kernels are disabled if the number of designs is larger than 500.

Fast mode for meta model creation

Settings are optimized for performance at the expense of accuracy. Check the command log for the settings that are used.

Use custom settings of internal meta model

Selecting this option displays the **Custom meta model settings** area with additional settings. For more information, see Custom Meta Model Settings (p. 97).

Use custom settings for MOP backend

Selecting this option displays the **MOP settings** area with the advanced MOP settings that are used in optiSLang. For more information, see MOP Settings (p. 98)

Custom Meta Model Settings

Use anisotropic function kernels

Select this check box if you want the RBFs scaled with a distance factor individual for each input parameter. This tends to improve the F-CoP at the expense of performance.

Use custom localization points, number

Number of support points to use for RBF localization. This number must not be greater than the number of designs. You use this option to reduce computation time of the field-MOP.

Filter input parameters (input-output polynomial correlations)

Select this check box to eliminate input parameters that do not contribute to a polynomial regression model.

Do not filter inputs with a linear correlation significance level

When filtering, do not exclude input parameters above the specified linear correlation significance level. The default is **0.99**.

Use custom number of reshuffles for sensitivities

Number of reshuffles of test sets being used to estimate the sensitivity indices. A greater number leads to more accuracy.

Filter input-input correlations, limiting eigenvalue of correlation

Smallest eigenvalue of the input-input correlation matrix being used to identify dependent inputs. For example, use this in robustness sampling or with measured input data.

Iteration tolerance

Termination tolerance for internal iterative optimization loop (for gradient condition and function decrease).

MOP Settings

From the check boxes on the left, select the models to use to build the MOP:

- - **Use polynomial models**
  - **Use Moving Least Squares models**
  - **Use Kriging as model**
  - **If Kriging: Use anisotropic function kernels**
  - **Use Box-Cox transformation**

From the check boxes on the right, apply filters:

- **Use input-output correlation filter**
- **Use input-input correlation filter**
- **Use significance filter**
- **Use CoI filter**
- **Use CoD filter**

Output Objects for Each Quantity

For each quantity, these output objects are generated:

- Field-MOP model
- Mean value vector and standard deviation (**`mean[FMOP]`** and **`sigma[FMOP]`**) that are used
- F-CoP for each input parameter and the accuracy of the whole field meta model (**`F-CoP[Total]`**).
- F-CoP values in absolute numbers (**`R*sigma`**).
- Log messages:

- SoS determines the fraction of explainable variation that is obtained for individual numbers of scatter shapes and prints this table in the log messages.
- SoS prints further intermediate results to the log messages.

8.1.2. Showing Field-MOP Interpolations in optiSLang

The **Field data models** menu option **Field-MOP > Show Field-MOP supspace interpolation in optiSLang** exports some debug information for field-MOP models. For all selected quantities, the associated field-MOP model is identified and an optiSLang OMDB file is created.

You select the directory to which to export OMDB files. An OMDB file contains the meta models and anthill plots of the input designs for each internal field-MOP amplitude (the interior reduced space). The OMDB files are then opened in optiSLang postprocessing (if installed).

Caution:

This function is currently restricted to meta models based on the MOP backend. For meta models not based on the MOP backend, an exported OMDB file contains all design points, inputs and outputs, that were under consideration. However, the file does not contain the interpolation model that was used. Instead, it shows a polynomial regression model, regardless of the model that was used.

8.2. Random Field Models

From the **Field data models** → **Random fields** menu, you can create random field models and compute errors for random fields:

8.2.1. Creating Empirical Random Field Model

The **Field data models** menu option **Random fields > Create empirical random field model** creates a random field model based on previously imported samples. The random field model is stored internally. The current list of created random field models is shown in the **Field data models** window. For more information, seeField Data Models Window (p. 33).

Before selecting this command, you must select the physical quantities in the quantity lists for which a random field model should be created. Only active samples are considered.

In the dialog box, you set these options:

Desired explained variation (%)

Accuracy of the random field model. For simulation of random designs, 90% might be sufficient. SoS automatically determines the number of scatter shapes required by the random field model to represent the desired variation.

Maximum number of shapes

Upper bound for the number of scatter shapes to be computed for the random field model. This option is used to limit hardware resource usage.

Find cross correlation

Clear this check box to create an individual random field model for each selected quantity. Select this check box to collect all selected quantities into a single random field model. In this case, the number of active designs must be equal for all selected quantities.

Field group ident

In case of a cross-correlated random field model, the new identifier.

Compute field amplitudes

Select this check box to compute samples for random field model amplitudes. You can export these amplitudes for further investigation in other software, such as optiSLang. The amplitudes of the same random field model are uncorrelated random variables with zero mean and unit standard deviation.

Compute cumulative variations

Select this check box to generate data objects that represent the cumulative explainable variation by the random field model, given a varying number of scatter shapes.

Compute individual variations

Select this check box to generate data objects that represent the individual explainable variation that is explained by each individual scatter shape.

Use interpolated missing items

Clear this check box to use the mean value for missing items or eroded elements. Select this check box to use the interpolated ("filled") value for missing items. Selecting this check box is recommended if the missing item originated from an incompatible mesh projection.

Include selected scalar quantities

Select this check box to also compute a random field model for selected scalar quantities. This might only make sense in case of cross-correlated fields.

Output Objects for Each Quantity

For each quantity, these output objects are generated:

- Random field model
- Individual scatter shapes as results in the data object table: `shape[1]` … `shape[n]`. The data object table shows the cumulative explainable variation for the complete field (as average) for each shape.

- Explainable variation of the random field model as a result in the data object table: **`variation`**. The data object table shows the explainable variation for the complete field (as average).
- Samples of amplitudes, quantity identifiers: **`amp[quantity name][data type]_shape[shape index]`**.
- Cumulative variation: **`cumulative variation[1]`** … **`cumulative variation[n]`**. The data object table shows the cumulative explainable variation for the complete field (as average) for each shape.
- Individual variation: **`variation[1]`** … **`variation[n]`**. The data object table shows the individual explainable variation for the complete field (as average) for each shape.
- Log messages: SoS determines the fraction of explainable variation that is obtained for individual numbers of scatter shapes and prints this table in the log messages.

8.2.2. Creating Synthetic Random Field Models

The **Field data models** menu option **Random field** > **Create synthetic random field model** creates a random field model based on simple numerical autocorrelation models.

In the dialog box, you set these options:

Ident

A unique identifier for the new quantity.

Data type

Data type of the new synthetic random field model. Choices are **Node** and **Element**.

Desired explained variation

Accuracy of the random field model. For simulation of random designs, 90% might be sufficient. SoS automatically determines the number of scatter shapes required by the random field model to represent the desired variation.

Maximum number of shapes

Upper bound for the number of scatter shapes to be computed for the random field model. This option is used to limit hardware resource usage.

Correlation model

Definition in standard normal space of the numerical autocorrelation model type and autocorrelation length distance (`l`), which you can specify as either an absolute or relative value. The following autocorrelation model types are supported:

- Exponential autocorrelation model using the distance between two nodes, $\vec{x}_i$ and $\vec{x}_k$, defined as:

$$\rho_{i,j}=\exp\left(-\frac{\|\vec{x}_i-\vec{x}_j\|}{l}\right)$$

- Squared exponential autocorrelation model using the squared distance between two nodes, $\vec{x}_i$ and $\vec{x}_k$, defined as:

$$\rho_{i,j}=\exp\left(-0.5\left(\frac{\|\vec{x}_i-\vec{x}_j\|}{l}\right)^2\right)$$

Mean value and standard deviation

Defines the random field model either as homogeneous or a database object. When **Define homogeneous** is selected, the value that you specify to the right is used for both the mean and standard deviation at each mesh point. When **Use data object** is selected, user-defined individual data objects are used for the mean value and standard deviation.

Compute cumulative variations

Select this check box to generate data objects that represent the cumulative explainable variation by the random field model, given a varying number of scatter shapes.

Compute individual variations

Select this check box to generate data objects that represent the individual explainable variation that is explained by each individual scatter shape.

Output Objects for Each Quantity

The output objects generated for each quantity are the same as those for the empirical random field model (p. 99).

Usage

The synthetic random field model offers the possibility of building a random field model even if insufficient data is available to estimate the true correlation structure.

8.2.3. Creating Free-form Variation Models (Beta)

The **Field data models** menu option **Random fields > Create free-form variation model (beta)** creates a random field model based on simple numerical autocorrelation models that have the property that each scatter shape contains a single maximum. This maximum is located at a support point (similar to RBF networks). Neighbouring support points are automatically collected into a single shape. Thus, you can manually define scatter shapes not only around points but also around edges or surface patches.

In the dialog box, you set these options:

Ident

The unique identifier for the new quantity.

Data type

Data type of the new free-form variation model. Choices are **Element** or **Node**.

Number of auto-parameters

Support points to find scatter shapes, placed by the algorithm. A higher number of auto-parameters shortens the autocorrelation length and generally leads to the generation of a higher number of more localized scatter shapes. Manual support points must be specified if set to zero.

Use manual support points

If this check box is selected, the algorithm is to incorporate the support points from a manually created component, in addition to the number of auto-parameters. The manual support points must match the data type selection (**Element** or **Node**).

Magnitude of variation

Create a free-form variation model with homogeneous variation at every mesh point or use a the standard deviation given by a data object, matching the data type selection (**Element** or **Node**).

Show advanced options

If this check box is selected, these advanced options display.

- **Normalized shape of variation patterns**

  the shapes are scaled such that their sum equals the standard deviation. If this check box is cleared, the standard deviation is the upper bound to the sum of the shapes.

- **Use mesh distance**

  If this check box is selected, distances are computed along the mesh edges. If this check box is cleared, the Cartesian coordinates are used to compute the distance between two points.

- **Use mean as parameter**

  If this check box is selected, the mean is associated with a scaling parameter and is the first shape. Use this option to scale a field property proportional to the mean. You cannot require this option if normalized shapes are used because you then might have the same effect when setting all random field scaling parameters constant.

- **Compute individual variations**

- If this check box is selected, the individual variation (explainable variance by a single scatter shape) are computed as additional data objects.

- **Erase degenerated shapes**

  If this check box is selected, automatically created variation patterns having a short correlation length are removed from the model.

- **Compute distance object**

If this check box is selected, the distances between the support points are computed and additional data objects are created. One object is created for each shape, representing the distance field around its support point and a distance field indicating the closest distance between all support points. This can help to debug issues with automatically generated support points.

- **Enforce zero variation at boundary**

  If this check box is selected, SoS tries to enforce a near-zero value at the boundary of the currently active set. Use this option if the active set lies directly next to a fixed region.

  Note:

  Enforcing zero variation at the boundary does not always work perfectly and requires normalized shapes.

- **Boundary to be used for automatic support**

  Distance calculations for automatic support point placement require SoS to identify boundaries. Choices are:

  – **Default**: SoS selects an appropriate mode.

  – **None**: When using manual support points only, no boundary is required.

  – **Nodes on surface and on reference set boundary**: SoS identifies free-floating boundaries.

  – **Nodes on boundary edges**: Use with shell-like meshes have a closed boundary.

- **Mean Value**

  Create a free-form variation model with a homogeneous mean value at every mesh point or use a the mean given by a data object, matching the data type selection (**Element** or **Node**).

Output Objects for Each Quantity

For each quantity, these output objects are generated:

- Random field model
- Individual scatter shapes as results in the data object table: **`shape[1]`**, …, and **`shape[n]`**. The data object table shows the cumulative explainable variation for the complete field (as average) for each shape.
- Explainable variation of the random field model as a result in the data object table: **`variation`**. The data object table shows the explainable variation for the complete field (as average).
- Distance objects:

  – For each shape, one distance object named **`shape_distance[n].`**

  - One object named **`shape_distance_supports`**, indicating the distance between closest support points.

- Log messages: SoS determines the fraction of explainable variation that is obtained for individual numbers of scatter shapes and prints this table in the log messages.

Usage

- The free-form variation model offers the possibility of building a random field model even if insufficient data is available to estimate the true correlation structure, with the additional option of centering scatter shapes on points of particular interest using manual support points.

- The mean and standard deviation from a small number or incomplete measurements can improve the free-form variation model.

8.2.4. Computing Errors for Random Fields

The **Field data models** menu option **Random fields > Compute errors for random fields** validate existing random field models with imported samples.

Before executing error computation:

1. Ensure that a random field model has been calculated for the field quantities selected for error computation.

2. Ensure that validation field samples have been imported for the selected field quantities.

When you select **Field data modelsRandom fieldsCompute errors for random fields**, error computation is performed for active field samples only. The calculated accuracy represents the random field model's ability to reproduce the samples as a relative value. When visualized, 100% means exact reproducibility. Values greater than 100% indicate overestimation of the true value.

After error computations are executed, the following new quantities are added to the database:

- One field quantity named **`accuracy[quantity`** ***`ident`*** **`]`**

- Three scalar quantities named **`err_rms[]`**, **`err_max[]`**, and **`err_min[]`**

  - **`err_rms[]`** stores the root mean square error averaged over the mesh.

  - **`err_max[]`** stores the largest positive local relative difference.

  - **`err_min[]`** stores the largest negative local relative difference

8.3. Field Sample Generation from Imported Data

From the **Field data models** → **Approximate from imported data** menu, you can generate new field samples from imported data:

8.3.1. Generating New Field Samples from a Field-MOP's Scalar Input Parameters and Field Quantities

The **Field data models** menu option **Approximate from imported data > Field-MOP from inputs** generates new field samples from selected scalar input parameters and selected field quantities.

- The respective field-MOP must exist in the database.
- You must select the field quantity of interest.
- Optionally, select a subset of scalar input parameters. Otherwise, SoS automatically uses all scalar input parameters.
- The new field quantity is named **`smoothened[quantity`** ***`ident`*** **`]`**.

Usage

- Simulation of random fields
  - Analyze a random field, including its scalar DOE input parameters.
  - Generate new scalar input parameters outside of SoS, ensuring that you respect the bounds of the scalar input parameters that were used to build the DOE. You can use CSV Files (p. 117).
  - Import these new scalar input parameters.
  - Generate one or more new field designs.
- Restore eroded data

  Instead of the interpolation of adjacent values of missing items, which is the default during import, you can use Karhunen-Loeve expansion in terms of a "statistical interpolation" of the missing data items.

8.3.2. Generating New Field Samples from Amplitudes

The **Field data models** menu option **Approximate from imported data > Random field from amplitudes** generates new field samples from selected amplitudes.

- The respective random field model must exist in the database.
- You must select the complete set of amplitudes, starting at **`[1]`**. For example, select:
  - **`amp[pstrain][element]_shape[1]`**
  - **`amp[pstrain][element]_shape[2]`**
  - **`amp[pstrain][element]_shape[`*`n`*`]`**
- SoS determines the random field models that are associated with the selected amplitudes. For each random field model, the generation of field samples is carried out.
- The new field quantity is named **`smoothened[quantity`** ***`ident`*** **`]`**.

Usage

- Simulation of random fields
  - Analyze a random field.
  - Delete computed amplitudes.
  - Generate new amplitudes (zero mean, unit standard deviation) outside of SoS. You can use CSV Files (p. 117).
  - Import these new amplitudes.
  - Generate one or more new field designs.
- Statistical smoothening

  Eliminate noise, where noise is considered to be the variations associated with small correlation lengths.
- Restore eroded data

  Instead of the interpolation of adjacent values of missing items, which is the default during import, you can use Karhunen-Loeve expansion in terms of a "statistical interpolation" of the missing data items.

8.3.3. Generating New Amplitude Samples from Random Field Data

The **Field data models** menu option **Approximate from imported data > Amplitudes from random field data** generates new amplitude samples from selected field quantities. For each active design, a new set of amplitudes is computed based on the random field model that is associated with the selected field quantity.

- The respective random field models of the selected quantities must exist in the database.
- You must select the complete set of field quantities. For example, if you have a cross-correlated random field model, then you must select all field quantities that are part of the model.
- SoS determines the random field models that are associated with the selected quantities. For each random field model, the generation of amplitude samples is carried out.
- The new scalar quantities are named as follows: **`amp[quantity`** ***`ident`*** **`][`*****`object type`*****`]_shape[`*****`number`*****`]`**.

Usage

- Parametrization of signals: Represent an input signal by few scalar parameters
  - Import signals from a DOE.
  - Create a random field model for the signal.
  - Delete all samples, save the database, and reload the database.

  - Import new signal data.
  - Project new signal samples into subspace of random field amplitudes.
  - Recompute smoothened field data from amplitudes to compare the original signal with the approximative representation.
- Identify meta models between signals as input and scalar responses
  - Import signals from a DOE.
  - Create a random field model for the signal.
  - Export amplitude samples to optiSLang.
  - Create a MOP between the amplitudes and the scalar response.
  - Delete all samples, save the database, and reload the database.
  - Import new signal data.
  - Project new signal samples into subspace of random field amplitudes.
  - Export the new amplitude samples to optiSLang.
  - Approximate response values for the new amplitude samples using optiSLang's MOP
- Also apply this for 3D field data on the input side.

Chapter 9: SoS File Format Specifications

SoS has its own native file formats and supports the use of many other file formats. The following topics supply file format specifications:

Input File Examples

The SoS installation includes simple input files for many supported file format specifications. If an input file example is supplied, the topic for the file format specification supplies the file name and refers you back here for more information. The objective of any input file example is to illustrate the input syntax of keywords supported by the SoS input interface. A FEM solver might refuse this input file as it likely is not exhaustive.

- On Windows, input file examples are in `C:\Users\Public\Dynardo\Statistics on Structures\`*`Version`*`\Examples\file_formats`.
- On Linux, the destination path of your installation contains input file examples in the folder `Examples`.

Note:

You can select **Help** → **Open examples folder** to quickly open the `Examples` folder.

9.1. File Formats Overview

Tables supply basic file format information and indicate what capabilities are supported. After the tables, some general notes are provided.

Native File Formats

File Format	Import	Export
SoS database (`.sdb`)	Yes	Yes
SoS script (`.ssc`)	Yes	Yes
SoS simulation archive (`.sim`)	No	Yes

Scalar Parameter Files

File Format	Import	Simulate	Export
optiSLang (p. 134) (`.bin`)	Yes	No	Yes
CSV (p. 118) (`.csv`)	Yes	Yes	Yes

FEM Mesh and CAE Input Files

File Format	Import	Simulate	Export	FEM Types
2D-Grid (`.csv`)	Yes	No	No	grid (2D)
3D-Grid (.csv)	Yes	No	No	grid (3D)
Abaqus (p. 116) (`.inp`)	Yes	Yes	No	triangles (3n, 6n)
				quadrangles (4n, 8n, 9n)
				tetrahedra (4n)
				prism (6n)
				hexahedra (8n, 20n)
ANSYS binary result (p. 111) (`.rst`)	Yes	No	No	quadrangles (4n, 8n)
				tetrahedra (4n, 10n)
				hexahedra (8n, 20n)
ANSYS Mechanical (p. 113) (`.dat`, `.cdb`)	Yes	Partial	Partial	quadrangles (4n, 8n)
				tetrahedra (4n, 10n)
				hexahedra (8n, 20n)
Images (p. 125) (`.png` and many more)	Yes	Yes	No	grid (2D)
LS-DYNA (p. 126) (`.k`, `.dynain`)	Yes	Yes	Yes	triangles (3n)
				quadrangles (4n)
				tetrahedra (4n)
				hexahedra (8n)
Nastran (p. 131) (`.dat`, `.bdf`, `.nas`)	Yes	Yes	Yes	triangles (3n, 6n)
				quadrangles (4n, 8n)
				tetrahedra (4n, 10n)
				pentahedra (6n, 15n)
				hexahedra (8n, 20n)
PERMAS (p. 136) (`.dat`)	Yes	Yes	No	triangles (6n)

File Format	Import	Simulate	Export	FEM Types
Signal mesh files (p. 123) (.csv)	Yes	Yes	No	grid (1D)
Simple legacy VTK ASCII (p. 138) (`.vtk`)	Yes	No	No	triangles (3n)
				quadrangles (4n)
				tetrahedra (4n)
				hexahedra (8n)
STL (p. 137) (`.stl`)	Yes	Yes	Yes	triangles (3n)

FEM Result Data Files

File Format	Import	Simulate	Export
ANSYS binary result (p. 111) (`.rst`)	Yes	No	No
CSV (p. 119) (`.csv`)	Yes	Yes	Yes
LS-PrePost (p. 129) (`.k`)	Yes	No	No
PAM-STAMP (p. 135) (`.asc`)	Yes	No	No
Signal data files (p. 123) (`.csv`)	Yes	Yes	No

General Notes

- File formats that are not supported or missing file format features can possibly be implemented on request. To request enhanced support for file formats, contact ANSYS Support (p. 12).
- Many CAE preprocessors and postprocessors support the implemented file formats and might also be able to export them partially in the respective format. For example, Autoform is supported by exporting to LS-DYNA.
- The SoS data import interface is not necessarily bound to the input import interface format. For example, you can use the SoS Nastran input import interface to import the CAE input or mesh file or use the SoS LS-PrePost data import interface to import the response data file.
- Because SoS's ability to write a format file from beginning to end is very limited, you typically modify existing files.

9.2. ANSYS Mechanical Files

The following topics explain how ANSYS Mechanical files can be used in SoS:

9.2.1. Importing ANSYS Binary Result Files

SoS supports only the ANSYS-written structural or coupled-field analysis binary result file format specification given in [2] (p. 179).

A typical file extension for this binary results file is `.rst`. Furthermore, only a very limited number of dataset identifiers at the very last converged sub-step of the very last converged load-step might be imported.

The results file format specification given in [2] (p. 179) lists a multitude of data structures and member variables to be read. However, SoS only partially supports them. The following list of limitations is not exhaustive:

- SoS supports the binary file format specification used in ANSYS version 16 and later.
- Only the very first geometry data header is used. Changing meshes during simulation is therefore not supported and should invoke an error.
- The topology definition is limited to supported element types, which are listed in the subsequent table.
- The SoS import interface reads only a very limited number of predefined dataset identifiers, which are listed in the subsequent table, at the very last converged sub-step of the very last converged load-step.

Element Routines

The table lists the element routines that are supported. For more information, see [1] (p. 179).

Element Routine	Description
`PLANE182`	4 Node Linear 2-D Structural Solid
`PLANE183`	8 Node Quadratic 2-D Structural Solid
`SHELL28`	4 Node Structural Linear 3-D Shear/Twist Panel
`SHELL41`	4 Node Structural Linear 3-D Membrane Shell
`SHELL181`	4 Node Structural Linear Shell
`SHELL281`	8 Node Structural Quadratic Shell
`SOLID5`	8 Node 3-D Linear Coupled-Field Solid
`SOLID65`	8 Node 3-D Linear Reinforced Concrete Structural Solid
`SOLID70`	8 Node 3-D Linear Thermal Solid
`SOLID87`	10 Node 3-D Quadratic Tetrahedral Thermal Solid
`SOLID90`	20 Node 3-D Quadratic Thermal Solid
`SOLID96`	8 Node 3-D Linear Magnetic Scalar Solid
`SOLID97`	8 Node 3-D Linear Magnetic Solid
`SOLID98`	10 Node 3-D Quadratic Tetrahedral Coupled-Field Solid
`SOLID122`	20 Node 3-D Quadratic Electrostatic Solid
`SOLID123`	10 Node 3-D Quadratic Tetrahedral Electrostatic
`SOLID164`	8 Node 3-D Linear Explicit Structural Solid
`SOLID168`	10 Node 3-D Quadratic Explicit Tetrahedral Structural Solid
`SOLID185`	8 Node 3-D Linear Structural Solid

Element Routine	Description
`SOLID186`	20 Node 3-D Quadratic Structural Solid
`SOLID187`	10 Node 3-D Quadratic Tetrahedral Structural Solid
`SOLID226`	20 Node 3-D Quadratic Coupled-Field Solid
`SOLID227`	10 Node 3-D Quadratic Coupled-Field Solid
`SOLID231`	20 Node 3-D Quadratic Electric Solid
`SOLID232`	10 Node 3-D Quadratic Tetrahedral Electric Solid
`SOLID236`	20 Node 3-D Quadratic Electromagnetic Solid
`SOLID237`	10 Node 3-D Quadratic Electromagnetic Solid
`SOLID239`	20 Node 3-D Quadratic Diffusion Solid
`SOLID240`	10 Node 3-D Quadratic Tetrahedral Diffusion Solid
`SOLID278`	8 Node 3-D Linear Thermal Solid
`SOLID279`	20 Node 3-D Quadratic Thermal Solid
`SOLSH190`	8 Node 3-D Linear Structural Solid Shell

Dataset Identifiers

The table lists supported dataset identifiers. They are read at the last converged sub-step of the last converged load-step.

Dataset Identifier	Description
`nodal displacements`	The degrees of freedom solution for each node in the nodal coordinate system. SoS supports only a limited number of degrees of freedoms per node including: **Ux**, **Uy**, **Uz**, **ROTx**, **ROTy**, **ROTz**, **Ax**, **Ay**, **Az**, **Vx**, **Vy**, **Vz**, **WARP**, **CONC**, **HDSP**, **PRES**, **TEMP**, **VOLT**, and **MAG**. For more information, see [2] and other relevant ANSYS guides.
`nodal component stresses`	Contains the stresses **SX**, **SY**, **SZ**, **SXY**, **SYZ**, and **SXZ** at each corner node.
`element nodal component stresses`	Contains the stresses **SX**, **SY**, **SZ**, **SXY**, **SYZ**, and **SXZ** at each corner node.

9.2.2. Importing ANSYS Mechanical Input Files

SoS supports the ANSYS Mechanical input file format specifications defined in [3] (p. 179) and the blocked and unblocked coded database format defined in [4] (p. 179). Support is limited to only a few commands and some of their options.

A typical file extension for an ANSYS Mechanical input file is `.dat`. The file extension for the coded database format is usually `.cdb`.

Compared to the ANSYS Mechanical input file syntaxes in [3] (p. 179) and [4] (p. 179), SoS supports only a very restricted number of syntax elements. The following list, not exhaustive, includes some of the limitations:

- All characters of any alphabetic or alphanumeric command and of any command option name must be given. You cannot use only the first four characters. SoS supports only a few ANSYS Workbench abbreviations.
- All supported commands are valid in all processors. All other commands are ignored.

Command	Description	Supported Options	Comments
CLOCAL	Defines a local coordinate system relative to the active coordinate system	—	Support is limited to track the active coordinate system only.
CMBLOCK	Defines the entities contained in a node or element component	**Cname**, **Entity**, **NUMITEMS**	—
CS	Defines a local coordinate system by three node locations	—	Support is limited to track the active coordinate system only.
CSKP	Defines a local coordinate system by three key point locations		Support is limited to track the active coordinate system only.
CSWPLA	Defines a local coordinate system at the origin of the working plane		Support is limited to track the active coordinate system only.
CSYS	Activates a previously defined coordinate system	—	Support is limited to track the active coordinate system only.
EBLOCK	Defines a block of elements	**NUM_NODES**, **SOLKEY**	—
ET	Defines a local element type from the element library	—	Element name (or number) support is limited to the list given in Importing ANSYS Binary Result Files (p. 111). Other element types are listed as unsupported at the end of the import process.
N	Defines a node	—	Only the default Cartesian coordinate system is supported. The N command in its slightly different unblocked coded database syntax is supported as well
NBLOCK	Assigns nodes to a node set	**NUMFIELD**, **SOLKEY**, **NDMAX**, **NDSEL**	The final line of the block always has a **-1** for the node number. This is given either in the unblocked N command syntax as typical for the **CDWRITE** command or in the **NBLOCK** command syntax as the first token.

Command	Description	Supported Options	Comments
			ANSYS Workbench uses the latter approach.
LOCAL	Defines a local coordinate system by a location and orientation		Support is limited to track the active coordinate system only.
***SET**	Assigns values to user-named parameters	—	The equivalent **Par = VALUE** format is also supported and translated internally into the ***SET** command. Due to the lack of the ***DIM** star set command, only scalar values are supported. Consequently, only one value per line is supported. Other assignments are skipped. **Par** identifier restrictions are reduced to the character length only. Other violations are not verified.
\TITLE	Defines a main title	—	—

Note:

The SoS installation includes a simple ANSYS Mechanical input file: `ansys.cdb`. For more information, see Input File Examples (p. 109).

9.2.3. Simulating ANSYS Mechanical DAT Files

ANSYS Workbench uses `ds.dat` files to transfer data to ANSYS Mechanical. These files contain APDL commands that describe and set up the numerical problem to be solved. SoS is using the same mechanism to generate random fields. You need only to input the SoS-generated DAT file before your solution step in ANSYS Mechanical. You must include this DAT file after the creation of the mesh. Otherwise, ANSYS Mechanical replaces the node coordinates that SoS exported with its predefined definition.

SoS always rewrites the DAT file, ignoring its original contents. SoS can use DAT files to export the following data to ANSYS Mechanical:

- Node coordinates: x, y, z
- Node coordinate deviations (with respect to the reference mesh): x, y, z, normal

You can use this command to import the DAT file:

```
/input,../../sos_output.dat
```

The DAT file already switches into **/PREP7** mode.

9.3. Abaqus Files

SoS supports the Abaqus file format specification defined in [5] (p. 179). Support is limited to a few keywords and some of their parameters in the *model data* part only. Due to the lack of proper keyword support, SoS does not support an assembly of part instances.

The following topics explain how to Abaqus files can be used in SoS:

9.3.1. Importing Abaqus Mesh Input Files

A typical file extension for Abaqus input files is `.inp`, although the SoS Abaqus import interface accepts all file extensions and does not distinguish between them.

Compared to the Abaqus syntax in [5] (p. 179), SoS supports only some of the syntax elements. The following list, not exhaustive, includes some current limitations:

- The input file does not need a carriage return at the end of each line.
- Keywords, parameters, and parameter values must be spelled out completely. SoS does not support auto-completion.
- The same parameter should not appear more than once on a single keyword line. If a parameter has multiple settings on a single keyword line, SoS ignores all but the last occurrence.
- SoS does not support the **`INPUT`** parameter.
- SoS supports floating point formats that are written in C syntax only. The following floating point number, which is legal in the Abaqus format specification, causes an error in the SoS Abaqus import interface:

  ```
  -1234.5D-2
  ```
- SoS does not support data line repetition.
- SoS does not support model data that is organized in an assembly of part instances.
- Because SoS does not support the **`*INCLUDE`** keyword, it cannot support specifying an external file that contains a portion of the Abaqus input file, especially the node definition.

Due to limitations of the Abaqus input interface, each passed Abaqus file must contain a non-empty model data definition. This definition can be given only with the keywords and their parameters in the following table. All other keywords are ignored. For example, the **`*NODE`** keyword can refer only to the implicitly defined Cartesian coordinate system. No other coordinate system can be defined due to the lack of keyword support. If an unsupported keyword parameter is used, the interface is likely to stop and display an error.

Keyword	Description	Supported Parameters	Comments
`*NODE`	Defines a node directly by specifying its coordinates	`NSET`	—
`*NSET`	Assigns nodes to a node set	`NSET`, `GENERATE`, `UNSORTED`	Assembly level node sets are not supported. Neither is using the `INSTANCE` parameter nor prefixing each node number with the part instance name and a period (`.`).
`*ELEMENT`	Defines an element directly by specifying its nodes	`TYPE`	`TYPE` support is limited to the stress/displacement elements `C3D4`, `C3D6`, `C3D8`, `C3D10`, `C3D20`, and all of their variants. Supported shells are `S3(R)`, `S4(R)`, `STRI65`, and `S8(R/RT/R5)`. Keywords with other `TYPE` values are ignored.
		`ELSET`	

Note:

The SoS installation includes a simple Abaqus input file: `abaqus.inp`. For more information, see Input File Examples (p. 109).

9.3.2. Simulating Abaqus Mesh Input Files

The SoS Abaqus simulate interface cannot write a valid Abaqus input file from start to finish. It can only modify valid Abaqus input files that:

- Meet the requirements given in Importing Abaqus Mesh Input Files (p. 116)
- Are compatible with the project-related reference mesh

Typically, the simulate feature expects to modify one of the SoS project-related valid mesh input files.

The Abaqus export interface is capable of modifying only the following fields of the **`*NODE`** keyword:

- x coordinate in the basic coordinate system
- y coordinate in the basic coordinate system
- z coordinate in the basic coordinate system

9.4. CSV Files

SoS can handle data objects in a CSV (comma-separated value) file. The SoS CSV interface tries to follow the format specification given by the RFC4180 memo in [15] (p. 179) and adds some program-related ones. A typical file extension for a CSV file is `.csv`.

The following topics explain how CSV files can be used in SoS:

The following list summarizes the CSV file format specifications accepted by the SoS CSV reader interface:

- Each record is located on a separate line, delimited by a line break (CRLF).
- The last record in the file may or may not have an ending line break (CRLF).
- There may be an optional or even required header line appearing as the first line of the file with the same format as normal record lines. The optional header record must contain the same number of fields as the longest record in the entire file. Its fields are interpreted as unique data object names corresponding to the fields in the file. Non-unique names are made unique by appending an underscore followed by a serial number.
- Within the header and each record, there may be one or more fields, separated by a delimiter. Except for the header record, each record may consist of a different number of fields or no fields at all. SoS interprets these missing fields as missing data. Spaces are considered part of a field. The last field in the record must not be followed by a delimiter.
- SoS is generally able to determine the delimiter automatically when one of these common delimiter characters is used: `,`, `;`, `:`. It also recognizes a tabbed space as a delimiter. You can manually change the suggested delimiter at any time.
- Each field may or may not be enclosed in double quotes.
- Fields containing double quotes and the delimiter must be enclosed in double quotes.
- Fields containing line breaks (CRLF), even those enclosed in double quotes, are not supported.
- If double quotes are used to enclose fields, then a double quote appearing inside a field must be escaped by preceding it with a backslash character (`\`).

9.4.1. Importing Scalar Data from CSV Files

SoS can import scalar data objects from a CSV file. In addition to the CSV file format specifications summarized in the previous topic, the CSV scalar data import interface has these restrictions:

- The CSV file must have only one header line, appearing as the first line. The header line must have the same format as normal record lines and contain as many fields as the longest record in the rest of the file.

- A header record with the hash character (#) in the first field indicates that the first field of each record in the file has to be read and interpreted as an integer design number.
- Empty files are accepted.

The following example shows scalar data in a CSV file:

```
#, first param, "second param"
0, 1.34, 1.345e5
2, 2.45
3,

10,,4.56e5
```

9.4.2. Exporting Scalar Data from CSV Files

SoS can write a valid CSV data file from beginning to end following the CSV file format specification summarized in CSV Files (p. 117).

By default the SoS CSV export interface chooses a comma (`,`) as the field delimiter and writes the design number in the first field of each record as stated by the SoS database. You cannot modify the digit precision of real numbers determined by their machine precision.

If a scalar data object to be exported contains a missing data item, the value remains empty in the CSV file.

9.4.3. Importing Field Data from CSV Files

SoS can import field data objects from a CSV file. In addition to the CSV file format specifications summarized in CSV Files (p. 117), the CSV field data import interface has these restrictions:

- There may be an optional header line appearing as the first line of the file with the same format as normal record lines. The optional header record must start with the hash character (#) and must contain the same number of fields as the longest record in the entire file. Fields after the hash character are interpreted as unique field quantity object names corresponding to the fields in the file. Non-unique names are made unique by appending an underscore followed by a serial number.
- If a header line is present, multiple-field data objects might be imported:
  - Records must start with the first data value of the first quantity identifier in the first field.
  - Data values are interpreted as double.
  - If the first field is interpreted as missing data, the record must start with the delimiter.
  - Missing fields, empty lines, or missing lines at the end of the file are interpreted as missing data.
  - Records must not contain any part or item ID.
  - Data values are automatically associated to their quantity idents given in the header line.

  - All data values are automatically associated to part ID 0 and an automatically generated item ID.
  - Item identifiers start counting with 0 and are incremented continuously and strictly monotonously with each new line.
- If no header line is present, only a single-field data object can be imported. The single mode allows you more control regarding the identifier associated with the data value:
  - Each record can have up to three fields. The data are interpreted as an optional part ID, optional item ID, and associated data value.
  - If no optional part identifier field is specified, the CSV field data import interface assumes the part ID is **`0`** for all records.
  - If only the data value column is provided, the CSV field data interface starts associating each value with an identifier starting at **`0`** (equivalent to line number minus 1).
  - Each record must count the same number of fields, separated by a delimiter. A record may also be completely empty. SoS interprets these missing fields as missing data. Spaces are considered part of a field. The last field in the record must not be followed by a delimiter.
  - Any optional identifier field must be convertible to an integer. The data value is interpreted as double.

Field Data Best Practice

- Without any indexing column, it is best practice to insert an empty record line after the header, avoiding potential zero-index issues between different FEM mesh formats.
- For tight adherence to the CSV specification, it is best practice to avoid white spaces and use only the comma (**`,`**) as the delimiter.

```
#field1,field2,field3
,,
0.1,0.1,0.1
0.2,0.2,0.2
```

Field Data Example Files

The following examples define the data values for the item identifiers 0 through 4 in part 0 in all valid format descriptions. Example 4 adds another data value for a different part just to demonstrate it.

Example 1: Uses multiple-field data mode

```
# s_eqv, temp, react
1.23, 3.45, 25.23

, , 23.457
14.5, 4.56, 0
5,23e-3, 5.17,
```

Example 2: Uses data value column only in single-field data mode

```
3.45

4.56
5.17
```

Example 3: Uses optional item identifier and data value columns in single-field data mode

```
0, 3.45

3, 4.56
4, 5.17
```

Example 4: Uses full-format specification in single-field data mode, namely the optional part and the item identifier and data value columns

```
0; 0; 3.45
0; 3;    4.56
0; 4;5.17
3;20;28.3333333
```

9.4.4. Writing Field Data to CSV Files

SoS can write field data to CSV files. This topic describes the CSV file formats used in Modifying and Exporting Multiple Field Designs (p. 70) and Generating Field Objects in optiSLang for Use in a Simulation (p. 74) when selecting **Create new file**.

Generic Text Format for Node and Element Data

SoS writes a CSV file using data from the currently loaded database. You cannot modify the digit precision of real numbers, which is determined by machine precision. Each file's content is limited to a single field data quantity.

Given the CSV file format specification in CSV Files (p. 117), the subsequent example file is written:

```
part-ID;[type-master-ID,]type-ID;double
part-ID;[type-master-ID,]type-ID;double
...
```

SoS writes the following data to each record:

- Part identifier in the first field
- Item identifier prefixed by an optional master identifier in the second field
- Field data as a double precision value in the last field
- No record is written for missing data items

ANSYS External Data for Displacements

The displacement field is exported for all of the mesh's nodes, ignoring the reference node set. You can export displacement fields in the normal, x, y, or z direction (defined in the reference mesh's coordinate system). Ithe option for mesh smoothening is selected, it is applied to displacement fields.

The CSV file uses a comma (`,`) as the delimiter and consists of the following columns:

- x coordinate of reference mesh
- y coordinate of reference mesh
- z coordinate of reference mesh
- x displacement
- y displacement
- z displacement

If you are interested in only the boundary nodes, you should import only the boundary of the mesh (through an STL file for example) and then export only this data.

ANSYS External Data for Node Data

For all nodes in the reference node set, the field of the specified quantity is exported.

The CSV file uses a comma (`,`) as the delimiter and consists of the following columns:

- x coordinate of each FEM node in the reference mesh
- y coordinate of each FEM node in the reference mesh
- z coordinate of each FEM node in the reference mesh
- Field data of each FEM node if chosen for export

ANSYS External Data for Element Data

For all elements in the reference element set, the field of the specified quantity is exported.

The CSV file uses a comma (`,`) as the delimiter and consists of the following columns:

- x center coordinate of each FEM element in the reference mesh
- y center coordinate of each FEM element in the reference mesh
- z center coordinate of each FEM element in the reference mesh
- Field data of each FEM element if chosen for export

9.4.5. Exporting Field Data Objects

SoS can export any field data objects in the data table via the Exporting Selected Field Data Objects to a CSV File (p. 68) option on the **Edit** menu.

A file is written according to the CSV file format specification in CSV Files (p. 117). The following example includes optional fields within brackets:

```
[Indices,][Coordinates,]quantity ident, quantity ident, ...
[part-ID,][type-master-ID,][type-ID,]design ident, design ident ...
[integers, ... ]double, double, ...
```

Each column represents one field data object:

- `quantity ident` in line 1
- `design ident` in line 2
- Lists the data type value at each mesh position as a double value in subsequent lines

Depending on user settings, the data output is prefixed with additional integers, such as the node identifier and part identifier if several parts exists. You can remove the header lines.

9.4.6. Importing 1D Grid Data Files (Signal Data)

SoS can import CSV-formatted 1D grid data files, also known as signal data files, including their data values. In addition to the specification requirements and limitations summarized in CSV Files (p. 117), the CSV import interface adds these requirements:

- You must not have a header line.
- Each record can have up to three fields. The fields are interpreted respectively as an optional node identifier, time-stamp value, and associated data value.
- Each record must have the same number of fields, separated by a delimiter.
- 1D grids do not support missing data. Therefore, empty records are not supported because SoS would need to interpret missing fields as missing data.
- Spaces are considered part of a field. The last field in the record must not be followed by a delimiter.
- The optional node identifier must be convertible to an integer. The other data value is interpreted as double.

1D Grid Example Files

The following examples define the mesh and associated data values for the item identifiers 1 through 4 in all valid format descriptions.

Example 1: Uses time stamp and data value columns

```
1e-5, 3.45
4.45, 4.56
5, 5.17
8e1  ,   2
```

Example 2: Uses full-format specification, namely the optional node identifier and time stamp and data value columns

```
 1 , 1e-5, 3.45
2,4.45, 4.56
3,5, 5.17
4,8e1  ,   2
```

9.4.7. Simulating 1D Grid Data Files (Signal Data)

Simulating 1D grid or signal data files means modifying their data values with values from the actual database loaded into SoS. Because SoS can write a valid CSV data file from beginning to end following the format specification given in CSV Files (p. 117), SoS does not modify any given CSV file but rather always overwrites it.

For signal data, SoS overwrites the given CSV file with the former time stamp value and its usually simulated data value. You cannot modify the digit precision of real numbers determined by their machine precision.

9.4.8. Importing 2D Grid Data Files (Signal Data)

SoS can import CSV-formatted 2D grid data files, also known as signal data files, including their data values. In addition to the specification requirements and limitations summarized in CSV Files (p. 117), the CSV import interface adds these requirements:

- No header line
- 3 columns (x, y, and z)
- 2D grid is parametrized two axes in Cartesian x and y direction
- Dimension of grid is auto-detected

Grid points are ordered as in the following example. First the x coordinates are changed while keeping y constant, and then y is increased.

```
-1.00;-1.00;2.00
0.00;-1.00;2.00
1.00;-1.00;2.00
-1.00;0.00;2.00
0.00;0.00;2.50
1.00;0.00;2.50
-1.00;1.00;2.00
0.00;1.00;2.50
1.00;1.00;2.50
```

In this example, the rows are associated with the (i,j) value pairs:

```
0,0
1,0
2,0
0,1
1,1
2,1
0,2
1,2
2,2
```

9.4.9. Importing 3D Grid Data Files (Signal Data)

SoS can import CSV-formatted 3D grid data files, also known as signal data files, including their data values. In addition to the specification requirements and limitations summarized in CSV Files (p. 117), the CSV import interface adds these requirements:

- No header line
- 3 columns (x, y, and z)
- 3D grid is parametrized by three axes in Cartesian x, y, and z direction
- Dimension of the grid is auto-detected

Grid points are ordered as in the following example. First the x coordinates are changed while keeping y and z constant, and then y is increased while keeping z constant. Lastly, z is increased.

```
-1.00;-1.00;1.00
0.00;-1.00;1.00
1.00;-1.00;1.00
-1.00;0.00;1.00
0.00;0.00;1.00
1.00;0.00;1.0
-1.00;1.00;1.00
0.00;1.00;1.00
1.00;1.00;1.00
-1.00;-1.00;2.00
0.00;-1.00;2.00
1.00;-1.00;2.00
-1.00;0.00;2.00
0.00;0.00;2.00
1.00;0.00;2.0
-1.00;1.00;2.00
0.00;1.00;2.00
1.00;1.00;2.00
```

9.5. Image Files

The SoS image interface builds on the Qt application framework, which supports several image formats by default (such as `.jpg` and **`.png`**), depending on the operating system on which it is running.

The following topics explain how image files can be used in SoS:

9.5.1. Importing Image Files as FEM Mesh

The SoS image import interface interprets each image pixel as a node. The visualization displays shell elements interconnecting the pixels (nodes).

Note:

The image preview in SoS shows only the mesh. It does not show the data. Consequently, an imported image might appear only as a gray area and not as the image you expect.

9.5.2. Importing Image Files as Data Files

The SoS image import interface can interpret the ARGB color channels and their respective gray value, which is calculated out of the RGB color space. These channels are imported as field quantities for each design. Each color channel is represented by a value in the interval from 0 to 1.

9.5.3. Simulating Image Files

The SoS image simulate interface cannot write a valid image file from beginning to end. It can only modify a valid image file.

The image simulate interface can modify the whole ARGB color space of each pixel. To do so, independently from the image format, the export interface needs to convert the way that the color information is stored in your picture (such as monochrome, gray, 16-bit versus 8-bit color channels, existence of alpha channel, and so). This procedure does not affect the image format of the file (such as `.jpg` or `.png`).

9.6. LS-DYNA Files

LS-DYNA supports a large variety of keywords, each including numerous options. SoS supports only a limited number of keywords and options for data import and export.

The following topics explain how LS-DYNA files can be used in SoS:

9.6.1. Importing LS-DYNA Input Files

Typical file extensions for LS-DYNA input files are `.k` or `.dynain`, although the SoS LS-DYNA import interface accepts all file extensions.

With respect to the input file format specifications given in [9] (p. 179), SoS supports only some of its syntax elements. The following list of limitations is not exhaustive:

- The SoS LS-DYNA import interface supports the *keyword input format* only.
- The SoS LS-DYNA import interface reads input decks from the first line onward. It does not require the **`*KEYWORD`** card. An optional memory size specified after the word `KEYWORD` is ignored. You cannot control the memory size that is used.
- The entire LS-DYNA input deck is order-independent, with the exception of the optional keyword **`*END`**, which can define the end of the input deck before encountering the end of the file.
- Other than stated in [9] (p. 179), the general keyword card format is not limited to 80 characters. The SoS LS-DYNA import interface reads until reaching a line break sequence.
- SoS supports the *fixed standard format* only. It does not support *CSV* or *long-formated* input cards. Format switches can result in parsing errors.

- The usage of the format indication signs **+** and **−** are ignored. Fixed standard format is still read.

Due to LS-DYNA input interface limitations, each passed input file must contain a valid finite element mesh definition. This definition can be given with the following keywords, which do not have any extra keyword options. All other keywords are ignored.

Keyword	Description
`*NODE`	Defines a node in the global coordinate system. Options are not supported
`*ELEMENT_SHELL_{THICKNESS}`	Defines three, four, six, and eight node elements, whereas SoS supports only three and four node elements. The option **`THICKNESS`** is supported.
`*ELEMENT_SOLID`	Defines 3D solid elements including 4-noded tetrahedrons and 8-noded hexahedrons. No options are supported. For elements with 4 or 8 nodes, the cards in the format of LS-DYNA versions 940-970 are still supported.
`*INITIAL_STRAIN_SHELL`	Initializes strain tensor for a shell element. No options are supported.
`*INITIAL_STESS_SHELL`	Initializes stresses, history variables, and the effective plastic strain for a shell element. No options are supported.
`*SET_NODE_{LIST}_{TITLE}`	Defines groups of nodes. Two options are supported: **`LIST`** and **`TITLE`**.

Note:

The SoS installation includes a simple LS-DYNA input file: `lsdyna.k`. For more information, see Input File Examples (p. 109).

9.6.2. Simulating LS-DYNA Input Files

The SoS LS-DYNA simulate interface can simulate valid LS-DYNA input files that:

- Meet the requirements given in Importing LS-DYNA Input Files (p. 126).
- Are compatible with the project-related reference mesh

Typically, the simulate feature expects to modify one of the SoS project-related valid LS-DYNA input files.

The LS-DYNA simulate interface can modify only the following cards of the following keywords, including their options:

Keyword	Description
`*NODE`	The x-, y-, and z-coordinate in the global coordinate system.
`*ELEMENT_SHELL_THICKNESS`	The shell thickness of all nodes as a constant value for all element nodes.
`*INITIAL_STRAIN_SHELL`	All (**`ij`**) strain components.

Keyword	Description
`*INITIAL_STRESS_SHELL`	All (`ij`) stress components. Plastic strain.

9.6.3. Exporting LS-DYNA Mesh Input Files

As mentioned in Exporting a Geometry (p. 65), the SoS LS-DYNA export interface cannot write a ready-to-use LS-DYNA input file from start to finish because the SoS kernel does not provide the information needed to do so. The interface exports only the mesh definition and any existing node or element set that meets the requirements in Importing LS-DYNA Input Files (p. 126).

The export interface uses the following keywords on output:

Keyword	Description
`*KEYWORD`	Indicates the start of an LS-DYNA mesh file.
`*TITLE`	Outputs the project title if not empty.
`*ELEMENT_SHELL`	Translates SoS shell element definitions into LS-DYNA shell element definitions. Check the comment line in the output file.
`*ELEMENT_SOLID`	Translates SoS solid element definitions into LS-DYNA solid element definitions. Check the comment line in the output file.
`*NODE`	Outputs the x, y, and z coordinates in the global coordinate system.
`*SET_NODE_LIST_{TITLE}`	Exports node sets using their set ID equally in LS-DYNA if, and only if, it can be converted into a valid numerical LS-DYNA set identifier. Otherwise, SoS generates a unique identifier using the set identifier from SoS as optional **`_{TITLE} in LS-DYNA`**.
`*SET_SHELL_LIST_{TITLE}`	Splits element sets of SoS into element type-dependent set definitions in LS-DYNA and handles set identifier assignments as per the description for **`*SET_NODE_LIST_{TITLE}`**.
`*SET_SOLID_LIST_{TITLE}`	See the description for **`*SET_SHELL_LIST_{TITLE}`**.

The export interface tries to keep SoS indices as long as possible, intervening in the following cases only:

- Index spaces starting in SoS at **`0`** are shifted by one to meet LS-DYNA requirements.
- If SoS contains several parts, the exporter interface automatically translates the index space into a continuous one. Furthermore, it generates node sets for each part, including all nodes of the respective part in a separate node set. Node sets generated this way have indices starting one after the last node set definition and an optional title set to the respective part identifier if not empty.

For titles written by the export interface, SoS does not translate any identifier into the restricted character set supported by LS-DYNA.

9.7. LS-PrePost Files

Many LS-DYNA preprocessors and postprocessors exist, such as ANSYS or LS-PrePost. Because LS-PrePost is free and widely used, it serves as a reference data file format specification. The typical file extension for an LS-PrePost output data file is `.k`.

You can import LS-PrePost output files into SoS. While the data file format requirements for importing these output files are very similar to those given for importing LS-DYNA input files (p. 126), SoS requires that the following specifications also be met:

- The LS-PrePost data interface skips all content up to the required **`*KEYWORD`** card.
- The LS-PrePost data interface ignores the optional **`*END`** card and continues parsing until the end-of-file marker is encountered.
- The number of characters per input line is not limited and is terminated by the first line break sequence.
- Fields within one line are separated by one or more spaces.

Of the many data output types provided by LS-PrePost, the SoS LS-DYNA import data interface supports importing the output data structures that follow the general steps that you must perform to get a result file within LS-PrePost:

1. Open a LS-DYNA `d3plot` project file.
2. Select the state or time at which the data is to be exported.
3. Plot the so-called fringe component that you want to export to SoS.

   For example, in the command line, type either **`fringe 7`** to print the plastic strain data at the current state or time or **`fringe 68`** to print the thickness reduction.
4. Write the data to an output file using either the GUI menu in [POST] -> [OUTPUT] or one of the commands that follow for the data structures that are supported by the SoS LS-DYNA import interface.

*NODAL_RESULTS

- The next line starts with **`$RESULT OF`**, followed by the data identifier.
- The following lines are expected to contain the nodal identifier, separated from the actual nodal scalar data value at a specific finite element solver state or time, such as a nodal displacement or stress or strain component.

```
*KEYWORD
$TIME_VALUE = 1.0099823e-003
$STATE_NO = 102
$Output for State 102 at time = 0.00100998
*END
$NODAL_RESULTS
$RESULT OF Y-displacement
         1       0.0
         2 1.3924E+1
```

- To create this output file using LS-PrePost, you must perform the general steps given earlier and include this:

```
fringe 18
output "absolute or relative dir path\nodal_y-displacement.k" 102 1 0 1 0 0 0 0 0 1 0 0 0 0 0 1.000000
```

*NODAL_DISPLACEMENT

- The following lines are expected to contain the nodal identifier, separated from the actual nodal displacements in the x, y, and z directions at a specific finite element solver state or time:

```
*KEYWORD
$TIME_VALUE = 1.0099823e-003
$STATE_NO = 102
$Output for State 102 at time = 0.00100998
*END
$NODAL_DISPLACEMENT
       1  0.0000000e+000  0.0000000e+000 -1.7422426e+001
       2  0.0000000e+000  1.3924263e+001 -1.9328943e-002
```

- To create this output file using LS-PrePost, you must perform the general steps given earlier and include this:

```
output "absolute or relative dir path\displacements.k" 102 1 0 1 0 0 1 0 0 0 0 0 0 0 0 1.000000
```

*SHELL_ELEMENT_RESULTS

- The next line starts with **`$RESULT OF`**, followed by the data identifier.
- The following lines are expected to contain the element identifier, separated from the actual element scalar data value at a specific finite element solver state or time, such as an element shell thickness (reduction) or von Mises stress.

```
*KEYWORD
$TIME_VALUE = 1.0099823e-003
$STATE_NO = 102
$Output for State 102 at time = 0.00100998
*END
$SHELL_ELEMENT_RESULTS
$RESULT OF Shell Thickness (Unaveraged)
       1  9.0582311e-002
       2  8.7422013e-002
```

- To create this output file using LS-PrePost, you must perform the general steps given earlier and include this:

```
fringe 67
output "absolute or relative dir path\shell_thickness.k" 102 1 0 1 0 0 0 0 1 0 0 0 0 0 0 1.000000
```

*ELEMENT_SHELL_PRINCIPAL_TRUE_STRAIN

- The following lines are expected to contain the element identifier separated from the actual element scalar data value at a specific finite element solver state or time.

```
*KEYWORD
$TIME_VALUE = 1.0099823e-003
$STATE_NO = 102
$Output for State 102 at time = 0.00100998
*END
$ELEMENT_SHELL_PRINCIPAL_TRUE_STRAIN()
$Elem Id   Lower Major   Lower Minor   Upper Major   Upper Minor    Thickness    Reduction
       1  1.21473e+001 -1.02536e+001  1.22579e+001 -1.01161e+001  9.05823e-002  2.01938e+000
       2  7.35848e+000 -7.53077e-001  6.47487e+000 -2.00809e+000  8.74220e-002  5.57624e+000
```

- To create this output file using LS-PrePost, you must perform the general steps given earlier and include this:

```
output "absolute or relative dir path\shell_p-strain.k" 102 1 0 1 0 0 0 0 0 0 1 0 0 0 0 1.000000
```

9.8. Nastran Files

SoS follows the MSC.Nastran file format specification given in [11] (p. 179), which may be very similar to most of the other Nastran programs used today. SoS supports only a very limited number of keywords for data import and even fewer keywords for data export.

The following topics explain how Nastran files can be used in SoS:

9.8.1. Importing Nastran Mesh Input Files

Typical file extensions for Nastran mesh input files are `.dat` and `.bdf`, although the SoS Nastran mesh import interface accepts all file extensions and does not distinguish between them. As described in [11] (p. 179), the input file must consist of sections arranged in the following order:

- Nastran Statement
- File Management Statements
- Executive Control Statements
- CEND
- Case Control statements
- BEGIN BULK
- Bulk Data Entries
- ENDDATA

While the first two sections are optional, the subsequent sections are all required, even if they do not contain an entry. The SoS Nastran mesh import interface checks for the existence of all required sections, informing you on the debug log level if any are missing. During this check, all the content until the required `BEGIN BULK` delimiter is skipped. The definition of the finite element mesh must then be given in the required *Bulk Data Entries* section.

Compared to the MSC.Nastran syntax in [11] (p. 179) and[12] (p. 179), only some of its syntax elements is supported. The following list of limitations is not exhaustive:

- More than 80 characters per line are not supported.
- **`System cell 363`** as well as **`STRICTUAI`** are ignored. This, the free-field format is not automatically continued by terminating the parent with a comma. Instead, it is clipped. Again, more than 80 characters per line are not supported.
- Comments may be inserted in any of the parts of the input file. They are identified by a dollar sign **`$`** in any column.
- SoS supports input in sorted order only. Therefore, continuation pointers are not resolved nor even checked at all. The Nastran mesh input interface takes only continuation entries into account, assuming an immediately following line not starting either with a delimiter or a keyword definition is a continuation line.
- The Nastran **`IFPSTAR`** is ignored, so all entries in free-field format are limited to 8 characters. However, as implemented, a keyword followed by a **`*`** character activates the automatic conversion of the free-field format to the large field format. This is not part of the MSC.NASTRAN format specification.
- Replication is not supported.
- The Nastran mesh input interface supports the small, large, and free-field formats. The **`MSGMESH`** format is not supported.
- Integer data types may contain decimal points.
- Real data types may be written in C syntax only. While the following real numbers are legal to the MSC.Nastran format specification, they cause an error in the SoS Nastran mesh import interface
  - **`0.7+1`**
  - **`.70+1`**
  - **`70.-1`**
- While not clearly documented in [12] (p. 179), the Nastran input interface assumes that the first field of any supported format is always left-justified.

Due to the Nastran mesh input interface limitations, each passed Nastran mesh file must contain a non-empty *Bulk Data Entries* section that defines either a valid finite element mesh or supported load bulk data. This definition can be given only with the bulk data entries in the following table. All other bulk data entries are ignored.

Entry	**Short Description**
`CHEXA`	Six-Sided Solid Element
`CPENTA`	Five-Sided Solid Element
`CQUAD`	Fully Nonlinear Surface Strain Element
`CQUAD4`	Quadrilateral Surface Element
`CQUAD8`	Curved Quadrilateral Surface Element

Entry	Short Description
CQUADR	Quadrilateral Surface Element
CQUADX	Fully Nonlinear Axisymmetric Surface Element
CTETRA	Four-Sided Solid Element
CTRIA3	Triangular Surface Element
CTRIA6	Curved Triangular Surface Element
CTRIAR	Triangular Surface Element
CTRIAX	Fully Nonlinear Axisymmetric Surface Element
GRID	Defines the geometry
PLPLANE	Fully Nonlinear Hyperelastic Surface Element
PLSOLID	Fully Nonlinear Hyperelastic Solid Element
PSHELL	Surface Element Property
PSOLID	Properties of Solid Elements
TEMP	Grid Point Temperature

For the entries **PSHELL**, **PSOLID**, and **TEMP**, these bulk data entry comments are applicable:

- The Nastran mesh input interface parses and validates the property bulk data entries but does not fully import them.
- With the exception of the property identifier, properties are not imported into SoS.
- Providing section definition is not obligatory because the Nastran mesh import interface creates SoS-compatible section definitions.

Note:

The SoS installation includes a simple Nastran input file: `nastran.bdf`. For more information, see Input File Examples (p. 109).

9.8.2. Simulating Nastran Mesh Input Files

The SoS Nastran simulate interface can modify valid Nastran input files that:

- Meet the requirements given in Importing Nastran Mesh Input Files (p. 131)
- Are compatible with the project-related reference mesh

Typically, the simulate feature expects to modify one of the SoS project-related valid Nastran mesh input files. The following list, not exhaustive, includes some limitations:

- Only the following fields of the **GRID** bulk data entry can be modified:
  - x coordinate in the basic coordinate system
  - y coordinate in the basic coordinate system

  - z coordinate in the basic coordinate system

- CSV-formatted input files cannot be simulated only. The free-field format is automatically translated into the short or long format respectively. Bulk data formatted in CSV is rewritten. Continuation markers are automatically generated if necessary.

- Real data types can miss the decimal point, although the decimal point is required by the MSC.Nastran format specification,

9.8.3. Exporting Nastran Mesh Input Files

As mentioned in Exporting a Geometry (p. 65), the SoS Nastran export interface cannot write a ready-to-use Nastran input file from start to finish because the SoS kernel does not provide the information needed to do so. The interface exports the mesh definition in the long format type as bulk data file only, meeting the requirements given in Importing Nastran Mesh Input Files (p. 131).

On output, the mesh definition can be represented using these bulk data entries:

Entry	Description
CEND	—
BEGIN BULK	—
CTRIA3	Used for SoS shell element definition of 3n triangles
CTRIA6	Used for SoS shell element definition of 6n triangles
CQUAD	Used for SoS shell element definitions of 4n, 8n, and 9n quadrangles
CTETRA	Used for SoS continuum element definitions of 4n and 10n tetrahedra
CHEXA	Used for SoS continuum element definitions of 8n, 20n, and 27n hexahedra
GRID	The x, y, and z coordinate in the global coordinate system

The export interface tries to keep SoS indices as long as possible, intervening in the following cases only:

- Index spaces starting in SoS at **`0`** are shifted by one to meet Nastran requirements.

- If SoS contains several parts, the exporter interface translates the index space automatically into a continuous one.

9.9. optiSLang Files

SoS supports a limited subset of the optiSLang classic postprocessing file format specification (**`.bin`**) to basically import or modify such binary files.

The following topics explain how optiSLang files can be used in SoS:

9.9.1. Importing optiSLang Data Files

Next to the more recent optiSLang monitoring data base file format specification (`.omdb`), optiSLang still writes its data in the *optiSLang classic prostprocessing* file format specification (`.bin`). The SoS optiSLang import interface supports the `.bin` format specification to import scalar input and output parameters and signals. The following list, not exhaustive, includes some limitations:

- Only scalar input and output parameters of value type **`bool`**, **`integer`**, or **`real`** are imported. **`String`** value types are not supported.
- Active constant input parameters are not imported
- SoS interprets scalar parameters of value **`NaN`** or **`inf`** as missing items.

9.9.2. Exporting optiSLang Data Files

SoS cannot create an optiSLang binary file from start to finish. It can only modify an existing file, basically creating a modified copy of the original file. This is necessary to keep information on the parameter definitions of optiSLang. You can use the exported binary file directly in optiSLang's MOP solver. When modifying the file, all defined **`responses`** are replaced by the parameters to be exported from SoS.

If any data is already imported from optiSLang, SoS tests the equivalence of the already imported file and the optiSLang binary file that serves as a reference for export. This step may be necessary to ensure data consistency of two independent samplings.

If a scalar data object to be exported contains a missing data item, the value is marked as **`NaN`** in the output file. optiSLang's MOP solver then ignores the specific value in the analysis.

9.10. PAM-STAMP Files

Although PAM-STAMP might provide its own input file format, the SoS PAM-STAMP import interface only imports PAM-STAMP output files. Because ANSYS has no access to documentation about the PAM-STAMP data file format, it has used customer-generated output as the reference data file format specification.

A typical file extension for a PAM-STAMP data file might be `.asc`. The file specification is given as follows:

- Lines 1 and 2 are skipped. They may or may not be empty.
- Words in a line are separated by one or more spaces.
- Line 3 is expected to contain at least two words. The first word is skipped. The remaining words in the line are interpreted as the data identifier.
- Lines 4 through 6 are skipped.
- Line 7 is expected to contain two words. The first word is skipped. The second word is either **`NODE`** or **`ELEMENT`**, which instructs the interface to expect either nodal or element data.
- Lines 8 and 9 are skipped.

- The remaining lines are read word-by-word until the end of the file is encountered. Exactly two words per line are expected. The first word must be of the data type **integer** and is interpreted as either a nodal or element identifier. The second word is interpreted as the associated scalar data value. An exception is a line with a zero entry only in the first column, which signals an imminent end of file. If valid data lines follow, they are still imported until the end of file is encountered.

PAM-STAMP Data File Example

```
     114734        bend 8
   Run_name
   Contour       bend
    Unit         NONE
    Dimension    1
    Type         SCALAR
    Entity       ELEMENT
   State         State 4/end : Time = 0.000000
    Number         2
       953005 -5.30040e-004
       953006 2.08850e-004
       0
```

9.11. PERMAS Files

SoS supports the INTES file format specification in [7] (p. 179). However, it can only import a very limited number of keywords of a basic component.

A typical file extension for PERMAS input files is `.dat`. The SoS PERMAS import interface accepts all file extensions and does not distinguish between them. As described in [7] (p. 179), the input file must consist of at least one **COMPONENT** and one **$STRUCTURE** bracket. The SoS PERMAS import interface is limited to one **COMPONENT** definition, which is the basic component, and one structure definition.

Compared to the INTES PERMAS syntax in [7] (p. 179), the SoS PERMAS import interface supports only some of its syntax elements. The following list, not exhaustive, includes some limitations:

- More than 256 characters per physical line are not supported.
- The environment variable **PERMAS_MAXCOL** is ignored. Again, more than 256 characters per line are not supported.
- Comments may be inserted in any of the parts of the input file. They are identified by an exclamation mark **!** in any column.
- Due to the lack of the proper keyword support, data generation is not supported.
- Keywords may be arranged in an arbitrary manner and must not follow any PERMAS restrictions to component configurations. Any bracket may appear before and outside any level definition. All keys are automatically aligned to the sole component definition.
- Only the default Cartesian component system is supported.
- Integer data types may contain decimal points.

- Floating point formats may be written in C-syntax only. While the following floating point numbers are legal to the PERMAS format specification, they cause errors in the SoS PERMAS import interface:
  - **`0.7+1`**
  - **`.70+1`**
  - **`70.-1`**

Due to the PERMAS input interface limitations, each passed PERMAS file must contain a non-empty **`$ENTER COMPONENT`** high-level bracket with a valid finite element mesh definition. This definition can be given with the keys in the following table only. All other bracket definitions are ignored. For example, the key **`$COOR`** can refer to the implicitly defined Cartesian component system only because no other coordinate system can be defined due to the lack of key support.

Key	Description	Limitations
`$ENTER`	High-level bracket definition	Support restricted to **`COMPONENT`** only
`$STRUCTURE`	Bracket for input of mesh and utility	Restricted keyword support
`$COOR`	Defines the geometry	Neither the node set definition set automatically nor the usage of a different coordinate system is supported
`$ELEMENT`	Element connectivity definition	Neither the element set definition set automatically nor the usage of a different element type than **`TRIM6`** is supported
`$NSET`	Defines a node set	Support restricted to **`RULE=ITE`**M only
`$ESET`	Defines an element set	Support restricted to **`RULE=ITEM`** only

Note:

The SoS installation includes a simple PERMAS input file: `permas.dat`. For more information, see Input File Examples (p. 109).

9.12. STL Files

SoS can read STL files in binary and alphanumeric format and write STIL files in binary format. A specification of the file format can be found in reference [14] (p. 179).

The following topics explain how STL files can be used in SoS:

9.12.1. Importing STL Mesh Input Files

Because a STL mesh input file consists only of triangular shell elements and their vertex coordinates, SoS does not import the mesh as it is defined in the STL file. To create a continuous FEM mesh, SoS automatically determines node indices and node coordinates and then associates the STL file's triangular segments with these nodes.

- Due to round-off errors, vertices are assumed to be the same node if the coordinates are similar (not only equal). Two coordinates are similar if their distance is smaller than a certain fraction of the minimum triangle size.
- Due to the merge of two or more vertices, some triangles may degenerate and be eliminated. As a result, the number of nodes (and also number of triangles) may be different from the original STL file.

SoS can export deformed geometries directly to binary STL format.

9.12.2. Exporting STL Mesh Input Files

Because the STL file format consists only of triangular shell elements and their vertex coordinate, SoS is likely not able to export the original SoS reference mesh. To create this format, SoS determines internally the boundary faces and triangulates them. It then exports the resulting tessellation to the binary STL file.

9.13. VTK Files

SoS supports the simple legacy VTK file format specification defined in [8] (p. 138). Support is limited to the **`UNSTRUCTURED_GRID`** dataset type only.

A typical file extension for simple legacy VTK input files is `.vtk`. The SoS simple legacy VTK import interface accepts all file extensions and does not distinguish between them. As described in [8], the input file consists of five basic parts in the following order:

1. File version and identifier
2. Header that can be used to describe the data and to include any other pertinent information
3. File format
4. Dataset structure
5. Dataset attributes

The definition of the finite element mesh must be given in the last two parts.

Compared to the simple legacy VTK Syntax in [8] (p. 138), the SoS simple legacy VTK import interface supports only some of its syntax elements. The following list, not exhaustive, includes some limitations:

- All parts are required
- Only the ASCII file format is supported.

Due to the VTK input interface limitations, each passed simple legacy VTK input file must contain a non-empty model data definition. This definition can be given only with the keywords and their parameters in the following table. All other keywords are ignored.

Keyword	Description	Supported Parameters	Comments
ASCII	Defines the file format type	—	The format type **BINARY** is not supported and leads to an exception.
DATASET	Starts the description of the geometry and topology	**UNSTRUCTURED_GRID**	—
POINTS	Defines the geometry	**n**	—
CELLS	Defines an element directly by specifying its nodes	**n**	—
CELL_TYPES	Defines the element type	**n**	**TYPE** support is limited to the linear cell types **VTK_TRIANGLE**, **VTK_PIXEL**, **VTK_QUAD**, **VTK_TETRA**, **VTK_VOXEL**, and **VTK_HEXAEDRON**. Other **TYPE** values are ignored.

Note:

The SoS installation includes a simple VTK input file: `vtk.vtk`. For more information, see Input File Examples (p. 109).

Chapter 10: SoS Scripting

These topics provide SoS scripting information:

10.1. SoS-Embedded Scripting Using Lua

SoS provides embedded scripting using the Lua script language. Nearly any command, including visualization, triggers a script execution. Hence, you can store the log of a session in a Lua script. By executing a previously stored script, you can:

- Execute a session to change input data
- Change certain session parameters
- Debug the application in case of an unforeseen event (crash)

Theoretically, you can even use the script engine to enhance SoS functionality.

Features of Interest:

- SoS makes the Lua language (version 5.3) fully available.
- You can execute individual commands using the Lua console at the bottom of the main window.
- You can execute multiple commands by copying and pasting them. You can insert a block of commands from the clipboard or press **Shift** + **Enter** in the Lua console.
- You can execute Lua scripts from the command line using the command line parameter **`-s`**.
- SoS enhances the Lua language by adding methods to the Lua table `sos`.
- Compatibility of the SoS script API cannot be guaranteed among different versions of SoS (as opposed to the binary data base format).
- An especially helpful Lua command is `info(object)`. You can use this command to list information on a given variable, table, or meta table (including methods and member variables).

10.2. SoS Python Package for optiSLang

SoS provides a Python package offering the full SoS script API compatible with optiSLang Python, meaning the Python executable delivered with optiSLang. The optiSLang Python executable is called when using the Python node in optiSLang, allowing you to execute SoS script code directly in optiSLang. You use the SoS Python package for seamless development of optiSLang custom integration nodes.

10.2.1. Installing the SoS Python Package

The SoS Python package is compatible with optiSLang Python only. To install the SoS Python package from a Windows command prompt, in your optiSLang installation path, use the command **`optislang-python.cmd`**:

```
%ANSYS211_DIR%\optiSLang\optislang-python-cmd -m pip install --user %SoS_install_dir%\py_sos_package--21.1.0.tar
```

10.2.2. Licensing the SoS Python Package

Importing the SoS Python module requires an SoS license. Because the Python module looks for license files (`*.lic`) in the folder defined by the environment variable **`DYNARDO_LICENSE_FILE`**, set **`DYNARDO_LICENSE_FILE`** to the folder containing your SoS license file.

The license remains locked for the lifetime of the Python program. There is no method for releasing it earlier.

10.2.3. Using the SoS Python Package

Here are some examples for using the SoS Python package:

Note:

Always treat quotation marks carefully.

Load modules "sos" and "tmath"

```
try:
    import os
    os.environ["DYNARDO_LICENSE_FILE"] = "C:/absolute/path" # path to folder containing your license file
    from sos_package import sos, tmath
except:
    print("ERROR: failed to load SoS module. Use 'optislang-python.cmd -m pip install --user py_sos_package-8.0.
```

Run SoS Lua code

From Python, you can call any SoS Lua code that is generated by the SoS GUI and written to the command log:

```
sos.execLua( 'settings = sos.LoadDataBaseSettings("myDatabase.sdb"); sos.loadDataBase(sos.database(), settings)'
sos.printMeshInfo(sos.database()) # print mesh information of the loaded database using Python code
```

Lua versus Python SoS script

To call class member methods, Lua uses **`":"`** while Python uses **`"."`**.

Lua code:

```
-- select all field data objects of quantity "pstrain"
pstrainDataObjects = sos.ElementDataObjectFilter(sos.database():elementData()):filterByQuantityIdent("pstrain")
```

Python code:

```
# select all field data objects of quantity "pstrain"
pstrainDataObjects = sos.ElementDataObjectFilter(sos.database().elementData()).filterByQuantityIdent("pstrain")
```

Python lists and sos.StringVector

Input arguments of type **`sos.StringVector`** are compatible with Python lists of strings.

```
# select all field data objects of quantities "pstrain" and "thickness"
dataObjects = sos.ElementDataObjectFilter(sos.database().elementData()).filterByQuantityIdent(["pstrain", "thick
```

Tmath Module and numpy

The module **`tmath`** is a fast linear algebra library. It is based on the C++ library Eigen. The following examples illustrate the compatibility with Python lists and numpy arrays.

```
# vector in numpy
vec_np = numpy.array([1,2,3])

# vector in tmath
vec_tmath = tmath.Matrix([1,2,3]) # tmath.Matrix() constructor accepts lists and tuples

# matrix in numpy
mat_np = numpy.array([[1,2,3],[4,5,6]])

# matrix in tmath
mat_tmath = tmath.Matrix([[1,2,3],[4,5,6]]) # tmath.Matrix() constructor accepts lists of lists

# numpy.array to tmath.Matrix
mat_tmath = tmath.Matrix(mat_np.tolist())

# tmath.Matrix to numpy.array
mat_np = numpy.array(mat_tmath)

# functions with tmath.Matrix input arguments
dataObject = sos.createElementDataObject(sos.database(), vec_tmath)
dataObject = sos.createElementDataObject(sos.database(), vec_np.tolist())
dataObject = sos.createElementDataObject(sos.database(), [1,2,3])
```

Chapter 11: SoS Integrations

The following topics describe SoS integrations:

11.1. optiSLang Integration Nodes

optiSLang is basically a process control software offering a variety of sensitivity, optimization, and robustness algorithms and a rich interface for data input and output handling. optiSLang includes *custom integration nodes* that incorporate SoS into optiSLang's flow management. As these custom integration nodes are subject to intensive development, this documentation applies only to optiSLang version 8.1 (2021 R1).

These topics describe custom integration nodes:

Note:

- By default, optiSLang sets the working directory of custom integration nodes directly to the `*.opd` path. As a consequence, using two custom integration nodes within the same system level is not recommended as the latter one overwrites the data of the first one. While the custom integration node **`SignalMOP_SoS`** automatically enforces a distinct working directory, you should choose the **Distinct working dir** option in the **Additional options** dialog box of the other custom integration nodes. Another simple workaround is to place the custom integration node into a simple **`System`** node, which creates its own working directory.

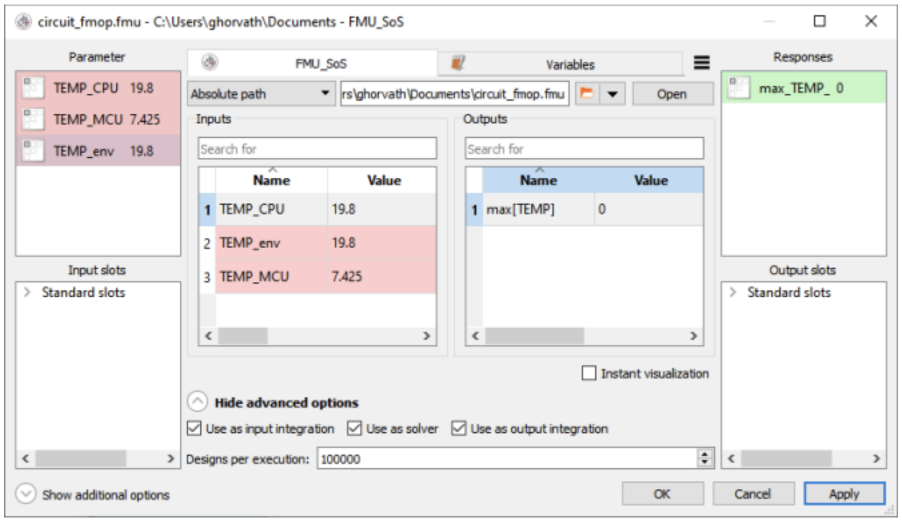
- You can find a custom integration node either by searching for the module name or at **Process chain elements** → **Add-Ins**.

11.1.1. FMU_SoS Integration Node

The **FMU_SoS** integration node in optiSLang consumes FMUs exported with SoS. For more information, see:

- Exporting a Macro-Based FMU 2.0 Archive File (p. 73)
- Exporting a Field-MOP as an FMU in the *Statistics on Structures Tutorials*

The Model Exchange FMU 2.0 file that SoS exports returns scalar outputs for discrete sets of scalar input parameters. You can use the **FMU_SoS** integration node to open any FMU file that SoS exported and embed it in an optiSLang workflow. The integration node detects all scalar input and output parameters exposed by the SoS FMU file.


circuit_fmop.fmu - C:\Users\ghorvath\Documents - FMU_SoS

Parameter

TEMP_CPU	19.8
TEMP_MCU	7.425
TEMP_env	19.8

Absolute path: rs\ghorvath\Documents\circuit_fmop.fmu

Inputs

	Name	Value
1	TEMP_CPU	19.8
2	TEMP_env	19.8
3	TEMP_MCU	7.425

Outputs

	Name	Value
1	max[TEMP]	0

Responses

max_TEMP_	0

☐ Instant visualization

Hide advanced options

☑ Use as input integration ☑ Use as solver ☑ Use as output integration

Designs per execution: 100000

FMU_SoS Integration Node Setup

- The **FMU_SoS** integration node must process all designs at once. In the advanced options, set **Designs per execution** to a value large enough.
- The FMU file's input and output values are written to the database file `FMU_SoS_out.sdb` in the working directory. One design identifier is appended to the database for each set of input parameters.
- The file `FMU.log` that is written to the working directory contains logging output of the FMPy Python module used to solve the FMU. You must install this Python module in optiSLang's Python environment.

FMPy Python Module Installation on Windows

To install the FMPy Python module on Windows:

1. Open a command prompt window. If optiSLang is installed in `C:\Program Files\`, you must open the command prompt window with administrator privileges.
2. Install the module:

- If you have access to the official remote repository, execute this command:

```
"%AWP_ROOT211%\optiSLang\optislang-python.cmd" -m pip install fmpy=0.2.24
```

- If you do not have access to the official remote repository, execute this command:

```
"%AWP_ROOT211%\optiSLang\optislang-python.cmd" -m pip install --no-index --find-links --no-deps "%SoS_in
```

The ANSYS installer for 2021 R1 sets the environment variable **AWP_ROOT211**. If necessary replace **AWP_ROOT211** with the path to your optiSLang installation.

11.1.2. Generate_SoS Integration Node

The **Generate_SoS** integration node generates new designs based on random field amplitudes or scalar input parameters of a field-MOP analysis:

Input

The randomfield_amplitudes.csv file or the *.sim output file of a random field or a field-MOP analysis. For more information, see Generating Field Objects in optiSLang for Use in a Simulation (p. 74), Generating Amplitudes from Fields in optiSLang for Use in a Simulation (p. 75), and the subsequent topics about the other integration nodes.

Output

A new field design in the current design directory of the optiSLang workflow. For more information, see Generating Field Objects in optiSLang for Use in a Simulation (p. 74)

11.1.3. AmplitudesFromField_SoS Integration Node

The **AmplitudesFromField_SoS** integration node automatically translate an input field into scalar parameters.

The transition is performed using random fields. The field data has to be written to a file within the current Design directory. SoS must be prepared to read this file and transform the contained fields into scalar parameters using a random field model. For more information, see Generating Amplitudes from Fields in optiSLang for Use in a Simulation (p. 75). The scalar coefficients are exposed to optiSLang in terms of responses or output slots.

11.1.4. SignalMOP Integration Node

The **SignalMOP** integration node is part of the MOP node. It simplifies 1D or signal field analysis. Node inputs and outputs can be summarized as follows:

Input

An optiSLang BIN or OMBD file. An OMBD file is recommended. It is automatically converted to the old BIN file format that is the only optiSLang input file format that SoS supports.

Output

SoS reads in all signals and all scalar input parameters from the input file and builds for each signal both an empirical random field model and a field-MOP. For each signal, SoS then writes out the following file structure as output into the current distinct working directory:

- `sig-name-dir` serves as the root directory for each signal. The directory name matches the unique signal name, which is unique within the optiSLang input file.
- `sig-name-dir/fmop_solver.sim` serves as the input simulation archive for further simulations based on the field-MOP of this specific signal. For more information, see Generating Field Objects in optiSLang for Use in a Simulation (p. 74).
- `sig-name-dir/rf_generate.sim` serve as the input simulation archive for further simulations based on the random field amplitudes of this specific signal. For more information, see Generating Amplitudes from Fields in optiSLang for Use in a Simulation (p. 75).
- `sig-name-dir/sig-name_channel-ID` serves as the root directory for all channel outputs. For each channel of the current signal, it contains a number of CSV files as analysis output, including:
  - `cop.*` defines the graph of the F-CoP values, including the single F-CoP values of the input parameters and the full model F-CoP value.
  - `mean.*` defines the graph of the mean value and the 5%-quartile and 95%-quantile of the current signal.
  - `rsigma.*` defines the graph of the F-CoP values in absolute numbers. For more information, see Creating Field-MOPs (p. 96).
  - `shapes.*` defines the graph of the random field model shapes.
- `sig-name-dir/sig-name.sdb` serves as the SoS database file used for the analysis. This database file can be used later for further checks, visualization, or just to continue working within SoS.
- `sig-name-dir/fmop-debug.omdb` contains the MOP-analysis output of the random field amplitudes for debugging purposes only.
- `signal_mop_solver.sim` serves as the input simulation archive for further simulations based on the field-MOP of all signals contained in the input file.
- `random_signal_generate.sim` serves as the input simulation archive for further simulations based on the random field amplitudes of all signals contained in the input file.
- `*.omdb` is the input file that the custom integration node automatically copies into its distinct working directory. If this input file is an optiSLang monitoring database, the integration node appends analysis output plots automatically for each channel of each signal. The following plots can then be visualized in a postprocessing node:
  - `F-CoP (%)` plots the F-CoP values, including the single F-CoP values of the input parameters and the full model F-CoP value

  - `F-CoP (abs)` plots the F-CoP values in absolute numbers. For more information, see Creating Field-MOPs (p. 96).
  - `Scatter shapes` plots random field model shapes.
  - `Signal statistics` plots the mean value and the 5%-quantile and 95% quantile of the current signal.

11.1.5. Convert_OMDB_to_BIN_SoS Integration Node

The **`Convert_OMBD_to_BIN_SoS`** integration node converts the OMDB file format of optiSLang to the legacy optiSLang BIN format that SoS uses for data exchange of scalar parameters and responses.

11.1.6. Viewer_SoS Integration Node

The **`Viewer_SoS`** integration node opens an SDB file in the SoS Viewer. The SoS Viewer is a read-only application that is capable of showing and visualizing all 2D and 3D data in an SoS database file.

11.1.7. Field_MOP_ANSYSMECH_SoS Integration Node

The **`Field_MOP_ANSYSMECH_SoS`** integration node postprocesses a DOE (Design of Experiments) in an ANSYS Workbench project using ANSYS Mechanical. The ANSYS Workbench project must be configured to create output for SoS using the SoS plugin for Mechanical. The integration node collects the OMDB file and all data from the design directories to create a field-MOP.

11.1.8. Field_MOP_2D_Nested_DOE_SoS Integration Node

The **`Field_MOP_2D_Nested_DOE_SoS`** integration node postprocesses a DOE (Design of Experiments), creating 2D performance maps. The idea behind this integration node is that the 2D grid is based on an encapsulated system, where the outer DOE varies the outer parameters and the inner DOE varies the two operational point parameters. All information is obtained by analyzing the OMDB files from the outer and inner systems.

11.1.9. Field_MOP_2D_Grid_SoS Integration Node

The **`Field_MOP_2D_Grid_SoS`** integration node postprocesses a DOE (Design of Experiments), creating 2D performance maps. The idea behind this node is that the 2D grid is created by some process and saved to disk as a CSV file. The DOE then varied outer parameters that lead to changed values of the matrix-like performance map or image.

11.2. SoS Extension for ANSYS Mechanical

The SoS extension for ANSYS Workbench is an ACT-created WBEX file that is referred to herein as an ACT plugin. This plugin helps to interact with ANSYS Mechanical, allowing you to:

- Import random field models for geometric imperfections from SoS. For more information, see Importing External Models (p. 151).

- Generate geometric random imperfections using random fields. For more information, see Generating Geometric Imperfections (p. 152).
- Generate geometric free-form variations. For more information, see Generating Geometric Free-form Variations (p. 155).
- Export results to SoS for a field-MOP or statistical analysis. For more information, see Exporting Results to SoS (p. 158).

The principal usage of this ACT plugin is explained in optiSLang Integration Tutorials in *Statistics on Structures Tutorials*.

ACT Plugin Installation

The ACT plugin is automatically placed at the correct hard disk location by the SoS setup for your respective Workbench versions. Alternatively, you can install it manually in Workbench after the installation of SoS. A copy of the ACT plugin (WBEX file) is in the SoS installation directory.

ACT Plugin Usage Recommendations

Mesh morphing, which includes generating geometric random imperfections (p. 152) and generating geometric free-form variations (p. 155), can lead to negative volumes or Jacobians for some elements. In such cases, the model cannot be processed by the ANSYS kernel. SoS provides a number of options that help to stabilize the morphing procedure, but they increase the computational effort. In general, you should work through the options given in the following table from top to bottom:

Desired Stability	**Computation Time**	**Setting Options**		
		Use mesh stabilization	**Linearize morphing for quadratic elements**	**Auto-delete bad elements**
Low	Low	False	False	False
-	High	True	False	False
-	High	True	True	False
High	High	True	True	True

Note:

The **Auto-delete bad elements** option should be used with great care. Most importantly, it should be checked whether only a few scattered elements are deleted or large clusters of elements are deleted. The latter case indicates a more general problem with the mesh. If only a few scattered elements are deleted, check whether these elements are located in important regions.

The mesh morphing algorithm works differently for different element types. If possible, using quadratic tetrahedrons is encouraged.

Desired Stability	**Element Type**
High	Linear tetrahedrons

Desired Stability	Element Type
—	Quadratic tetrahedrons
—	Linear hexahedrons
Low	Quadratic hexahedrons

The following topics describe scenarios for using the ACT plug-in:

11.2.1. Importing External Models

If you prepared a random field model in SoS, you can use it in ANSYS Mechanical to generate geometric imperfections. This can be based, for example, on measurements that are to be imported into SoS. In SoS, you prepare a random field simulation and export the geometric changes to Mechanical APDL format.

When selecting **Import** → **Import external model** from the extension's toolbar, you must specify the file `randomfield_amplitudes.csv` from the simulation directory. The directory is then imported into the ANSYS Workbench project by a copy and the new node appears in the Mechanical tree with the name **Generate variations (SoS)**.

Generate variations (SoS)

- **Solver options**

  – **Number of CPUs used by SoS**

    If you plan to use Mechanical in a DOE, you should limit the number of CPUs used because SoS typically takes 100% CPU load and reduces the efficiency of parallel tasks. If you want to get full CPU power, for example, enter **`0`** to get all CPU cores.

  – **Auto-delete bad elements**

    If you change this to **`Yes`**, all elements with too large distortions or negative Jacobians are erased from the mesh before the actual analysis is done. This option is helpful for complex meshes if only a small number of elements is failing and responsible for a failed design. If this option is selected, you can check the log file `solve.out` to learn how many elements were deleted.

- **Visualization**

  Enable visualization

  After creating the model, you can visualize the individual scatter shapes associated with the random field parameters as a colored contour plot. Use this parameter to turn visualization off (default) or on. For a larger model, you should turn visualization off because visualizing a large model can be slow.

- **Input parameters**
  - This lists all available random field parameters that control the outcome of the geometric imperfection.
  - You can specify the value of each parameter (or define it to be a Workbench parameter for exposure to optiSLang -> check it)
  - Also the internal name of the parameter in SoS is shown. This identifier may be different from the identifier being used in Workbench or later in optiSLang. It is listed here to understand the log files in SoS.
- **SoS simulation data location**
  - **Source item (internal)**

    If the Generate-node is connected with another node, such as a synthetic random field model or a free-form variation model node, this option displays the source item in terms of a local directory path. This path is a local directory in the `SYS/MECH` path of the Workbench project directory.
  - **Simulation data path item (internal)**

    This path is a local directory in the `SYS/MECH` path of the Workbench project directory. It contains all data that SoS requires to generate imperfections. You may also find the log files in there (in the `sos_log` subdirectory), which is a good starting point in case of errors. Also, you can open the `sos_data.sdb` file in that directory in the standalone SoS GUI for detailed analysis of the random field model

Caution:

You must use the same mesh in Mechanical as in SoS. Never clear or reset the mesh.

11.2.2. Generating Geometric Imperfections

Geometric imperfections are generated using synthetic random field models. Selecting the option for creating geometric perfections creates two nodes:

- Synthetic random field (SoS) (p. 152)
- Generate variations (SoS) (p. 155)

Synthetic random field (SoS)

Boundary to be parameterized

The part of the boundary for which to find a random field parametrization. It can be a named selection of type face, element face, or node.

Fixed boundary

The part of the boundary that is not to be modified (fixed). It can be a named selection of type face, element face, or node. All boundary parts that are not fixed or parametrized are "free" in the sense that SoS can modify them to interpolate between directly changed boundaries and fixed boundaries. It is recommended that the fixed boundary does not touch the directly parametrized boundary.

Mesh part to be exported to SoS

The substructure that is to be exported to SoS. The boundary to be parametrized or fixed should belong to this substructure. The smaller the exported substructure, the smaller are the computational resources required by SoS. The scoping can be a named selection of type body or element.

Definition

- **Desired variability (%)**

  Target value for the accuracy of the random field model in percent of the defined statistical variance.

- **Maximum number of parameters**

  An upper bound for the parameters to be used. The actual number of parameters may be less if the desired variability is to be obtained with a smaller number of shapes. If a greater number is required, the series is truncated here.

- **Correlation length**

  The correlation length parameter for the squared-exponential autocorrelation model. The larger the value, the larger the wave length. The smaller the value, the shorter thee wave length. White noise is a correlation length of zero. The latter case must have more shape parameters.

- **Standard deviation of geometric variation**

  A constant value for the statistical standard deviation.

- **Mean geometric variation**

  A constant value for the statistical mean value. This is basically a constant shift around which the variation is applied in both directions.

Visualization

- **Enable visualization**

  After creating the model, you can visualize the individual scatter shapes associated with the random field parameters as a colored contour plot. Use this parameter to turn visualization off (default) or on. For a larger model, you should turn visualization off because visualizing a large model can be slow.

- **Visible variation shape index**

The number for the index of the shape. A special value of `0` displays the mean, which should be a constant.

Advanced options

- **Use mesh stabilization**

  When set to **False**, the computed geometric deviation is applied directly to the nodes. However, there is no check if the result is feasible. Thus, the result is fast, but it is very likely not solvable by ANSYS Mechanical.

  When set to **True**, an iterative algorithm tries to relax element distortions that are too strong. This avoids situations with negative volumes or negative Jacobians. The larger the applied deformations and the greater the number of nodes or elements, the more computation time that is needed.

- **Linearize Morphing for quadratic elements**

  This option affects only quadratic elements.

  When set to **True**, for quadratic elements, SoS places the mid-edge nodes using a linear interpolation of the finite element geometry between the corner vertices. During morphing, some elements can be greatly distorted, leading to negative volumes or negative Jacobians. This option can help to avoid these problems and tends to produce more stable results.

  When set to **False**, quadratic elements are morphed with quadratic curvature.

- **Erase degenerated shapes**

  When set to **True**, shapes with short wavelengths or short correlation lengths (that is with a very localized effect) are deleted. When set to **False**, all shapes are retained.

- **Test on mesh distortion**

  When set to **True**, SoS throws an error before the mesh is transferred to the ANSYS Mechanical kernel. Because SoS uses a different FEM formulation, however, the test in SoS may be stricter than in Mechanical. You should use this option if the preparation in the Mechanical kernel takes a lot of time and if you do not want to start it when creating non-meaningful geometric variations.

- **Move nodes along**

  When set to **Boundary normal**, the variations are applied along the direction of the normal vector assigned with each FEM node on the boundary.

  When set to **Three coordinate axis**, each variation pattern can be applied along the x, y, and z axis. This results in three times as many parameters, one parameter for each direction.

  The variation along the normal direction is the default. The other option is more stable in the context of complex geometries.

Solver options

- **Number of CPUs used by SoS**

The number of CPU cores that SoS can use during the creation of the random field model. The default of `0` means that SoS is to use all cores.

- **Internal directory**

  This path is a local directory in the `SYS/MECH` path of the Workbench project directory. It contains all data that SoS requires to generate the random field model. You may also find the log files in there (in the `sos_log` subdirectory), which is a good starting point in case of errors. Also, you can open the `sos_data.sdb` file in that directory in the standalone SoS GUI for detailed analysis of the random field model.

Generate variations (SoS)

You must update the **Synthetic random field** model node before options are visible for defining or changing the second node.

See the previous option descriptions for Synthetic random field (SoS) (p. 152).

Caution:

Never clear or reset the mesh. You may clear the generated data of the **Generate variations** node, but a reset of the **Synthetic random field** node also clears the random field model. Further, ANSYS Mechanical cannot proceed with the **Generate variations** node automatically if the **Synthetic random field** mode is updated.

11.2.3. Generating Geometric Free-form Variations

Geometric variations can be created using a free-form variation model. In contrast to a synthetic random field model, the variation patterns are tent-like shapes with a single maximum (> 0) going asymptotically to zero with growing distance to its support point. Creating the option for generating geometric free-from variations creates two nodes:

- Free-form variation (SoS) (p. 155)
- Generate variations (SoS) (p. 158)

Free-form variation (SoS)

Boundary to be parameterized

The part of the boundary for which to find a random field parametrization. It can be a named selection of type face, element face, or node.

Fixed boundary

The part of the boundary that is not to be modified (fixed). It can be a named selection of type face, element face, or node. All boundary parts that are not fixed or parametrized are "free" in the sense that SoS can modify them to interpolate between directly changed boundaries and fixed boundaries. It is recommended that the fixed boundary does not touch the directly parametrized boundary.

Mesh part to be exported to SoS

The substructure that is to be exported to SoS. The boundary to be parametrized or fixed should belong to this substructure. The smaller the exported substructure, the smaller are the computational resources required by SoS. The scoping can be a named selection of type body or element.

Definition

- **Support points**

  The mode for choosing the support points for each shape.

  When set to **Auto**, the support node identifiers are automatically generated. Each shape is associated with a single FEM node. The location of the nodes are chosen to distribute them homogeneously on the boundary with respect to the distance between the individual supports.

  When set to **Manual + Auto**, manually defined support points can be added to the automatically generated support points.

- **Scoping**

  This option is visible only when **Support points** is set to **Manual + Auto**. It defines the (named) selection of the manual supports. FEM nodes, vertices, and edges are allowed. Neighbouring FEM nodes are collected into a single support region, which allows you to influence the shape of the associated variation pattern.

- **Number of auto-parameters**

  The number of automatically generated support points.

- **Magnitude of free variation**

  The number for the index of the shape.

Visualization

- **Enable visualization**

  After creating the model, you can visualize the individual scatter shapes associated with the random field parameters as a colored contour plot. Use this parameter to turn visualization off (default) or on. For a larger model, you should turn visualization off because visualizing a large model can be slow.

- **Visible variation shape index**

  The number for the index of the shape.

Advanced options

- **Constant variation**

  The default setting is **Do not use**. If you set this to **Use extra parameter**, an additional parameter is created that is associated with a constant variation for all FEM nodes.

- **Magnitude of constant variation**

  This option is visible only when **Constant variation** is set to **Use extra parameter**. It defines the magnitude of variation being associated with a parameter of 1.

- **Use mesh stabilization**

  When set to **False**, the computed geometric deviation is applied directly to the nodes. However, there is no check if the result is feasible. Thus, the result is fast, but it is very likely not solvable by ANSYS Mechanical.

  When set to **True**, an iterative algorithm tries to relax element distortions that are too strong. This avoids situations with negative volumes or negative Jacobians. The larger the applied deformations and the greater the number of nodes or elements, the more computation time that is needed.

- **Linearize Morphing for quadratic elements**

  This option affects only quadratic elements.

  When set to **True**, for quadratic elements, SoS places the mid-edge nodes using a linear interpolation of the finite element geometry between the corner vertices. During morphing, some elements can be greatly distorted, leading to negative volumes or negative Jacobians. This option can help to avoid these problems and tends to produce more stable results.

  When set to **False**, quadratic elements are morphed with quadratic curvature.

- **Erase degenerated shapes**

  When set to **True**, shapes with short wavelengths or short correlation lengths (that is with a very localized effect) are deleted. Wehn set to **False**, all shapes are retained.

- **Test on mesh distortion**

  When set to **True**, SoS throws an error before the mesh is transferred to the ANSYS Mechanical kernel. Because SoS uses a different FEM formulation, however, the test in SoS may be stricter than in Mechanical. You should use this option if the preparation in the Mechanical kernel takes a lot of time and if you do not want to start it when creating non-meaningful geometric variations.

- **Move nodes along**

  When set to **Boundary normal**, the variations are applied along the direction of the normal vector assigned with each FEM node on the boundary.

  When set to **Three coordinate axis**, each variation pattern can be applied along the x,y, and z axis. This results in three times as many parameters, one parameter for each direction.

  The variation along the normal direction is the default. The other option is more stable in the context of complex geometries.

Solver options

- **Number of CPUs used by SoS**

The number of CPU cores that SoS can use during the creation of the random field model. The default of `0` means that SoS is to use all cores.

- **Internal directory**

  This path is a local directory in the `SYS/MECH` path of the Workbench project directory. It contains all data that SoS requires to generate the random field model. You may also find the log files in there (in the `sos_log` subdirectory), which is a good starting point in case of errors. Also, you can open the `sos_data.sdb` file in that directory in the standalone SoS GUI for detailed analysis of the random field model.

Generate variations (SoS)

You must update the **Synthetic random field** model node before options are visible for defining or changing the second node.

See the previous option descriptions for Generate variations (SoS) (p. 158).

Caution:

Never clear or reset the mesh. You may clear the generated data of the **Generate variations** node, but a reset of the **Synthetic random field** node also clears the random field model. Further, ANSYS Mechanical cannot proceed with the **Generate variations** node automatically if the **Synthetic random field** mode is updated.

11.2.4. Exporting Results to SoS

Using the ACT plugin, you can export meshes and results from ANSYS Mechanical into SoS through specific text file formats that SoS can understand. Tthese files are located in a subdirectory in the `SYS/MECH` folder within the ANSYS Workbench project directory. Therein, you can find the mesh file `sos_mesh.cdb`, which is in the ANSYS CDB format, and the data file `sos_results.k` in the LS-PrePost format.

The following export options are defined. Each of them creates a single node in the solution section of ANSYS Mechanical and are associated with an individual subdirectory.

Caution:

The **General result** can only be configured if the solution is already available. That means that you must suppress it before solving the system for the first time and then unsuppress it and configure the node.

Mechanical results

Data saved for mesh part

The part of the structure (faces, bodies, nodes, or elements) for which the data is to be exported to SoS. It must be a subset of the actual mesh part to be exported (or must be equal to that).

Mesh part to be exported to SoS

The mesh to export to SoS (bodies or elements). It may be a substructure of the entire mesh, depending on the region of interest that you want to analyze with SoS. The exported mesh contains named selections for finite element nodes and elements that contain the selected region in **Data saved for mesh part**.

General options

- **Element types of mesh**

  Because a statistical analysis of the result data does not require the full resolution, you should generally export linearized finite elements to SoS. No mid-edge nodes are exported.

  When set to **Linear**, only linearized finite elements are exported. When set to **Original**, all nodes, including the mid-edge nodes of quadratic elements, are exported.

- **Load step**

  The number of the load step for which the solution is to be exported. You can either define the explicit number of the load step or enter **0** to automatically export the data for the last load step.

Data to export

- **Displacements**

  Exports the displacements (**UX**, **UY**, **UZ**, and **USUM**) as node values.

- **Stress tensor**

  Exports the stress tensor (**SX**, **SY**, **SZ**, **SXY**, **SXZ**, and **SYZ**) as node values (elemental averages for each node).

- **Stress (principal)**

  Exports the principal stresses (**S1**, **S2**, and **S3**) as node values (elemental averages for each node).

- **Stress (von Mises)**

  Exports the equivalent von Mises stress (**SEQV**) as node values (elemental averages for each node).

Solver options

Internal directory

This path is a local directory in the `SYS/MECH` path of the Workbench project directory. It contains all data that is being exported from Mechanical (mesh and data).

Thermal results

Data saved for mesh part

The part of the structure (faces, bodies, nodes, and elements) for which to export data to SoS. It must be a subset of the actual mesh part to be exported (or it must be equal to that).

Mesh part to be exported to SoS

The mesh to export to SoS (bodies or elements). It may be a substructure of the entire mesh, depending on the region of interest that you want to analyze with SoS. The exported mesh contains named selections for finite element nodes and elements that contain the selected region in **Data saved for mesh part**.

General options

- **Element types of mesh**

  Because a statistical analysis of the result data does not require the full resolution, you should generally export linearized finite elements to SoS. No mid-edge nodes are exported.

  When set to **Linear**, only linearized finite elements are exported. When set to **Original**, all nodes, including the mid-edge nodes of quadratic elements, are exported.

- **Load step**

  The number of the load step for which the solution is to be exported. You can either define the explicit number of the load step or enter **`0`** to automatically export the data for last load step.

Data to export

Temperature

Exports the temperature **`TEMP`** as node values.

Solver options

Internal directory

This path is a local directory in the `SYS/MECH` path of the Workbench project directory. It contains all data that is being exported from Mechanical (mesh and data).

General Result

Data saved for mesh part

The part of the structure (faces, bodies, nodes, and elements) for which to export data to SoS. It must be a subset of the actual mesh part to be exported (or it must be equal to that).

Mesh part to be exported to SoS

The mesh to export to SoS (bodies or elements). It may be a substructure of the entire mesh, depending on the region of interest that you want to analyze with SoS. The exported mesh contains named selections for finite element nodes and elements that contain the selected region in **Data saved for mesh part**.

General options

- **Element types of mesh**

  Because a statistical analysis of the result data does not require the full resolution, you should generally export linearized finite elements to SoS. No mid-edge nodes are exported.

  When set to **Linear**, only linearized finite elements are exported. When set to **Original**, all nodes, including the mid-edge nodes of quadratic elements, are exported.

- **Load step**

  The number of the load step for which the solution is to be exported. You can either define the explicit number of the load step or enter `0` to automatically export the data for last load step.

Data to export

You can specify any solution object that is available in the ANSYS RST or RTH solution file. The availability of the individual results depends on the type of analysis (Thermal, Electrical, Mechanical, and so on.). Only a single result is exported for each Export-node.

- **Kind of data**

  How the data is stored in ANSYS.

  When set to **Node**, the data to export is stored at the nodes (such as displacements and temperatures).

  When set to **Element node**, the data to export is stored at the nodes, but with different values for each element. These values are averaged and then exported as node values to SoS (such as stresses). When set to **Element**, the data to export is stored at the elements (such as element quality and elemental volume).

- **Result**

  Select a vector-values result from the list. For example, you might select **U** for the displacement vector.

- **Component**

  Select the component of the vector-values result from the list. For example, you might select **X** for the **X** component of the displacement vector **U**, leading to a result named **UX** in SoS.

Solver options

Internal directory

This path is a local directory in the `SYS/MECH` path of the Workbench project directory. It contains all data that is being exported from Mechanical (mesh and data).

11.3. FMOPSolver Dynamic Linked Library

Selected SoS features can be exported to a dynamic linked library with the C Application Programming Interface (API).

The C API of the **`FMOPSolver`** dynamic linked library allows you only to query and evaluate an already created field-MOP. Hence, reference mesh definition, data input, and field-MOP creation all must be done within the SoS main executable. Once created, the SoS database can be exported and used within this dynamic linked library.

Installation Directories

On Linux, the **`FMOPSolver`** integration gets unpacked automatically during installation. The Windows installation is optional and must be selected explicitly during setup.

If successfully installed, you can find the following `FMOPSolver` subdirectories within the SoS installation directory:

FMOPSolver/share/

Demonstration examples in C, C++, python, and MATLAB languages as well as developer documentation in HTML and PDF format. To read the documentation in HTML format, open `share/doc/html/index.html`.

FMOPSolver/lib/

The dynamic link library itself, without any dependency.

FMOPSolver/include/

The C/C++-API header file.

Linkage to Other Shared Libraries and Compatibility Notes

The installation subdirectory `share` contains some demonstration examples. These examples gets compiled and tested in the SoS build and test environments on a regular basis, using tools of the GNU Compiler Collection (GCC) for programming languages and python 2.7.x for the scripting example.

To use the C and C++ examples in a custom environment, compile the respective source code unit and link to the **`FMOPSolver`** dynamic linked library. Because the **`FMOPSolver`** library depends on several other libraries, you must further ensure that the loader executable finds them. Because a typical SoS installation is not located in any default search paths, you must add the `${SoS_install_dir}/sos` directory to the list of search paths.

On most Linux systems, you can do this either by using the **`LD_LIBRARY_PATH`** environment variable or by adding the search path to the `/etc/ld.so.conf` file. Windows follows different strategies with different search orders, depending on operating system settings. In most cases, the directory from which the application was loaded, the current directory, and the directories that are listed in the **`PATH`** environment variable are scanned. For more information, refer to the manual for your operating system.

The python example is written for version 2.7.x. It uses the **`ctypes`** module to access the **`FMOPSolver`** dynamic linked library API. Due to its dependencies to other libraries, you mus ensure that the loader executable can find these dependencies. Refer to the previous paragraph for help.

The next two sections introduce the **`FMOPSolver`** API using a general workflow example and a Python example.

General Workflow Example

A typical **`FMOPSolver`** dynamic linked library workflow executes API calls in the following order:

1. Add an additional license search path if your license file is not placed at a default location. Default locations are listed in the developer documentation. For more information, refer to `FMOP_appendLicenseSearchPath`.

2. Acquire a license. The library cannot be used without a valid license.

3. Load the SoS database containing the field-MOP of interest.

4. Get a handle to the field-MOP in question.

5. Get the parameters and evaluate the field-MOP, repeating this step as needed

6. Release the field-MOP handle.

7. Release the database handle.

8. Release all licenses. You cannot release acquired licenses until all **`fmop_handle_*t`** handles are successfully released.

Demonstration examples in `MOPSolver/share` show this workflow for C, C++, and Python code. For more information, see the Python example.

Note:

- Be sure to load SoS databases only once. Because a database file is usually quite large, loading it requires a significant amount of time and resources. Therefore, you should keep the database file handler open as long as one of its field-MOP models is needed. If you only need one field-MOP handle from a specific database, you can release this database handle immediately after loading it. However, because this field-MOP handle stores a reference on its parent object, resources are not released until the last child object is released.

- Be sure to release your license if no longer needed. Licenses are not released automatically. Licenses are guaranteed to be returned only at program or process termination.

Python Example

The installed version 2.7.x `Python` example follows in principal the suggestions of the general workflow example (p. 163). To use the C-API, the ctypes module gets imported:

```
import ctypes
```

In the **Variables** section, you must define the system environment, which is where the **FMOPSolver** dynamic linked library can be found: `CDLL_lib_path = C:\your\path\FMOPSolver0.dll`. The python example assumes that library dependencies are located in one of the default search paths, next to the **FMOPSolver** dynamic linked library. Furthermore, you must define the path to the SoS field-MOP demo or any other SoS field-MOP database, the number of field-MOP input parameters, and some parameter values set within their DOE (Design of Experiments) boundaries.

Using the following code, the current working directory gets set to the path of the **FMOPSolver** library and loaded into the SoS handle:

```
os.chdir ( os.path.dirname( CDLL_lib_path ) )
sos = ctypes.CDLL ( os.path.basename( CDLL_lib_path ) )
```

Because the **ctypes** module assumes that **int** data is to be returned by default, the subsequent line declares some other return types:

```
sos.FMOP_getLastLogString.restype = ctypes.c_char_p
```

Enum variable definitions are not known by python and are therefore redefined for convenience. Furthermore, the function **CheckError** and the class **LicManager** are also for convenience only. The first one checks the **FMOPSolver** API returned error code and throws an error. The latter one closes all open handles and returns the license on error.

To acquire a license:

```
sos.FMOP_appendLicenseSearchPath ( user_lic_path )
sos.FMOP_acquireLicenses ( FMOP_MESH_ALL )
```

To load the database of interest into the database handle:

```
database = ctypes.c_void_p()
sos.FMOP_loadDbFile ( ctypes.byref( database ), db_path )
```

As noted in the general workflow example (p. 163), you should keep the database handle in scope as long as its field-MOP models get used. You should not load and unload the SoS database for each FMOP handle.

To get a list of known field-MOP identiers:

1. Call **sos.FMOP_getModelIdents (database, FMOP_NODE_DATA, ctypes.byref(fmop_idents), ctypes.byref(num_idents))**.
2. Print the **fmop_idents** array:

   ```
   fmop = ctypes.c_void_p()
   sos.FMOP_getModel ( database, FMOP_ELEMENT_DATA, fmop_ident, ctypes.byref( fmop ) )
   ```

An element field-MOP handle with the given **fmop_ident** is loaded into your prepared **fmop** variable.

At this point, you can query some interesting properties, like the **FCOP[total]** value or the number of active parameters and the model dimension. The latter two parameters are of special interest because they are needed to prepare input arrays for a field-MOP evaluation call. For example, the following code queries the needed field-MOP input array sizes and evaluates the **`fmop`** handle at the given **`param_values`** point:

```
param_idents = ctypes.POINTER( ctypes.c_char_p )()
num_idents = ctypes.c_ulonglong(0)
sos.FMOP_getModelParamIdents ( fmop, ctypes.byref( param_idents ), ctypes.byref ( num_idents ) )

num_mesh_items = ctypes.c_ulonglong(0)
sos.FMOP_getModelDim ( fmop, ctypes.byref ( num_mesh_items ) )

param_values = ( ctypes.c_double * num_idents ) ( 1., 2., 3., 4., 5., 6. )
approx_field = ( ctypes.c_double * num_mesh_items.value ) ()
sos.FMOP_approxField ( fmop, param_values, ctypes.byref( approx_field ) )
```

This can be done as often as a different **`param_values`** combination of interest exists and these parameters are within their DOE boundaries.

Finally, to release all handles and licenses:

```
sos.FMOP_releaseModel ( ctypes.byref ( fmop ) )
sos.FMOP_releaseDb ( ctypes.byref ( database ) )
sos.FMOP_releaseLicenses ()
```

MATLAB Example

The **`FMOPSolver`** shared library can be used inside MATLAB™. The provided example was tested on Microsoft Windows using MATLAB R2015b with MinGW64/GCC4.9 as the MEX compiler.

MATLAB provides a simple mechanism for binding external libraries. It parses the API and links against the dynamic library by declaring the API through the ANSI C header file. The command **`loadlibrary`** is used to import the library:

```
loadlibrary ( CDLL_lib_file ,'fmop_solver.h')
```

You can call functions in the dynamic library:

```
calllib ( CDLL_lib_file, function_name, parameter_1, parameter_2, ... )
```

To acquire a license:

```
calllib ( CDLL_lib_file, 'FMOP_appendLicenseSearchPath', user_lic_path );
errno = calllib ( CDLL_lib_file, 'FMOP_acquireLicenses', feature );
FMOPSolver_checkError ( errno, 'Error acquiring licenses', calllib( CDLL_lib_file, 'FMOP_getLastErrorString' ) );
```

The function **`FMOPSolver_checkError`** is a MATLAB function provided in a separate example file for convenience.

To load the database of interest into the **`database`** handle:

```
database = libpointer( 'voidPtr' );
errno = calllib ( CDLL_lib_file, 'FMOP_loadDbFile', database, db_path );
```

You should keep the **`database`** handle in scope as long as its field-MOP models get used. You should not load and unload the SoS database for each field-MOP handle. To get a list of known field-MOP identifiers:

```
num_idents = uint64 (0);
[errno, ~, num_idents] = calllib ( CDLL_lib_file, 'FMOP_getModDim', database, 'fmop_node_data', num_idents );
```

To get the number of identifiers and then access the string identifier of each field-MOP:

```
ident = calllib ( CDLL_lib_file, 'FMOP_getModelIdent', database, 'fmop_element_data', uint64(i) );


fmop = libpointer( 'voidPtr' );
calllib ( CDLL_lib_file, 'FMOP_getModel', database, 'fmop_element_data', fmop_ident, fmop )
```

An element field-MOP handle is loaded with the given **`fmop_ident`** into your prepared **`fmop`** variable. At this point, you can query some interesting properties, like the **`FCoP[Total]`** value or the number of active parameters and the model dimension. The latter two parameters are of special interest because they are needed to prepare input arrays for a field-MOP evaluation call.

For example, the following code queries the needed field-MOP input array sizes and evaluates the **`fmop`** handle at the given **`param_values`** point:

```
[errno, ~, num_idents] = calllib ( CDLL_lib_file, 'FMOP_getModelParamIdentsDim', fmop, num_idents );
for i = 1 : num_idents-1
    param_ident = calllib ( CDLL_lib_file, 'FMOP_getModelParamIdent', fmop, uint64(i) );
    ...
end
num_mesh_items = uint64 (0);
[errno, ~, num_mesh_items] = calllib ( CDLL_lib_file, 'FMOP_getModelDim', fmop, num_mesh_items );
param_values = [ 43341.092 134.12064 0.093418892 0.042747076 2.7525392e-09 58.798039 ];
approx_field = libpointer( 'doublePtr', zeros ( num_mesh_items, 1 ) );
calllib ( CDLL_lib_file, 'FMOP_approxField', fmop, param_values, approx_field );
% approx_field.Value ( 1:10, 1 )
```

This can be done as often as a different **`param_values`** combination of interest exists and these parameters are within their DOE boundaries.

Finally, to release all handles and licenses:

```
calllib ( CDLL_lib_file, 'FMOP_releaseModel', fmop );
calllib ( CDLL_lib_file, 'FMOP_releaseDb', database );
calllib ( CDLL_lib_file, 'FMOP_releaseLicenses' );
```

Chapter 12: SoS Theory

The following topics provide the theory for these SoS algorithms:

12.1. Import Algorithm

During the import of data, for each design, one of these behaviors occurs:

- The geometric deviations between the measurement (the design) and the reference mesh are determined.
- A field data object is transferred to the reference mesh.

The following steps are performed during the import:

1. Import the reference mesh.
2. For each design:
    a. Import the design mesh and design field data.
    b. Optionally change the position of the design mesh.
    c. Perform a geometric search that associates segments and nodes of both meshes to each other.
    d. Optionally determine the coordinate deviations between both boundaries.
    e. Optionally project the coordinate deviation vectors onto the normal vectors of the boundary of the reference mesh (recommended for small deviations).
    f. Map field data from the design mesh topology to the reference mesh.

12.2. Geometric Search and Computation of Geometric Deviations Algorithms

Determining the coordinate deviations between two *non-matching* or incompatible meshes is particularly challenging and requires highly sophisticated algorithms. Mapping fields between two meshes with differing geometry and topology is achieved with some form of projection. To perform mesh mapping, SoS provides multiple projection algorithms, each with its own set of advantages and disadvantages.

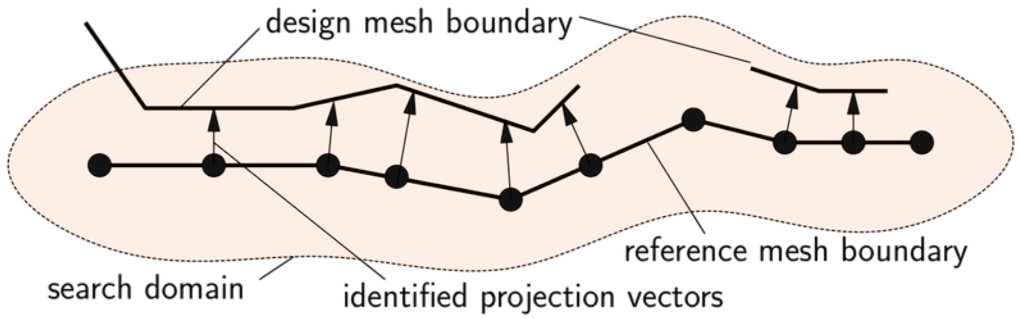
12.2.1. Compatible Mesh Mapper

The compatible mesh mapper's algorithm assumes that the topology of the design mesh is equal to or a subset of the reference mesh’s topology. It does not perform any projection among non-matching meshes. In this context, *mesh topology* refers to the connections and identifiers of nodes and finite elements.

The compatible mesh mapper can detect and import eroded elements.

12.2.2. Closest Point Projection Based on Smooth Boundary

The closest point projection based on a smooth boundary implements the node-to-segment projection search algorithm "Inside-Outside test" by Wang et al. [13] (p. 179).



The algorithm first associates each node on the reference mesh boundary with a normal vector. This nodal normal vector is obtained as a weighted average of the normal vectors of the adjacent boundary segments. If the FEM mesh is a volume, the nodal normal vector points outside.

Projection points on the design mesh are found by extending the nodal normal vector in either direction. Along these directions the algorithm searches for intersections with the design mesh’s boundary segments. Only segments within a user-specified maximum search distance are considered. If the intersection search fails, a missing item is created instead. If multiple segments are intersected along the given direction, the segment within the shortest distance is chosen. Only segments equally oriented in space are matched. This means that they point into the same half-space. (The dot product of the reference mesh nodal normal vector and the design mesh boundary segment normal vector is positive.)

With thin-walled structures defined by shell elements, each shell element is associated with two boundary segments (upper and lower side: SPOS and SNEG).

Finally, the actual deviation vector is defined as the vector between the reference node and the intersection point on the design mesh segment. The normal coordinate deviation is the magnitude of this vector. The sign of the normal deviation is positive if the dot product of the deviation vector and the reference mesh’s nodal normal vector is positive.

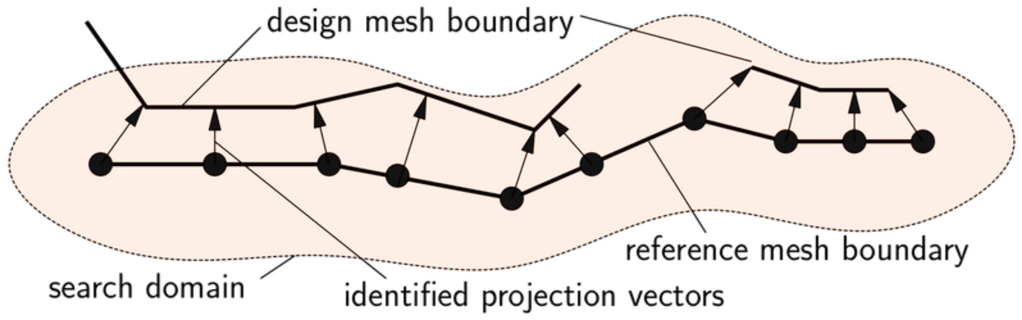
Features

- Requires well defined orientation of shell elements in FEM meshes.
- May not detect a projection near certain corner situations.

- Can treat holes, overlaps, and cut-offs. It does not identify a projection in such cases.

12.2.3. Explicit Closest Point Projection

The explicit closest point projection implements the point-to-plane closest point projection by Hallquist et al. [6] (p. 179).



The algorithm first identifies all segments on the design mesh boundary that are in the neighborhood of any node on the reference mesh boundary. On these candidate segments, the point closest to the reference node is determined. If this closest point is inside of a segment, the resulting deviation vector is always perpendicular to the design mesh boundary segment. Otherwise, the projection point is located on a segment's edge or vertex.

The algorithm only considers segments within a user-specified search distance. If multiple segments are intersected along the given direction, the segment with the shortest distance is chosen. Only segments equally oriented in space are matched. (The dot product of the reference mesh nodal normal vector and the design mesh boundary segment normal vector is positive.)

To simplify the generation of new geometries, the actual deviation vector is defined as the projection of the vector to the closest point (on the matched segment) onto the nodal normal vector of the reference mesh boundary. The normal coordinate deviation is the magnitude of this projected vector. The sign of the normal deviation is positive if the dot product of the deviation vector and the reference mesh's nodal normal vector is positive.

Features

- Requires well defined orientation of shell elements in FEM meshes.
- Always detects a projection within the search distance. It can also find projection points in vicinity of holes, cut-offs, and so on.
- Search distance influences the numerical efficiency of the search. Ideally, the magnitude of the search distance should be close to the average finite element size.

Note:

- Try to limit the search distance such that you find a projection in a reasonable amount of time. When starting with small values, check the maximum found deviation. If it is too close to the maximum search distance, increase the distance and repeat the projection.

- You should visualize the imported deviations for each design in SoS. There, you can see the regions for which no projection could be detected.

12.2.4. Automatic Positioning

Because the coordinate systems of the measurements and the reference mesh may be different, the imported design mesh is repositioned before determining the coordinate deviation. SoS supports the following algorithms:

Coarse coordinate transformation (Auto translate)

Determines the center points of gravity. The design mesh is translated such that the center points are equal.

Coarse coordinate transformation (Auto translate + rotate)

Determines the center points of gravity and the inertia tensors of both meshes. The design mesh is translated such that the center points are equal. Then the design mesh is rotated such that the error between the two inertia tensors is minimized.

Note:

The coarse rotation is not unique. For nearly symmetric bodies in particular, the result of this rotation may be wrong. You should visualize and check the results.

Fine coordinate transformation (Auto translate + rotate with best fit)

Combines a pre-alignment adjustment as described for the previous algorithm and runs a modification of the Iterative Closest Point (ICP) Projection algorithm for surface registration afterwards. The ICP Projection algorithm also uses an iteration to minimize the actual "error" between both geometries. Each iteration step involves the computation of the geometric deviations between both meshes. For more information, refer to [10] (p. 179).

Pre-aligned fine coordinate transformation (Auto translate + rotate with best fit)

Assumes already pre-aligned meshes and runs only the modified ICP projection algorithm as described for the previous algorithm.

The algorithms may also be instructed to work with a node subset only. Then the center of gravity and the inertia tensors are computed only for these nodes that are included by the subset. This option further limit iterative algorithms, particularly the rotation and the whole ICP algorithm, to build their optimization criteria only with nodes included in the subset given. It might further occur that the transformation reference node set is only defined in the reference mesh, but it is missing in the design mesh file. Such a case splits the actual coordinate transformation into three principle steps:

1. Perform a course alignment. Depending on the chosen transformation algorithm, a translation and rotation based on all nodes is applied.

2. Use the user-defined mesh mapper algorithm to project the reference node set onto the design mesh.

   Note:

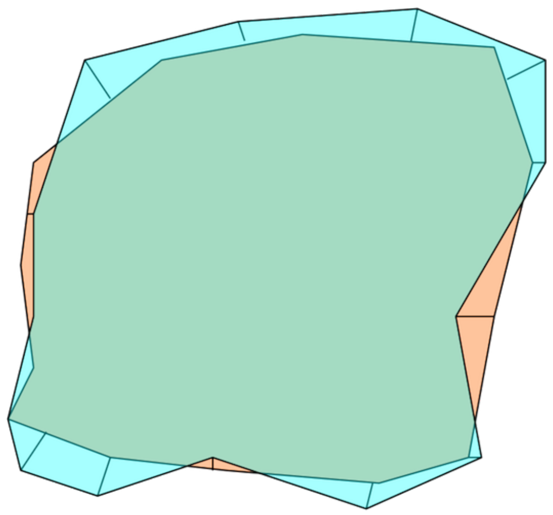
   The user-specified direction mesh mapper might be a poor choice for this task. Furthermore, be aware that SoS typically uses all nodes for the projection step and postprocesses this information only. The compatible mesh mapper is currently the only one that runs the projection step already on the reduced node set.

3. With node subsets defined for both meshes, the chosen transformation algorithm is applied.

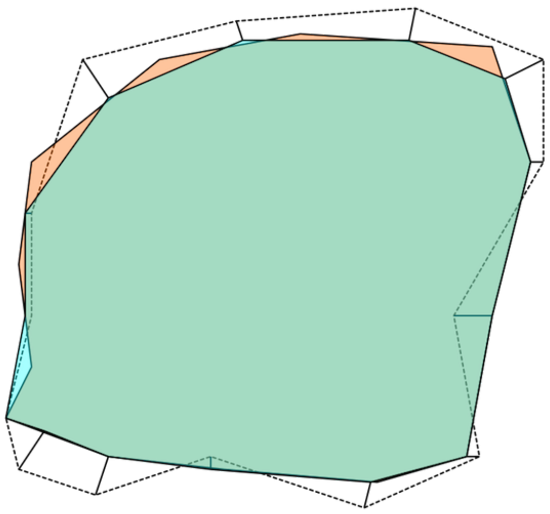
12.2.5. Mapping of Data

The mapping of data among non-matching meshes consists of these steps:

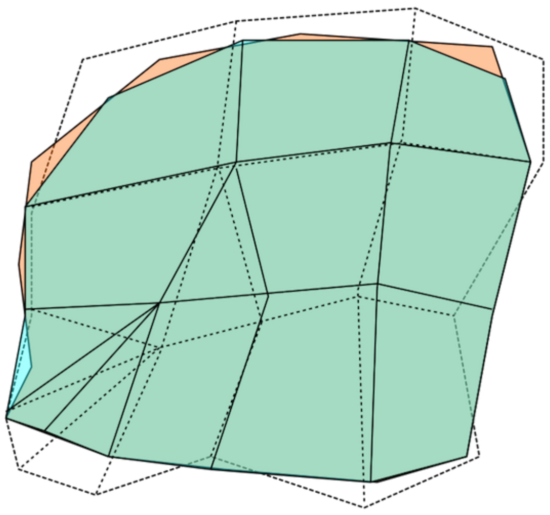
1. Find closest point projection by projecting boundary points of the reference mesh onto the boundary of the design mesh.



2. Create a deformed boundary of the reference mesh. The objective is to represent the geometry of the design mesh using the reference mesh topology.



3. Apply displacements to the interior nodes of the reference mesh to represent the changed shape of boundary.



4. Map data between both meshes (coordinate deviations, strain, and stress).

Chapter 13: SoS Troubleshooting

These topics provide information about known issues and limitations and about dealing with unforeseen events:

Note:

For information about obtaining technical support, see ANSYS Support (p. 12).

13.1. Known Issues and Limitations

Known issues and limitations are organized as follows:

13.1.1. General Issues and Limitations

The following issues and limitations can occur, regardless of your operating system:

Error in optiSLang Custom Integration Node

SoS is not correctly installed if the tooltip to the error message on the **optiSLang Logger** tab states:

```
Cannot find the environment variable 'SoS_exe'[...]
```

Step: For Windows, refer to Missing environment variable (p. 174) in the next topic. While this environment variable is also needed for Linux, it is assumed that Linux users know how to set environment variables.

Errors in optiSLang SignalMOP nodes

The integration of SoS in optiSLang is mostly realized through Lua and Python scripts. You can modify optiSLang such that it uses a more recent version of these scripts.

Steps:

1. On Windows, go to the public documents folder (such as `C:\Users\Public\Documents\Dynardo`).

2. Copy all files from `Statistics on Structures\`*`x.y.z`*`\examples\optislang\`*`x.y`*`\` to ANSYS `optiSLang\`*`x.y.z`*`\scripting\integrations`.

3. Restart optiSLang and reset the SignalMOP nodes.

File format is not supported

If your file format is not supported by SoS:

Steps:

- Try to convert your files to a supported format using your CAE postprocessor or preprocessor.
- Contact ANSYS Support (p. 12).

If your file format should be supported by SoS, but it is not correctly imported:

Steps:

- For LS-DYNA, check out the demonstration example files to see how your files must be formatted.
- Email your files along with a description of the intended behavior and experienced behavior to `<support@ansys.com>`.

13.1.2. Issues and Limitations Unique to Windows

If you are running on Windows, you should be aware of the following limitations:

Random crashes or black rendering of windows

SoS crashes or displays black windows. The last message is **`D3D11 disconnected`** or something similar. In this case, Microsoft Windows installed graphics drivers in the background. A reboot of the computer system typically resolves this issue.

Missing environment variable

SoS was installed for the current user only by switching to **`Install just for me`** during the SoS setup. You must add the default environment variable manually.

Steps:

- Open the **Edit environment variables for your account** control panel and then either press the Windows key and type **`environment variables`** to search for the control panel or ask your system administrator for assistance.
- Add a new variable, setting **Variable name** to **`SoS_exe`** and **Variable value** to **`%LOCALAPPDATA%\Programs\Dynardo\Statistics on Structures\3.3.1\sos.exe`**. Adapt this default path as needed.

No Direct3D

SoS is started in batch mode (headless mode) without the graphics system enabled. This happens, for example, in ANSYS EKM. Because SoS requires availability of Direct3D, it cannot be executed in pure batch mode.

Black windows in Microsoft Remote Desktop Client

When using Microsoft Remote Desktop protocol to connect to a Microsoft Windows client, SoS sometimes renders a black window in visualization. In this case, the Direct3D information is not correctly transferred to the host system.

Steps:

- Try to change the session settings of your remote desktop software.
- Sometimes, it helps to change the graphics color depth to the maximum (32 bit).

13.1.3. Issues and Limitations Unique to Linux

If you are running on Linux, you should be aware of the known issues and limitations:

Sudden crash at startup

SoS suddenly crashes after successfully acquiring license features.

Steps::

- Verify that the hidden subdirectory `$DEST/sos/.sos_help` exists in the SoS installation directory $DEST. If SoS is not called directly using an absolute path (such as `"/opt/Dynardo/Statistics on Structures/sos/sos.sh"`) but rather is located automatically from the system (such as by typing **`sos`** in a terminal), you might locate the installation path typing **`which sos`** in a terminal.
- If the hidden subdirectory `$DEST/sos/.sos_help` does not exist, the administrator must either create it or change directory permissions. For more information, see Installation on Linux (p. 9).

Error message displays: QXcbConnection: Could not connect to display

When running SoS in batch mode (**`-b`**), a running X server is still required. This is because SoS has the ability to export images in batch mode.

Solution:

Set the environment variable **`DISPLAY`** to the display of your X server. For example:

```
# Get running X servers:
ps aux|grep X

# output:
root     119066  0.0  0.2 137904 35064 tty7     Ss+  09:46   0:01 /usr/bin/Xorg :1

# set DISPLAY
export DISPLAY=:1
```

On systems without display hardware, Xvfb - virtual framebuffer X server can be used to emulate a framebuffer using virtual memory.

Fonts are unavailable

SoS displays weird characters instead of useful text. In the terminal, it displays: **`Cannot find font directory /usr/local/Qt-5.4.2/lib/fonts`** on startup of SoS.

Reason: ASoS uses the default Qt installation of Redhat Enterprise Linux 6, which is Qt 5.4.2. This version has a static reference to the location of font files.

Workarounds:

- Delete the '`Qt*.so`' files from the subdirectory `lib` in your SoS path and delete the subdirectories `iconengines`, `imageformats`, `platforms`, and `sqldrivers` in the subdirectory `sos` in your SoS installation path.
- Copy a few font files into the specified directory.
- Wait for a fix in a future SoS release.

X11 errors and terminate at startup

SoS requires at least OpenGL 2.2 with shader support. You may either need a graphics card that supports it through appropriate drivers or MESA emulation rendering OpenGL using the CPU. Further, OpenGL transmissions over network are not supported by many available X clients and servers.

Steps: Send ANSYS Support (p. 12) an incident report (p. 177) that includes a description of your problem along with screen shots and information on your graphics card and graphics driver, including version numbers and dates.

Reasons:

- Too weak of a graphics card
- Too large of a FEM mesh
- Too little RAM

Steps:

- Use the fast rendering mode when changing camera position by mouse instead of the default rendering mode. In the fast rendering mode, only outlines are drawn while changing the camera. In the default rendering mode, outlines and surfaces are drawn while changing the camera.
- Reduce the size of the scene, either by reducing the size of the SoS main window or by selecting **Show 4 scenes** instead of **Show 1 scene**.
- Before loading any data into SoS, run the following script command to disable visualization of the interior of structures:

  ```
  sos.sceneManager():setGraphicsSlicingEnabled(false)
  ```

Additionally, you can run this script command to disable the whole graphics system:

```
sos.sceneManager():setGraphicsEnabled(false)
```

13.2. Unforeseen Events

These topics describe how to deal with unforeseen events, which are typically crashes:

13.2.1. Responding to Crashes

When a crash occurs, you can email an incident report (p. 177) with supporting files to <support@ansys.com>.

Crashes on Windows

If a crash occurs when running on Windows, restart your project with the debug executable and reproduce the crash. You can find the debug executable in the installation path (`sos_console.exe`) or using the environment variable **`SoS_debug_exe`**. When you start this command from a terminal, you get the complete traceback and possibly even error messages that can occur before the log system of SoS starts.

Crashes in optiSLang integrations

If a crash in an optiSLang integration occurs, attach the file in the subfolder `sos_log` from the respective working directory to the incident report. For example, the **`SignalMOP`** node creates such a folder in `project.opd/SignalMOP/sos_log`.

Crashes in the ACT extension for ANSYS Workbench

If a crash occurs in the ACT extension for ANSYS Workbench, add a user environment variable named `SOS_ANSYS_LOG` and set the value to `1`. This causes the ACT plugin to write a log file to `%USERHOME%\AppData\Local\Dynardo\Statistics on Structures` on Microsoft Windows, including any debug messages.

13.2.2. Filing Incident Reports

To file an incident report, send an email to `<support@ansys.com>` with a precise and reproducible description of the usage scenario, the expected behavior, and the experienced behavior.

For a crash, your report should contain:

- Name and version number of your operating system.
- Description of the SoS version found in the **About** dialog box.
- Files `log_messages.txt`, `autosave.ssc`, and `processed_commands.txt`. You can find these files in:

  - `sos_log` in the working directory
  - `%USERHOME%\AppData\Local\Dynardo\Statistics on Structures` on Microsoft Windows
  - `$HOME/.local/share/data/Dynardo/Statistics on Structures` on Linux
  - `sos_log` directory if the present working directory that SoS was started from is writeable to the executing user

  Note:

  If the last commands in `autosave.ssc` and `processed_commands.txt` differ, the crash very likely appeared within the respective command. If they are equal, the crash appeared either in the rendering engine or while invoking some GUI events. In this case, try to remember and include in your report what you did with the mouse and keyboard during the crash, what GUI element got the keyboard focus, and what data objects were visible in the GUI tables.

- Description of the project, including the number of nodes and elements in the reference mesh and the file formats that were used.

13.2.3. Recovering from Crashes

SoS stores an autosave script in the paths mentioned in the previous topic. The file, `autosave.ssc`, contains all script commands that were executed successfully. To recover from a crash, you can restore your work:

1. Copy the file `autosave.ssc` to a new location.
2. Restore the session by running this new script file when starting SoS.

A likely error while re-executing the autosave script is that certain output files did not exist in the first run, but in the second run, SoS indicates that it cannot replace them. In such a case, you have two options:

- Delete the output files before re-executing the script
- Modify the Lua script (p. 141). In the respective settings classes, change the behavior of creating output files using:

```
settings.replace_files = true;
```

References

[1] ANSYS, Inc.. (2015). *ANSYS Mechanical APDL Element Reference*. 16th Edition.

[2] ANSYS, Inc.. (2015). *ANSYS Mechanical APDL Programmer's Reference*. 16th Edition.

[3] ANSYS, Inc.. (2017). *ANSYS Mechanical APDL Command Reference*. 18th Edition.

[4] ANSYS, Inc.. (2017). *ANSYS Mechanical APDL Programmer's Reference*. 18th Edition.

[5] Dassault Systèmes. (2014). *Abaqus Analysis User's Guide*. 18th Edition.

[6] Hallquist, J. O., G. L. Goudreau, and D. J. Benson. (1985). Sliding interfaces with contact-impact in large-scale Lagrangian computation. *Computer Methods in Applied Mechanics and Engineering 51(1/3)*. 107-137.

[7] Inges GmbH, Stuttgart. (2012). *PERMAS - User's Reference Manual*. PERMAS Version 14.00.334 Edition.

[8] Kitware, Inc.. (2010). *The VTK User's Guide*. 11th Edition.

[9] Livermore Software Technology Corporation. (2013). *LS-DYNA Keyword User's Manual, Volume I*. Release 7.0 Edition.

[10] Kok-Lim Low. (2004). Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration . *Technical Report TR04-004*. Department of Computer Science, University of North Carolina at Chapel Hill, North Carolina.

[11] MSC.Software Corporation. (2003). *MSC.Nastran 2004 - Reference Manual*.

[12] MSC.Software Corporation. (2011). *MSC.Nastran 2012 - Quick Reference Guide*.

[13] Wang, S. P. and E. Nakamachi. (1997). The inside–outside contact search algorithm for finite element analysis. *International Journal for Numerical Methods in Engineering 14(19)*. 3665-3685.

[14] Wikipedia Foundation. (2014). Wikipedia: STL (file format). Last opened 2014/03/10.

[15] Shafranovich, Y. (2005). *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. October.