InsightCAE User Manual

May 11, 2020

Contents

1 Introduction

1.1 License and Copyright

1.2 Contributors

The following people have so far contributed to InsightCAE:

- Hannes Kröger (silentdynamics GmbH)
- Johann Turnow (silentdynamics GmbH)

If you want to get involved in the development, please feel invited to do so! We greatly appreciate any contribution and we would very much like to add you to the above list.

If you made some modification or addition to the code, which you would like to be merged into the main development line, please consider to send us a pull request[2].

1.3 About InsightCAE

InsightCAE is a workbench for Computer-Aided Engineering. Individual open source projects often meet only subtasks in analysis process. To automate repetitive computational tasks, a combination of several open source engineering software tools is often required. In order to use open source software productive and efficient for everyday tasks, the analysis automation framework InsightCAE was created.

InsightCAE serves as a framework for the implementation of analysis procedures. The objective is to provide interfaces to the tools and simulation programs that are needed for a specific computing task.

1.4 Features and Highlights

InsightCAE's objective is to create automated analysis workflows. The high level API resides in the core "toolkit" library. Automated workflows usually involve different external programs and utilities. For the realization of automated workflows, it is sometimes required, to create add-ons to these external programs. Thus InsightCAE is also a container for add-ons to other programs.

- InsightCAD script-based, fully parametric CAD

[1] http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

[2] https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-pull-requests

  - OpenCASCADE geometry kernel
  - assemblies, constraint-based sketches, part library, drawing export

- OpenFOAM add-ons (schemes, boundary conditions, models, ...)
- Pre- & Postprocessing tool (OpenFOAM Case Builder)
- analysis workflow automation tools (GUI)

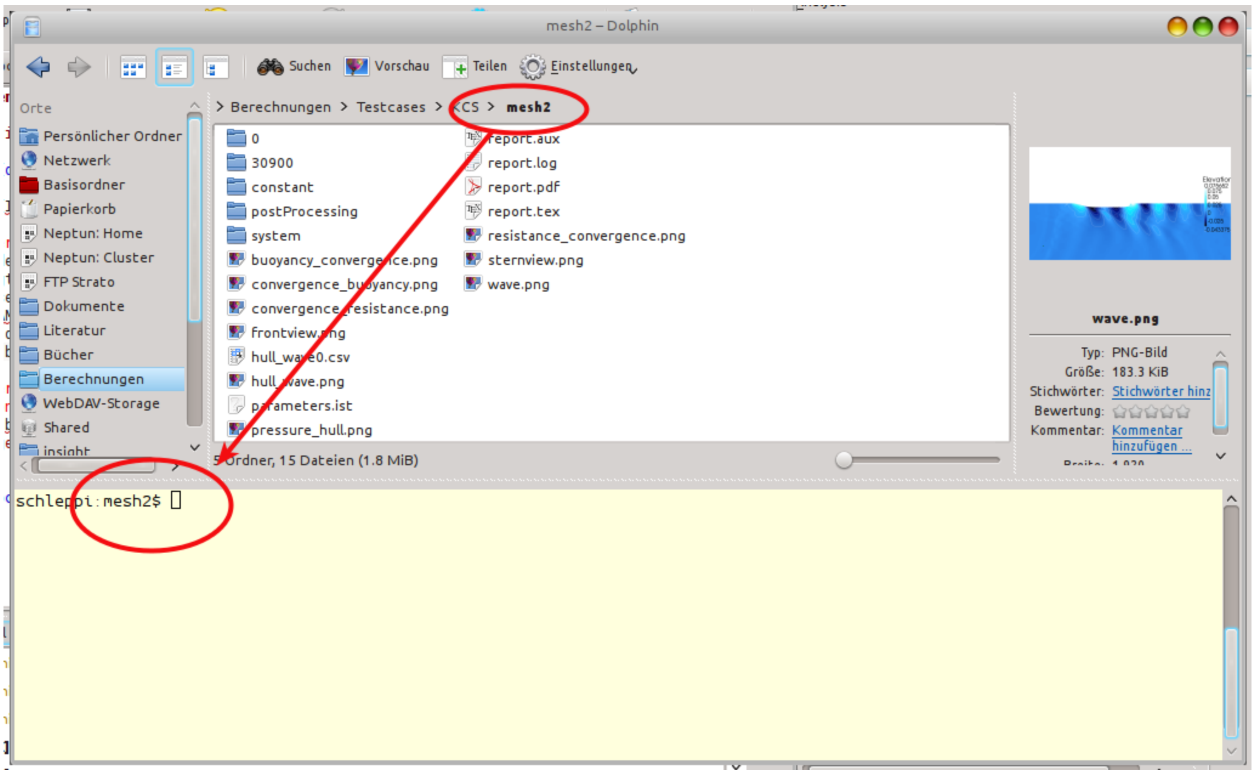
1.5 Reading this Manual

1.6 Recommendations for Working with Shell-based Tools in Linux

Sometimes it is necessary to execute tools in a bash shell, e.g. because no GUI for it is available. And often it is easier to keep an overview in graphical file manager. Having a console window open together with a file manager window at the same time is an obvious solution but to do so for many cases at a time may easily confuse the desktop.

Here are two useful hints to solve this issue:

1. The KDE file manager **dolphin** offers the possibility to display a console in the lower half of the window (figure 1). The working directory is synchronized with the directory shown in the graphical window by injection of **cd** commands. Vice versa, when the working directory is changed in the shell, the graphical display is updated as well.



0, 30900, constant, postProcessing, system, buoyancy_convergence.png, convergence_buoyancy.png, convergence_resistance.png, frontview.png, hull_wave0.csv, hull_wave.png, parameters.ist, pressure_hull.png, report.aux, report.log, report.pdf, report.tex, resistance_convergence.png, sternview.png, wave.png

5 Ordner, 15 Dateien (1.8 MiB)

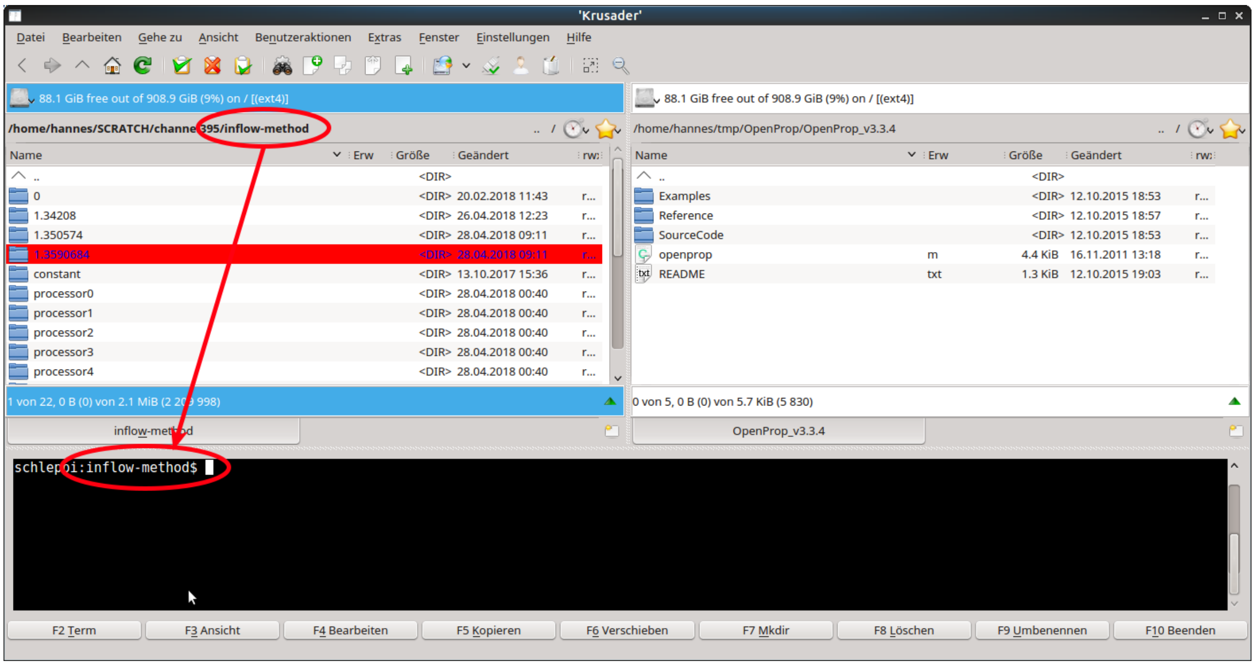
Typ: PNG-Bild
Größe: 183.3 KiB
Stichwörter: Stichwörter hinz
Bewertung: ☆☆☆☆☆
Kommentar: Kommentar hinzufügen ...

```
schleppi:mesh2$
```

Figure 1: Dolphin file manager with embedded terminal

2. There is a Norton-Commander-like file manager named **krusader** (figure 2). It offers the same functionality regarding the embedded terminal but a more flexible way of displaying multiple folders. In addition to the two list view on the left and right, multiple tabs for folders

can be added in each list view. And there is also a rich interface to define custom commands and file associations.



/home/hannes/SCRATCH/channe395/inflow-method

Name	Erw	Größe	Geändert	rw:
..		<DIR>		
0		<DIR>	20.02.2018 11:43	r...
1.34208		<DIR>	26.04.2018 12:23	r...
1.350574		<DIR>	28.04.2018 09:11	r...
1.3590684		<DIR>	28.04.2018 09:11	r...
constant		<DIR>	13.10.2017 15:36	r...
processor0		<DIR>	28.04.2018 00:40	r...
processor1		<DIR>	28.04.2018 00:40	r...
processor2		<DIR>	28.04.2018 00:40	r...
processor3		<DIR>	28.04.2018 00:40	r...
processor4		<DIR>	28.04.2018 00:40	r...

1 von 22, 0 B (0) von 2.1 MiB (2 203 998)

/home/hannes/tmp/OpenProp/OpenProp_v3.3.4

Name	Erw	Größe	Geändert	rw:
..		<DIR>		
Examples		<DIR>	12.10.2015 18:53	r...
Reference		<DIR>	12.10.2015 18:57	r...
SourceCode		<DIR>	12.10.2015 18:53	r...
openprop	m	4.4 KiB	16.11.2011 13:18	r...
README	txt	1.3 KiB	12.10.2015 19:03	r...

0 von 5, 0 B (0) von 5.7 KiB (5 830)

```
schlepi:inflow-method$
```

Figure 2: Krusader file manager with embedded terminal

1.7 Resources

1.7.1 In the Web

- Source Code Repository https://github.com/hkroeger/insightcae
- Issue Tracker https://github.com/hkroeger/insightcae/issues
- Web Forum https://groups.google.com/forum/#!forum/insightcae

1.7.2 Reporting Bugs and Feature Requests

Please use the issue tracker https://github.com/hkroeger/insightcae/issues to report any bugs.

We also monitor the web forum https://groups.google.com/forum/#!forum/insightcae for questions or feature requests.

1.7.3 Getting Professional Help and Support

Beyond the web resources above, silentdynamics GmbH offers commercial support[3] for professional users of InsightCAE. Automated analyses according to customer needs and specifications are implemented by creating new specialized modules for Insight CAE. Typical support contracts include also user training and continuous customization and updating of InsightCAE and its add-ons.

[3] http://silentdynamics.de/en/oss-cae/

2 Obtaining InsightCAE

2.1 Installation using Linux Package Manager

We provide installation packages for Ubuntu-based Linux distributions. We specifically support the latest Ubuntu LTS distibution.

The packages are held in our repository. To install them, first add the silentdynamics apt repository and the associated key by executing:

```
$ sudo apt-key adv --fetch-keys http://downloads.silentdynamics.de/
  SD_REPOSITORIES_PUBLIC_KEY.gpg
$ sudo add-apt-repository http://downloads.silentdynamics.de/ubuntu
$ sudo apt-get update
```

Then, install the software by executing

```
$ sudo apt-get install insightcae-base
```

Note: in Ubuntu 18.04 LTS, the file `/etc/ImageMagick-6/policy.xml` needs to be modified. Otherwise the charts in the reports will not be created properly. Find the line

```
<policy domain="coder" rights="none" pattern="PDF" />
```

and change it to

```
<policy domain="coder" rights="read|write" pattern="PDF" />
```

2.2 Installation on Windows

Currently, there is no native Windows version of InsightCAE available. However, it can be run in Windows 10 Enterprise (64 bit only) or Windows Server using the "Linux Subsystem for Windows" (WSL). This should be preferred over virtualization solutions like VirtualBox, because it offers the best parallel performance. The processes share the memory with the Windows processes and the files are stored in the Windows file system. Various X servers are available for graphical output under Windows; Xming is used below.

To use this solution, the Linux subsystem must first be activated. Adminstration privileges are required. It is done by the following steps:

Start Windows PowerShell as an administrator (right-click on Start menu >"PowerShell (Admin)") and execute:

```
> Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-
  Linux
```

Restart the computer when prompted.

Download the Linux image:

```
> Invoke-WebRequest -Uri https://aka.ms/wsl-ubuntu-1804 -OutFile ~/Ubuntu.zip -
  UseBasicParsing
```

Unpack:

```
> Expand-Archive ~/Ubuntu.zip c:\Distros\Ubuntu
```

Configuring the Linux System: Switch to user "root":

```
> c:\Distros\Ubuntu\ubuntu.exe config --default-user root
```

Set the password of the user "user":

```
> c:\Distros\Ubuntu\ubuntu.exe
# passwd user
```

Install InsightCAE with OpenFOAM:

```
# echo deb http://downloads.silentdynamics.de/ubuntu bionic main > /etc/apt/
    sources.list.d/sd.list
# apt-key adv --fetch-keys http://downloads.silentdynamics.de/
    SD_REPOSITORIES_PUBLIC_KEY.gpg
# apt-get update
# apt-get install insightcae-base
```

Switch back to standard user (”user”):

```
> c:\Distros\Ubuntu\ubuntu.exe config --default-user user
```

Add the file `c:\Distros\Ubuntu\ubuntu.exe` to the start bar
Then install the Xming server for graphical output:
Download Xming-Installer[4], then execute the installer:

- Check: ”Install quick launch icon”
- Allow network access after installation

Set the DISPLAY variable in the Bash environment

```
$ sudo -i
# echo "export DISPLAY=:0" >> /etc/profile.d/xming.sh
# exit
```

Usage First start ”Xming” X-Server, then start Ubuntu Bash command line. Execute the application in the shell, e.g. the InsightCAE Workbench:

```
$ workbench
```

2.3 Building from Sources

The source code of InsightCAE is hosted at Github. Cloning a working copy constitutes the best way to get updates and the latest bug fixes.

Alternatively, you can download snapshot archives from github, if you don’t want to bother with a git client. In this case, just replace the cloning step below with unpacking the archive.

First, get the the sources by cloning the git repository:

```
$ git clone https://github.com/hkroeger/insightcae.git insight-src
```

CMake is utilized for managing the build. The preferred way is to build the software out of source in a separate build directory. Create a build directory, then configure the build using e.g. ccmake and finally build using make:

```
$ mkdir insight && cd insight
$ ccmake ../insight-src
```

[4]e.g. https://sourceforge.net/projects/xming/files/Xming/6.9.0.31/Xming-6-9-0-31-setup.exe

In the CMake GUI, you may need to change paths or adapt settings. Please refer to the CMake documentation[5] on how to use CMake. The biggest difficulty will probably be, to provide all the software packages, on which InsightCAE depends.

The final step in CMake is, to generate the Makefiles. Once this is done without errors, start the compilation of the project by executing:

```
$ make
```

To work with InsightCAE, some environment variables are needed to be set up. A script is provided therefore. It can be parsed e.g. in your `~/.bashrc` script by adding to the end:

```
source /path/to/insight/bin/insight_setenv.sh
```

3 Overview

3.1 Concept and Parts of the Project

The InsightCAE builds on top of available open-source engineering analysis tools (though closed-source tools could be used as well). The most used tool by far is the CFD software OpenFOAM.

For certain tasks, custom extensions to these tools are required. InsightCAE contains a selection of extensions, which are included in the distribution packages, when these are created.

The core of the project is the toolkit library. It contains the implementation of core classes in workflow management as the parameter set storage and tools and result set storage classes and handling tools. The toolkit library also maintains a list of available analysis modules, which is dynamically extensible by loadable add-ons.

Finally, on top of the toolkit library, custom analysis modules can be created. The InsightCAE base distribution contains a number of generic analysis modules as the ”Numerical Wind Tunnel” module and some generic test case analyses like flat plate, channel flow, pipe flow and so on. These might serve as examples for custom implementations.

3.2 Terms and Defintions

Analysis InsightCAE bundles tools to create automated simulation procedures. These procedures can be written in Python or C++. The term **analysis** or **workflow module** is used for such a procedure.

Actually, an analysis is like a function in programming: it gets input parameters, executed some actions and returns a results data structure.

Parameter Set The analysis modules need input data, i.e. parameters. All parameters for a module are grouped together in a parameter set. Parameter sets are stored in XML files in human readable format (extension *.ist). They can also be edited in a GUI editor (application workbench).

Many analysis modules need external files (like CAD data in STL or any other format). These files can simply be referred in a parameter set by their relative path (relative to the *.ist-file containing the reference) or absolute path. It is also possible to embed the external files into the XML file in an base64-encoded form. This simplifies transfer of analysis setup from one computer to another, e.g. for remote execution.

[5]https://cmake.org/cmake/help/latest/guide/tutorial/index.html

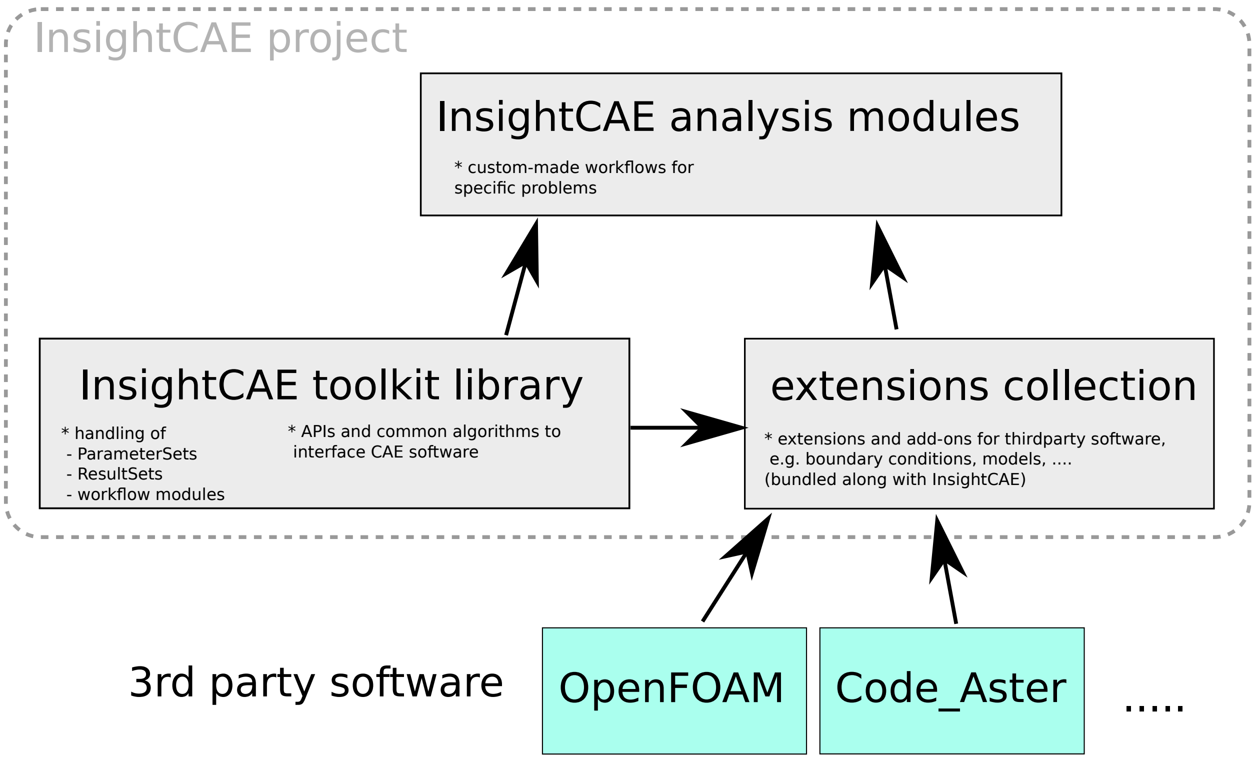

InsightCAE project
InsightCAE analysis modules
* custom-made workflows for specific problems
InsightCAE toolkit library
* handling of
- ParameterSets
- ResultSets
- workflow modules
* APIs and common algorithms to interface CAE software
extensions collection
* extensions and add-ons for thirdparty software, e.g. boundary conditions, models, ...
(bundled along with InsightCAE)
3rd party software
OpenFOAM
Code_Aster
.....

Figure 3: Structure of the InsightCAE project

Result Set The result of an analysis is returned in a **result set**. This is a hierarchical data structure which contains nodes like scalar values, charts, contour plots, images, tables or similar entities. It can be stored on disk in XML format. There is also a renderer which converts a result set into a PDF report.

Case Element The setup of CFD-runs is put together from a number of so-called **case elements**. Each case element represents some functional feature. It might be a boundary condition or a turbulence model or something similar. A case element is not necessarily associated with a certain configuration file but can modify multiple configuration files of a CFD case configuration.

There is also an editor for composing OpenFOAM cases from the available case elements: the OpenFOAM Case Builder.

- Solver extensions
- OpenFOAM Case Builder
- Workbench

4 Automation of Simulation Workflows

4.1 Building Bricks

4.1.1 Input: Parameter Sets

Parameter sets are a hierarchical collection of parameters. Parameters can be of different type, see table 1 for a list.

A parameter set constitutes the full input data set for an analysis.

Beyond these primitive parameter types, there can be compounds, namely:

- `subset`

  a subdirectory in a parameter set

- `selectablesubset`

  a `subset` that switches its content based on a `selection` parameter

- `array`

  an array of either a compound parameter or a primitve parameter

Type	Description
`double`	a scalar floating point value
`int`	an integer value
`bool`	a boolean value
`vector`	a 3-tuple of doubles
`string`	a simple text
`selection`	a selection from a predefined set
`path`	a file path, relative to the input file
`matrix`	a matrix with arbitrary number of rows and cols
`doublerange`	an array of double values

Table 1: Available primitive parameter types in a parameter set

4.1.2 Output: Results Sets

The output of an analysis is stored in a result set. It may contain elements like charts, images, tables, number etc. A result set can be saved to a XML file and/or rendered into a PDF report.

4.1.3 Analyses: the Simulation Procedure

The procedure of a simulation is contained in a so-called analyses or workflow module. You may think of it as an app for executing a certain type of simulation. It is essentially a program. Oftentimes, people create scripts or macros for speeding up their simulation workflows. The InsightCAE workflow modules serve a similar purpose, though the intention is to fully cover the whole simulation including all pre and post processing steps.

There is a GUI available which provides an editor for the input parameter set (see section 4.2.1). The GUI is also capable of displaying a 3D preview of the involved geometry and other elements, which are displayable. The analysis module is responsible of creating the 3D preview elements. Creation of the preview is optional and not every analysis module produces a preview.

The GUI also includes a viewer for the result set. From that viewer, the reports may be exported.

InsightCAE holds a runtime-extensible list of available analyses. This list can be extended at run time either by providing shared libraries containing more analysis modules or by python scripts.

The analyses can be programmed essentially in C++ or in Python. InsightCAE is essentially written in C++. Python wrappers for the relevant functions and objects are created using SWIG. So the API can also be accessed from Python.

If a new analysis shall be created, this parameter editing and result viewing infrastructure is easily reusable:

- The new analysis module just needs to provide the parameter set layout by returning the default parameter set (a parameter set filled with suitable default values),
- it has to provide a function for executing the analysis.
- This function needs to return a result set.

The analysis modules may group themselves into categories. This is used in the initial analysis creation dialog in the GUI.

Python Script A python based analysis comprises a single dedicated python script file. The InsightCAE functions are included by an initial statement like `from Insight.toolkit import *`.

This script needs to define three standalone functions:

1. `category()`

   This functions returns a plain string with the category of the analysis. Multiple hierarchy levels are supported and have to be separated by slashes.

2. `defaultParameters()`

   This function shall return an object of type `ParameterSet`, containing the default values of all the needed parameters. The user cannot add or delete parameters, just modify their values. So this defines the layout.

3. `executeAnalysis(parameters, workdir)`

   This function finally runs the simulation. It gets an object of type `ParameterSet`, containing all the input parameters and the string `workdir`, which contains the execution directory.

Upon loading, the InsightCAE base library searches in

- `$INSIGHT_GLOBALSHAREDDIRS/python_modules` and
- `$INSIGHT_USERSHAREDDIRS/python_modules`

for files named `*.py`. Each of these files will be loaded and their defined analyses will become available in the GUI and all other InsightCAE tools.

C++ In C++, a new analysis is derived from the common base class `Analysis`. The necessary functions have to be overridden. The new analysis should be put into a dedicated shared library.

Upon loading, the InsightCAE base library searches in

- `$INSIGHT_GLOBALSHAREDDIRS/modules.d` and
- `$INSIGHT_USERSHAREDDIRS/modules.d`

for files named `*.module`. Each of these files has to contain a statement `library <FILENAME>`, where FILENAME is the name of the shared library file. These libraries will all be loaded and their defined analyses will become available in the GUI and all other InsightCAE tools.

4.2 User Front Ends

4.2.1 GUI: workbench

The GUI for editing analysis parameters, run analyses and monitor their progress and to review the result sets is called `workbench`.

It can be started from the command line by executing

```
$ workbench
```

or from the global application menu.

The workbench can open and handle multiple analyses at a time. For each analysis, a sub window is opened (analysis form).

An analysis can either be created from scratch (see "Creating a New Analysis") or by opening an existing analysis input file (select Analysis » Open... in the menu).

The analysis form has three tabs. From left to right: the input parameter edit tab, the progress display tab and the result viewer tab.

Editing the Input Parameters The parameter input tab is shown in figure 5. On the left, there is a tree view showing all the available parameters of the analysis. The font display style of the parameter entry denotes the importance of the parameter:

- highlighted yellow background: these parameters have no reasonable default values and need to be changed for every case. The displayed default values are only chosen to demonstrate some functional value, e.g. for speeds or rpms. But for parameters like file names, for example, there is no reasonable default at all.
- normal black text color: these parameters have reasonable default values, with which an analysis can be started. Though it is quite normal, that these parameters might need to be adapted from case to case.
- dimmed gray text color: these parameters should normally be left at their default values. But it for rather rare special cases, it might become necessary to adapt them.

The parameters can be selected my mouse click and edited (see "Modifying a Parameter").

Some analyses (this is optional) support a 3D preview of the configured case. When an analysis is loaded, which supports 3D preview, a 3D view and a 3D element tree appears right of the parameter edit column. All views in the Input tab are shown in a horizontal splitter and their width can be adapted by dragging the splitter handle between them. It is also possible to hide the 3D preview widgets by collapsing their width to zero using the splitter handle (This might also occur accidentally). The collapsing can be undone by locating the splitter handle of the collapsed widget and dragging it back to some non-zero width. The layout of the splitter is saved on program exit and restored on the next start.

Saving Parameters Once all parameters are edited, the parameter set can be saved to a file. Very often, external files are referenced from a parameter set. Since it is also often necessary, to relocate parameter sets from one computer to another or between directories, InsightCAE parameter sets provide some special features which shall simplify relocation of parameter sets:

- file paths are always stored as relative paths inside the parameter set file.

  Even if the workbench editor displays an absolute path, the paths are made relative to the saving location of the parameter set file. It is still possible to insert absolute paths into a parameter set file. But when it is loaded and saved again, these will be made relative.

If a parameter set is moved around together with the files it references, they will be found, as long as they remain in the same relative location, e.g. side by side with the parameter file or in a sub directory.

- external files can be embedded into the parameter file.

  This is the preferred option, when parameter files shall be moved across computers. Upon saving, the files will be base-64 encoded (not compressed) and stored in the XML file. When the analysis is run, the files are extracted to a temporary location. The extraction is aware of the case, that different files with the same name might exist in different directories and extracts them to different locations. By default, the files are extracted to the system temporary location. When the InsightCAE application, which triggered the extraction of the parameter, exits, the files are deleted again from the temporary location. Note that this might not be achieved, if the application crashes.

  Packing of external files can be enabled by checking Parameters》Pack external files into parameter file or by checking the appropriate option in the "Save Parameters" dialog. Once, a parameter file was saved as packed or unpacked, the selection state is saved and used for all subsequent save operations.

Running the Analysis With a properly edited parameter set, the analysis run can be started. This is achieved by clicking on the button "Run" on the right side or by selecting Actions》Run Analysis in the menu. This analysis form switches automatically to the "Run" tab (see figure 6). In the lower half, the log message are displayed. Sometimes, errors in an external program occur and are not correctly reported. It is therefore a good practice to watch the log messages for errors or warnings.

When the simulation solver runs, InsightCAE shows progress variables graphically. The plots are shown above the log widget. The number and meaning of the progress variables depend on the analysis conducted.

Especially for OpenFOAM-bases analyses, a number of quantities are extracted from solver output and forwarded to the GUI as progress variables:

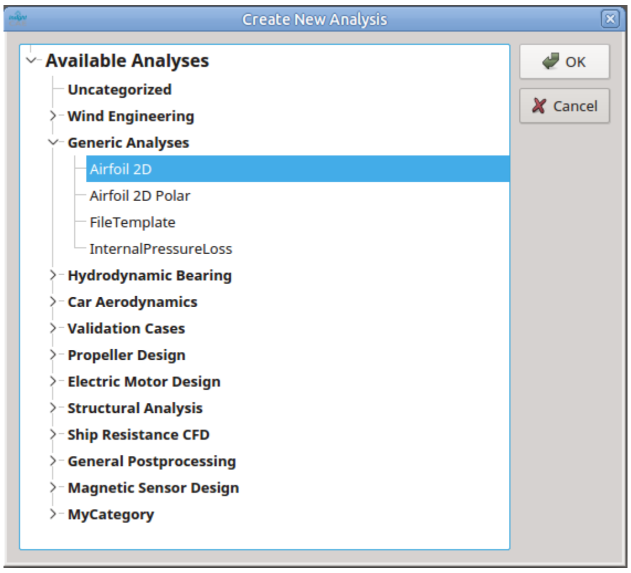
- continuity errors
- equation residuals
- forces
- moments
- execution time (per timestep)

Cancel a Simulation To cancel a simulation run, click on the button "Kill" on the right side or by select Actions》Stop Analysis in the menu.

Viewing the Results Once the simulation run is finished (that means that the solver has finished and that all post processing steps are done), a result set is created and loaded into the workbench. It is displayed in the "output" tab (see figure 7). The result set can be rendered into a report (see "Rendering Results into a Report").

Rendering Results into a Report

Modifying a Parameter



Available Analyses
- Uncategorized
- Wind Engineering
- Generic Analyses
  - Airfoil 2D
  - Airfoil 2D Polar
  - FileTemplate
  - InternalPressureLoss
- Hydrodynamic Bearing
- Car Aerodynamics
- Validation Cases
- Propeller Design
- Electric Motor Design
- Structural Analysis
- Ship Resistance CFD
- General Postprocessing
- Magnetic Sensor Design
- MyCategory

Figure 4: Dialog for selection of the type of a new analysis

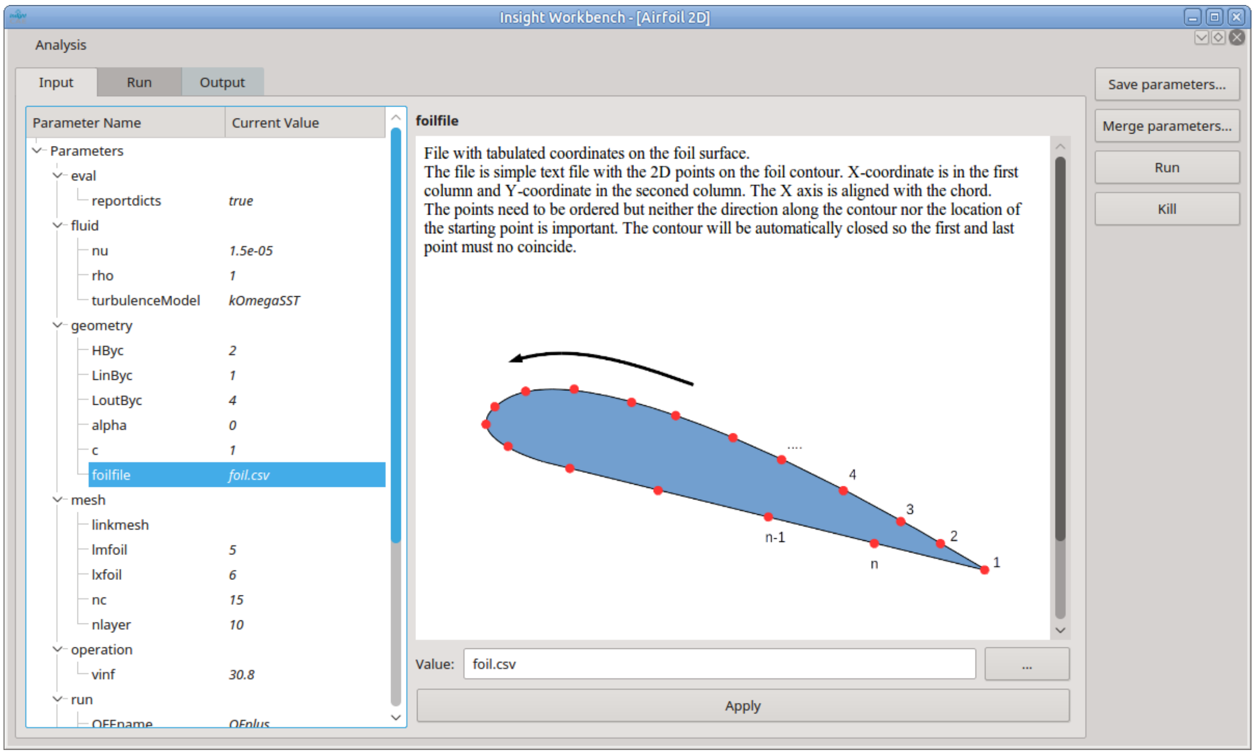
Creating a New Analysis A new analysis is created by selecting in the menu Analysis › New... . Then a new dialog appears, in which the list of available analyses is displayed (figure 4). The required analysis should be selected and confirmed by clicking "Ok".

4.2.2 Console on Headless Computers: analyze

4.3 Available Simulation Workflows

4.3.1 Numerical Wind Tunnel

4.3.2 Internal Pressure Loss



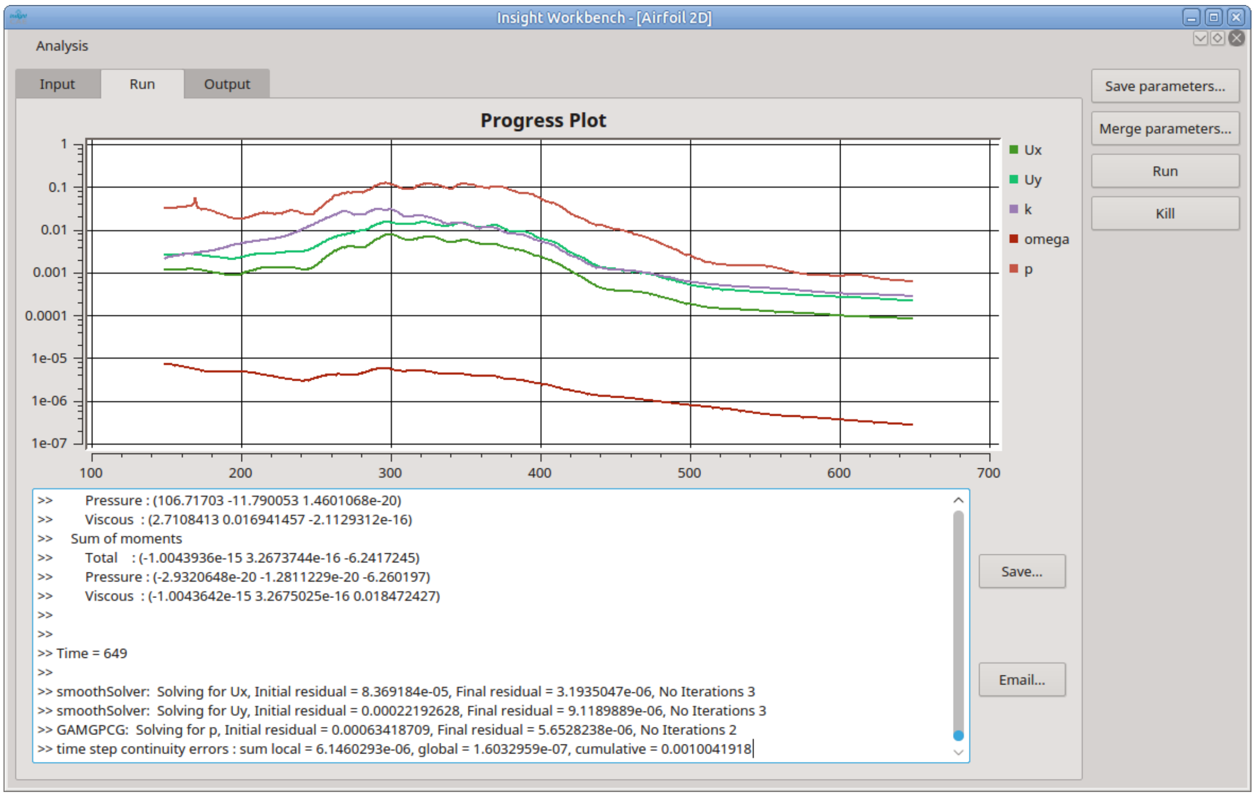
Parameter Name	Current Value
Parameters	
eval	
reportdicts	true
fluid	
nu	1.5e-05
rho	1
turbulenceModel	kOmegaSST
geometry	
HByc	2
LinByc	1
LoutByc	4
alpha	0
c	1
foilfile	foil.csv
mesh	
linkmesh	
lmfoil	5
lxfoil	6
nc	15
nlayer	10
operation	
vinf	30.8
run	
OFEname	OFplus

foilfile

File with tabulated coordinates on the foil surface.
The file is simple text file with the 2D points on the foil contour. X-coordinate is in the first column and Y-coordinate in the seconed column. The X axis is aligned with the chord.
The points need to be ordered but neither the direction along the contour nor the location of the starting point is important. The contour will be automatically closed so the first and last point must no coincide.

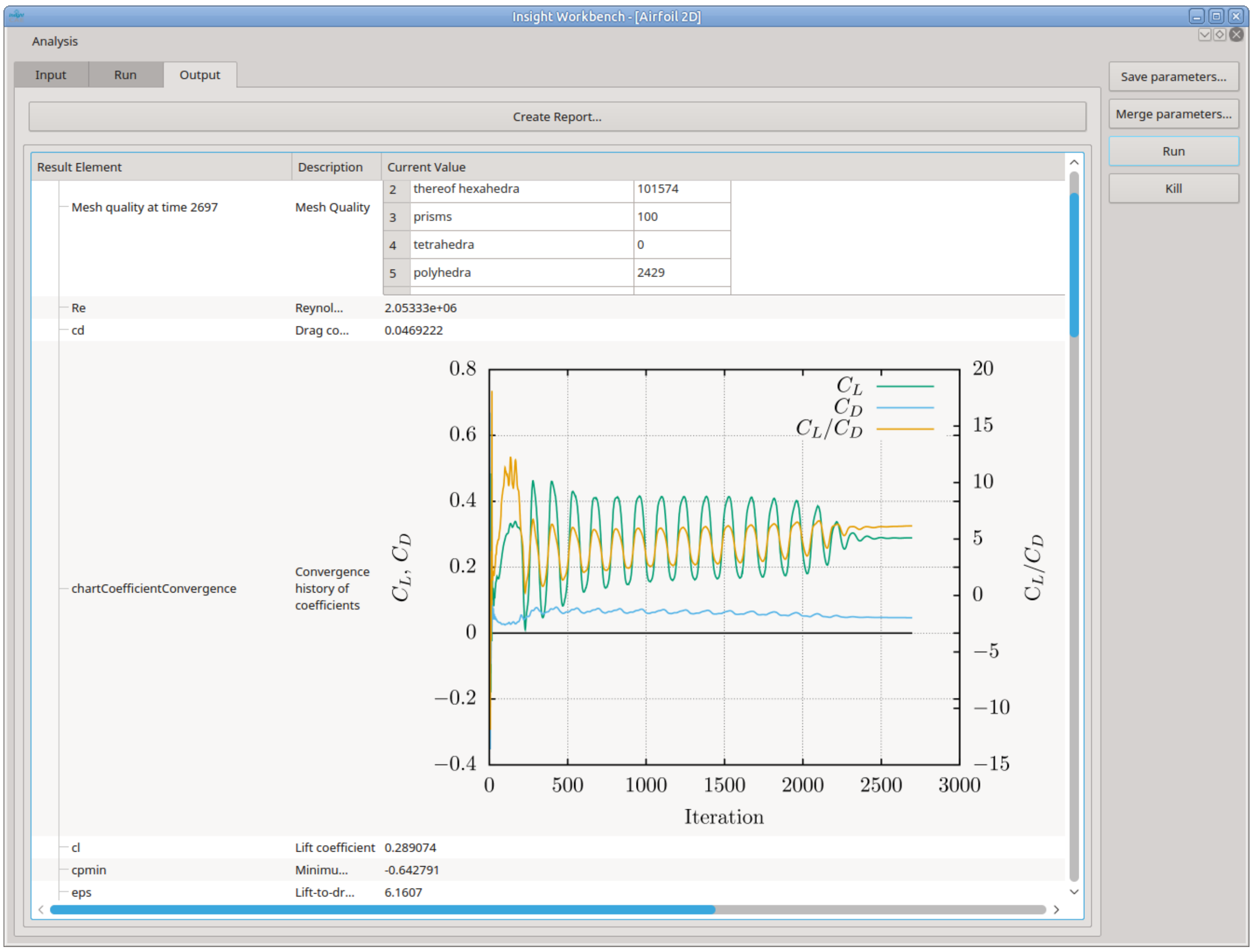
Value: foil.csv

Figure 5: Parameter editing tab of the workbench



```
>>      Pressure : (106.71703 -11.790053 1.4601068e-20)
>>      Viscous  : (2.7108413 0.016941457 -2.1129312e-16)
>>    Sum of moments
>>      Total    : (-1.0043936e-15 3.2673744e-16 -6.2417245)
>>      Pressure : (-2.9320648e-20 -1.2811229e-20 -6.260197)
>>      Viscous  : (-1.0043642e-15 3.2675025e-16 0.018472427)
>>
>>
>> Time = 649
>>
>> smoothSolver:  Solving for Ux, Initial residual = 8.369184e-05, Final residual = 3.1935047e-06, No Iterations 3
>> smoothSolver:  Solving for Uy, Initial residual = 0.00022192628, Final residual = 9.1189889e-06, No Iterations 3
>> GAMGPCG:  Solving for p, Initial residual = 0.00063418709, Final residual = 5.6528238e-06, No Iterations 2
>> time step continuity errors : sum local = 6.1460293e-06, global = 1.6032959e-07, cumulative = 0.0010041918
```

Figure 6: Solution progress tab of the workbench



Result Element	Description	Current Value
Mesh quality at time 2697	Mesh Quality	2 thereof hexahedra 101574 3 prisms 100 4 tetrahedra 0 5 polyhedra 2429
Re	Reynol...	2.05333e+06
cd	Drag co...	0.0469222
chartCoefficientConvergence	Convergence history of coefficients	
cl	Lift coefficient	0.289074
cpmin	Minimu...	-0.642791
eps	Lift-to-dr...	6.1607

Figure 7: Result explorer tab of the workbench

5 Supporting Simulations with OpenFOAM

5.1 Case Setup GUI: `isofCaseBuilder`

5.2 Execution Monitor GUI: `isofExecutionManager`

5.3 Tabular Output Data Plot: `isofPlotTabular`

6 Extensions to Back-End-Tools

6.1 OpenFOAM

7 CAD

7.1 Introduction

InsightCAD is a script-based tool for creating three-dimensional geometry models. All geometric operations are based on the OpenCASCADE geometry kernel. It uses the boundary representation approach (BREP) for treating the geometry. Thus it is possible to import and export the common exchange formats IGES and STEP.

Although the primary intention of InsightCAD is creation of fully parameterized CAD models for systematic numerical simulations, it can also be used for mechanical design purposes. It therefore provides the possibility to export projections and sections of the created models in DXF format for use in drawings. Additionally, there is support for a library of parametric standard parts.

7.2 Basic Concept

The basic entity in InsightCAD is a ”model”. A model is described by a script and stored in an ASCII script file (extension ”.iscad”). Inside a model script, symbols are defined, which can represent the following data types:

- Scalars
- Vectors
- Datum objects (axes, planes)
- 3D geometry objects (features)
- Selections of vertices, edges, faces or solids of 3D geometry objects

There is no explicit type declaration: the data type of each symbol is deduced from the defining expression.

Beyond their geometry representation, geometry feature objects are containers for scalar, vector, datum and feature objects. These symbols can be accessed in the model script but are read-only.

It is possible to load another model into the current one by the ”loadmodel” command (see section ”Features”). In this case, the loaded model represents a subassembly, i.e. a compound of features. Since not all defined geometry objects in the submodel may represent assembly components, marking of components is supported by the feature definition syntax and needs to be utilized properly in the definition script of the submodel. Also, when loading models (subassemblies), parameters can be passed to the submodel. These can be scalars, vectors, datums or features.

In a model script, after the definition of the aforementioned symbols, an optional section with postprocessing actions can follow. These can be e.g. file export, drawing export or others.

7.3 Syntax

The general model script layout begins with a mandatory symbol defining section, followed by an optional postprocessing section, started by "@post":

```
<identifier> [ = | ?= | : ] <expression>;
...

@post

<postprocessing action>;
...
```

Comments are lines starting with "#" or text regions enclosed by "/*" and "*/" (C-style).

7.4 Symbol Definition

7.4.1 Scalars

Example:

```
D ?= 123.5;
L = 2.*D;
```

The "?=" operator assigns a default value. This needs to be used for symbols which are intended to be used as parameters in the "loadmodel" feature command. Symbols defined by the equal sign operator ("=") cannot be overridden during the loadmodel command.

Supported operations are listed in table 2.

InsightCAD script	Description
+ - * /	basic algebra
mag(<vector >)	magnitude of vector
sqrt(<scalar >)	square root
sin(<scalar >)	sin(x)
cos(<scalar >)	cos(x)
tan(<scalar >)	tan(x)
asin(<scalar >)	arcsin(x)
acos(<scalar >)	arccos(x)
ceil(<scalar >)	smallest following integer
floor(<scalar >)	largest previous integer
round(<scalar >)	round to next integer
pow(<scalar:a >, <scalar:n >)	aˆn
atan2(<scalar:y >, <scalar:x >)	arctan(y/x)
atan(<scalar >)	arctan(x)
volume(<feature >)	volume of feature
cumedgelen(<feature >)	sum of all edges lengths
<vector >.x	x component of vector
<vector >.y	y component of vector
<vector >.z	z component of vector
<feature > $ <identifier >	scalar property of feature
<vector > & <vector >	Scalar product

Table 2: Algebraic operations in ISCAD scripts

There are scalars predefined in each model. They are listed in table 3.

InsightCAD script	Description
M_PI	pi
deg	180/pi

Table 3: Predefined scalar symbols in ISCAD scripts

7.4.2 Vectors

Example:

```
v ?= 5*EX + 3*EY + [0,0,1];
ev = v/mag(v);
```

The "?=" operator assigns a default value. This needs to be used for symbols which are intended to be used as parameters in the "loadmodel" feature command. Symbols defined by the equal sign operator ("=") cannot be overridden during the loadmodel command.

Supported operations are listed in table 4.

InsightCAD script	Description
+ -	basic algebra
[<x >, <y >, <z >]	vector from components
<feature >@<vector >	vector property of feature
<scalar >*<vector >	scaled vector
<vector >/<scalar >	scaled vector
<vector:a >ˆ<vector:b >	a x b
bbmin(<feature >)	minimum corner of feature bounding box
bbmax(<feature >)	maximum corner of feature bounding box
cog(<feature >)	center of gravity coordinates of feature
refpt(<datum >)	reference point of datum (base point of axis or plane)
refdir(<datum >)	reference direction of datum (direction of axis or normal of plane)

Table 4: Vector operations and functions in ISCAD scripts

There are vectors predefined in each model. They are listed in table 5.

InsightCAD script	Description
EX	$\vec{e}_x = (100)^T$
EY	$\vec{e}_y = (010)^T$
EZ	$\vec{e}_z = (001)^T$
O	$\vec{O} = (000)^T$

Table 5: Vector operations and functions in ISCAD scripts

7.4.3 Datums

Some simple examples are given below.

```
myaxis ?= RefAxis(O, EX+EY); # diagonal axis
myplane = Plane(5*EX, EY); # offset plane
```

```
myplane2 = XZ << 5*EX; # same offset plane
axis2 = xsec_plpl(XY, myplane); # axis at intersection of XY-Plane and offset
    plane
```

The "?=" operator assigns a default value. This needs to be used for symbols which are intended to be used as parameters in the "loadmodel" feature command. Symbols defined by the equal sign operator ("=") cannot be overridden during the loadmodel command.

Supported operations are listed in table 6.

InsightCAD script	Description
<feature >%<identifier >	Access datum inside another feature.
<datum > <<<vector >	Copy of datum, translated by $\vec{\Delta}$
Plane(<vector:p0 >, <vector:n >)	Datum plane with origin p0 and normal n.
SPlane(<vector:p0 >, <vector:n >, <vector:e_up >)	Additionally, the y-direction of the plane CS is aligned with $\vec{e}_{up}$.
RefAxis(<vector:p0 >, <vector:e_x >)	Axis with origin p0 and direction $\vec{e}_x$.
xsec_axpl(<datum:ax >, <datum:pl >)	Datum point at intersection between axis ax and plane pl
xsec_plpl(<datum:pl1 >, <datum:pl2 >)	Datum axis at intersection between plane pl1 and pl2
xsec_ppp(<datum:pl1 >, <datum:pl2 >, <datum:pl3 >)	Datum point at intersection between three planes

Table 6: Vector operations and functions in ISCAD scripts

There are datums predefined in each model. They are listed in table 8.

InsightCAD script	Description
XY	X-Y-Plane
XZ	X-Z-Plane
YZ	Y-Z-Plane

Table 7: Vector operations and functions in ISCAD scripts

7.4.4 Features

A very simple example is given below. It consists of two primitive features (cylinder) and a boolean operation (subtraction).

```
tool = Cylinder(-10*EY, 10*EY, 2);

pierced_cylinder:
Cylinder(0, 20*EX, 10, centered)
-
tool;
```

Geometry symbols can be defined by a "=" or a ":" operator. The difference comes from a possible use of the model as a subassembly later on. Since usually not all defined features in a model are assembly components but some are only intermediate modeling steps, there are these two syntaxes for defining a feature with a subtle difference: "<identifier > = <expression >;" defines an intermediate feature while "<identifier >: <expression >;" does the same geometry operation but marks the result as being an assembly component. In the above example, only the feature "pierced_cylinder" is marked as a component. Thus if the above example would be loaded as a subassembly, only the "pierced_cylinder" will be shown and included in e.g. mass calculations.

InsightCAD script	Description
<feature:a > - <feature:b >	Boolean subtract of feature b from feature a
<feature:a > | <feature:b >	Boolean unite of feature a and feature b
<feature:a > & <feature:b >	Boolean intersection of feature a and feature b
<feature > <<<vector:delta >	Copy of feature, translated by vector delta
<feature > * <scalar:s >	Copy of feature, scaled by s (relative to global origin O)
<feature >.<identifier:subfeatname >	Access of subfeature

Table 8: Vector operations and functions in ISCAD scripts

7.5 Feature Commands

In this section, an incomplete subset of the avilable feature commands is described in detail. For a complete list, please refer to the online documentation in the iscad editor (press Ctrl+F).

7.5.1 Transformation

`Transform(<feature:f >, <vector:delta >, <vector:phi >)`

Transformation of feature f. Translation by vector delta and rotation around axis vector phi (magnitude of phi gives rotation angle).

`Place(<feature:f >, <vector:p_0 >, <vector:e_x >, <vector:e_z >)`

Places the feature f in a new coordinate system. The new origin is at point $\vec{p}_0$, the new x-axis along vector $\vec{e}_x$ and the new z-direction is $\vec{e}_z$.

7.5.2 Import

`import(<path >)`

Imports solid geometry from a file. The format is recognized from the filename extension. Supported formats are IGS, STP, BREP.

`Sketch(<datum:pl >, <path:file >, <string:name > [, <identifier >=<scalar >, ... ])`

Reads a sketch (i.e. a closed contur) from a file. The geometry in the sketch is expected to be in the X-Y-Plane. It is placed on the given plane pl. Sketch file format is recognized from the file name extension. Supported are ".dxf" and ".fcstd" (FreeCAD). The name is interpreted as layer name in DXF and sketch name in FreeCAD files.

For FreeCAD sketches, a list of parameter values can optionally be supplied. Upon loading, the sketch will be regenerated through FreeCAD with these values.

`loadmodel( <identifier:modelname > [, <identifier > = <feature >|<datum >| <vector >|<scalar >, ... ] )`

Parses another InsightCAD model (submodel) and inserts a compound of all features into the current model, which were marked as components in the submodel (i.e. which were defined with the colon ":" operator instead of the equal sign "=" in the submodel).

The model filename has to be "<modelname >.iscad". It is searched for in the following directories:

1. the directories listed in the environment variable "ISCAD_MODEL_PATH" (separated by ":")
2. the subdirectory "iscad-library" in InsightCAEs shared file directory
3. in the current directory

Optionally, a list of symbols is inserted into the namespace of the submodel (additional optional parameters).

7.5.3 Geometry Construction

Primitives There are commands for creation of several different geometrical primitives, e.g.

- 1D: Arc, Line, SplineCurve
- 2D: Quad, Tri (triangle), RegPoly (regular polygon), Circle, SplineSurface
- 3D: Sphere, Cylinder, Box, Bar, Cone, Pyramid, Torus

`Extrusion(<feature:f >, <vector:L > [, centered ] )`
Extrude the feature f with direction and length vector L. When the keyword "centered" is given, the extrusion is centered around f.
`Revolution( <feature:f >, <vector:p_0 >, <vector:axis >, <scalar:phi > [, centered] )`
Creates a revolution of the planar feature f. The rotation axis is specified by origin point $\vec{p}_0$ and the direction vector axis. Revolution angle is specified by phi. By giving the keyword "centered", the revolution is created symmetrically around the base feature.

Other Feature Commands There are more features available. A comprehensive list is obtained by pressing Ctrl+F in the iscad editor.

7.6 Lower Dimensional Shape Selection

ISCAD supports rule based selection of lower dimensional features (i.e. edges or faces of a solid). The selection is generated by a selection command: a question mark, followed by the type of shape to query. The result is a selection object:

```
<feature expression|feature selection>?(vertices|edges|faces|solids);('<command␣
    string>' [, parameter 0 [, ..., parameter n] ] )
```

It is possible to supply additional arguments to the selection expression, like scalars, vectors or features.
An example: the following expression selects the circumferential face of the cylinder c (all faces, which are not plane) and stores the selection in "shell_faces":

```
c = Cylinder(O, 5*EZ);
shell_face = c ? faces('!isPlane');
min_end_face = c ? faces('isPlane␣&&␣minimal(CoG.z)');
```

The selection command string contains rules for the selection. Finally, the command string is evaluated as a boolean expression comprising comparison operators, boolean operators and query functions. Within these boolean expressions, quantity functions can be used. The available set of query functions and quantity functions depends on the type of shape which shall be queried.
General functions available for all kinds of lower dimensional shapes (Q: returns quantity, B: returns boolean):
Specialized functions:

Command	Description
==, <, >, >=, <=	value comparison
!	not
&&	and
||	or
<value 1 > ˜<value 2 > { <tolerance > }	approximate equality
in(<selection set >)	true if shape is in other selection
maximal(<quantity >)	true for the shape with maximum quantity
minimal(<quantity >)	true for the shape with minimum quantity

Table 9: General boolean functions and operators available for all kinds of lower dimensional shape

Command	Description
angleMag(<vec 1 >, <vec 2 >)	angle between vec 1 and vec 2
angle(<vec 1 >, <vec 2 >)	angle between vec 1 and vec 2
%d<index >	parameter <index > as scalar
%m<index >	parameter <index > as vector
%<index >	parameter <index > as selection set

Table 10: General quantity functions available for all kinds of lower dimensional shape

7.6.1 Vertices

Command	Description
loc	location of the vertex

Table 11: Quantity functions available for vertex selections

7.6.2 Edges

Command	Description
isLine	true, if edge is straight
isCircle	true, if edge is circular
isEllipse	true, if edge is elliptical
isHyperbola	true, if edge is on a hyperbola
isParabola	true, if edge is on a parabola
isBezierCurve	true, if edge is a bezier curve
isBSplineCurve	true, if edge is a BSpline curve
isOtherCurve	true, if edge is none of the above
isFaceBoundary	true, if edge is boundary of some face
boundaryOfFace(<set >)	true, if edge is boundary of one of the faces in set
isPartOfSolid(<set >)	true, if edge is part of one of the solids in set
isCoincident(<set >)	true, if edge is coincident with one of the edges in set
isIdentical(<set >)	true, if edge is identical with one of the edges in set
projectionIsCoincident(<set >, <vec:p0 >, <vec:n >, <vec:up >, <scalar:tol >)	true, if projection of edge is coincident with some edge in set

Table 12: Boolean functions available for edge selections

Command	Description
len	length of edge
radialLen(<vec:ax >, <vec:p0 >)	
	radial distance between ends with respect to axis (p0,ax)
CoG	center of gravity of edge
start	start point coordinates
end	end point coordinates

Table 13: Quantity functions available for edge selections

7.6.3 Faces

Command	Description
isPlane	true, if is a plane
isCylinder	true, if is a cylindrical surface
isCone	true, if is a conical surface
isSphere	true, if is a spherical surface
isTorus	true, if is a toroidal surface
isBezierSurface	true, if is a bezier surface
isBSplineSurface	true, if is a BSpline surface
isSurfaceOfRevolution	true, if is a surface of revolution
isSurfaceOfExtrusion	true, if is a surface of extrusion
isOffsetSurface	true, if is a offset surface
isOtherSurface	true, if is some other kind of surface
isPartOfSolid(<set >)	
isCoincident(<set >)	
isIdentical(<set >)	
adjacentToEdges(<set >)	
adjacentToFaces(<set >)	

Table 14: General boolean functions available for face selections

Command	Description
area	area of the face
CoG	center of gravity of the face
cylRadius	radius of a cylindrical face
cylAxis	axis direction of a cylindrical face

Table 15: Quantity functions available for face selections

7.6.4 Solids

7.7 Postprocessing Actions

DXF(<path:outputfile >) << <feature:f > <view_definition > [, <view_definition >, ...]
The DXF postprocessing action creates a DXF file for further use in drawings. Several views are derived from the feature f. A <view_definition > takes the following form:

Command	Description
CoG	center of gravity
volume	volume of the solid

Table 16: Quantity functions available for solid selections

```
<identifier:viewname> (
 <vector:p_0>, <vector:n>, up <vector:e_up>
  [, section]
  [, poly]
  [, skiphl]
  [, add [l] [r] [t] [b] [k] ]
  )
```

It defines a view on the point vector $\vec{p}_0$ with normal direction vector $\vec{n}$ of the view plane. The upward direction (Y-direction) is aligned with vector $\vec{e}_{up}$.

The keyword "section" toggles whether only the outline is projected or if the view plane creates a section through the geometry.

If keyword "poly" is given, the DXF geometry will be discretized. This is more robust but creates much larger DXF files.

Keyword "skiphl" toggles whether hidden lines are output.

The keyword "add" followed by the key letters l, r, t, b and/or k enables creation of additional projections from the left, right, top, bottom and/or back, respectively.

An example is given below:

```
c: Cylinder(O, 100*EX, 20);

@post

DXF("c.dxf") << c
    top ( O, EX, up EY )
    front ( O, EZ, up EY )
;
```

gmsh(<path:outputfile >) << <feature:f > as <identifier:l > <mesh_parameters > Generates a (triangular or tetrahedral) mesh for an FEA analysis of the feature f. Gmsh is used as a meshing backend. The mesh format is determined by the outputfile extension (.med = MED format). The label of the mesh is set to l.

The syntax of the <mesh_parameters > are as follows:

```
L = ( <scalar:Lmax> <scalar:Lmin> )
 [linear]
 vertexGroups( [ <identifier:group name> = <vertex set> [ @ <scalar:size> ], ...
    ] )
 edgeGroups( [ <identifier:group name> = <edge set> [ @ <scalar:size> ], ... ] )
 faceGroups( [ <identifier:group name> = <face set> [ @ <scalar:size> ], ... ] )
 [ vertices ( [ <identifier:vertex name> = <vector:location> ], ... ) ]
```

The general mesh size is set by Lmax and Lmin. The optional keyword "linear" switch from quadratic to linear elements. Named groups of vertices, edges and faces can be created using the keywords "vertexGroups", "edgeGroups" and "faceGroups" repectively. Each group definition takes the form ”groupname” = ”selection set” (see section ”Lower dimensional shape selection” for definition of selection sets). Optionally, a mesh size can be assigned to each defined group by appending an @ sign followed by a scalar value.

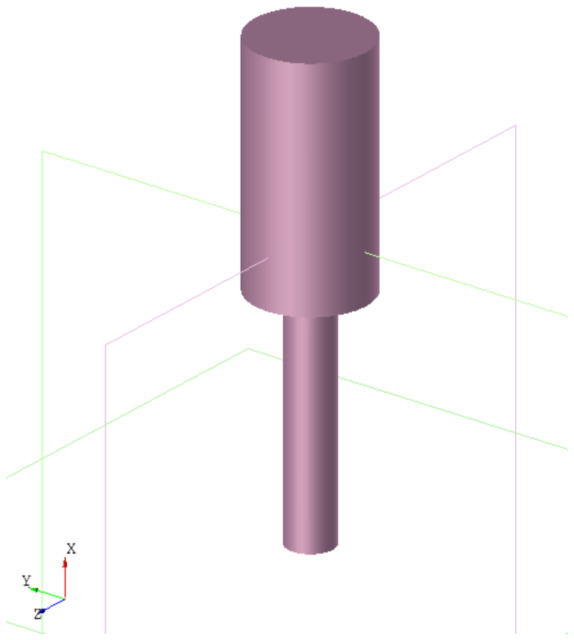
An example is given below:

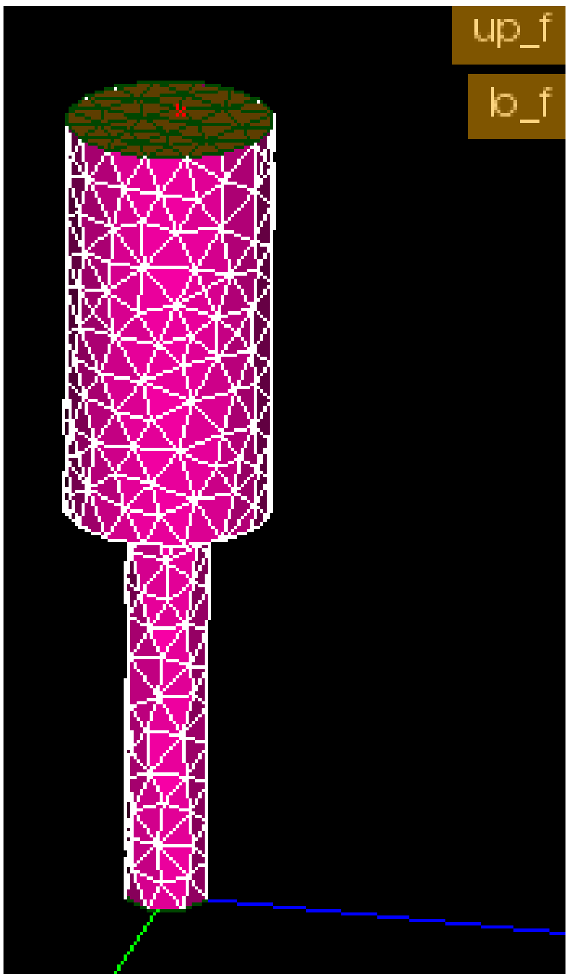
```
c:
Cylinder(O, 100*EX, 20)
 |
Cylinder(100*EX, ax 100*EX, 50)
;

@post

gmsh("c.med") << c as cyl
     L = (10 0.1)
     linear
     vertexGroups()
     edgeGroups()
     faceGroups(
      lo_f = c?faces('isPlane&&minimal(CoG.x)')
      up_f = c?faces('isPlane&&maximal(CoG.x)')
     )
;
```

Result: ISCAD model (left) and resulting mesh (right):





7.8 Graphical Editor ISCAD

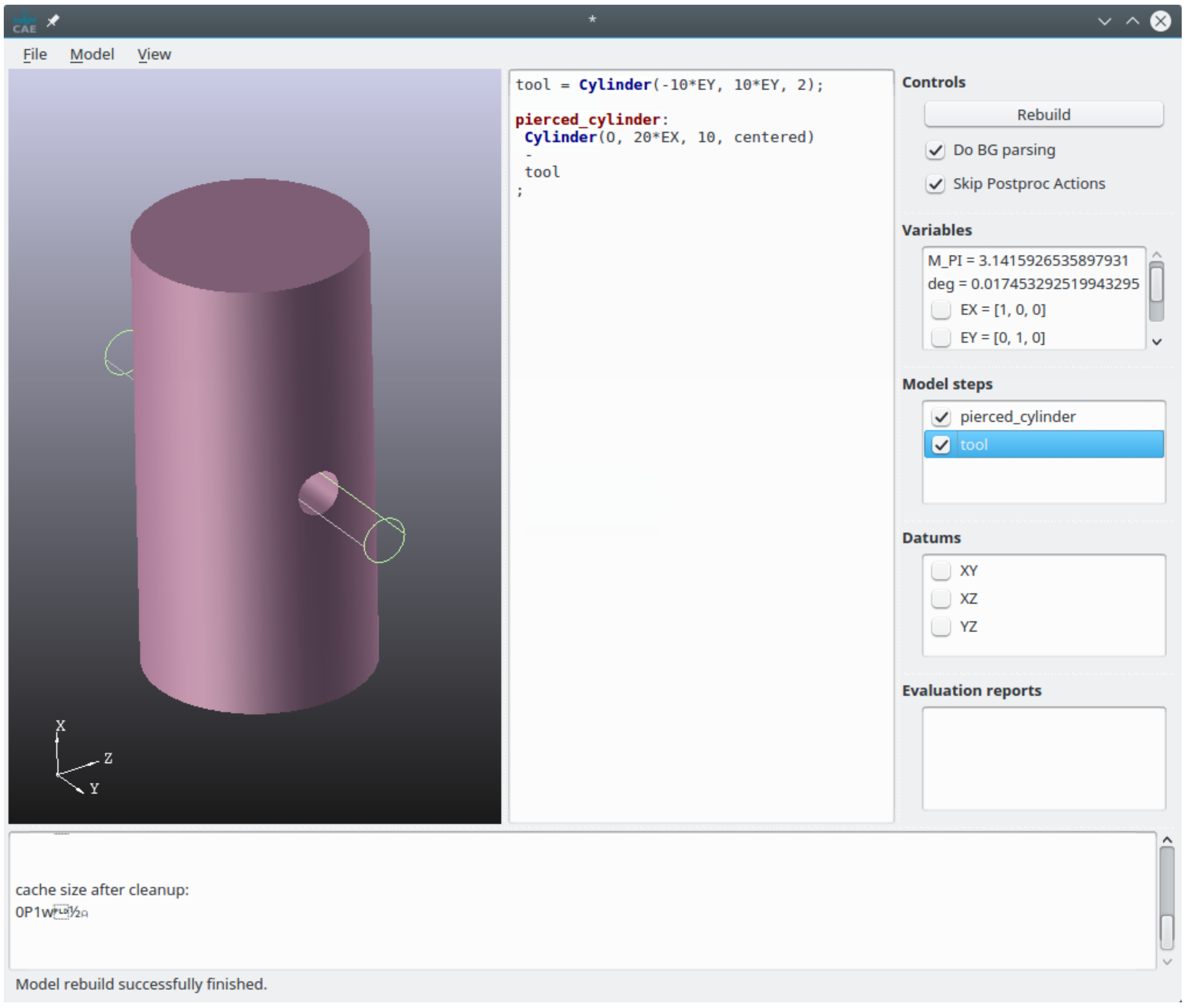
ISCAD is a graphical editor for InsightCAD scripts. It consists of a text editor for editing the script contents and a 3D view and some other elements for inspecting the resulting model. A screenshot is shown in the figure below.

The model script is entered into the text editor widget right of the 3D display. Once a script shall be evaluated, it can be parsed by clicking in the button ”Rebuild” or pressing Ctrl+Return.

After parsing the model, the following results are displayed:

- A list of all created feature symbols in the ”Model Steps” list box. If the check box is checked, the 3D geometry is displayed in the 3D view.
- The values of all scalars and vectors in the ”Variables” list box.

  For each vector variable, a check box is displayed. If it is checked, the vector is interpreted as a point location and the point is shown in the 3D display window.
- All defined datums are listed in the ”Datums” list box. Again, the check box controls, whether the datum is displayed in the 3D view.



```
tool = Cylinder(-10*EY, 10*EY, 2);

pierced_cylinder:
 Cylinder(0, 20*EX, 10, centered)
 -
 tool
;
```

✓ Do BG parsing

✓ Skip Postproc Actions

```
M_PI = 3.1415926535897931
deg = 0.017453292519943295
EX = [1, 0, 0]
EY = [0, 1, 0]
```

✓ pierced_cylinder

✓ tool

XY

XZ

YZ

```
cache size after cleanup:
0P1w[illegible]½
```

Model rebuild successfully finished.

7.8.1 3D Graphics Display

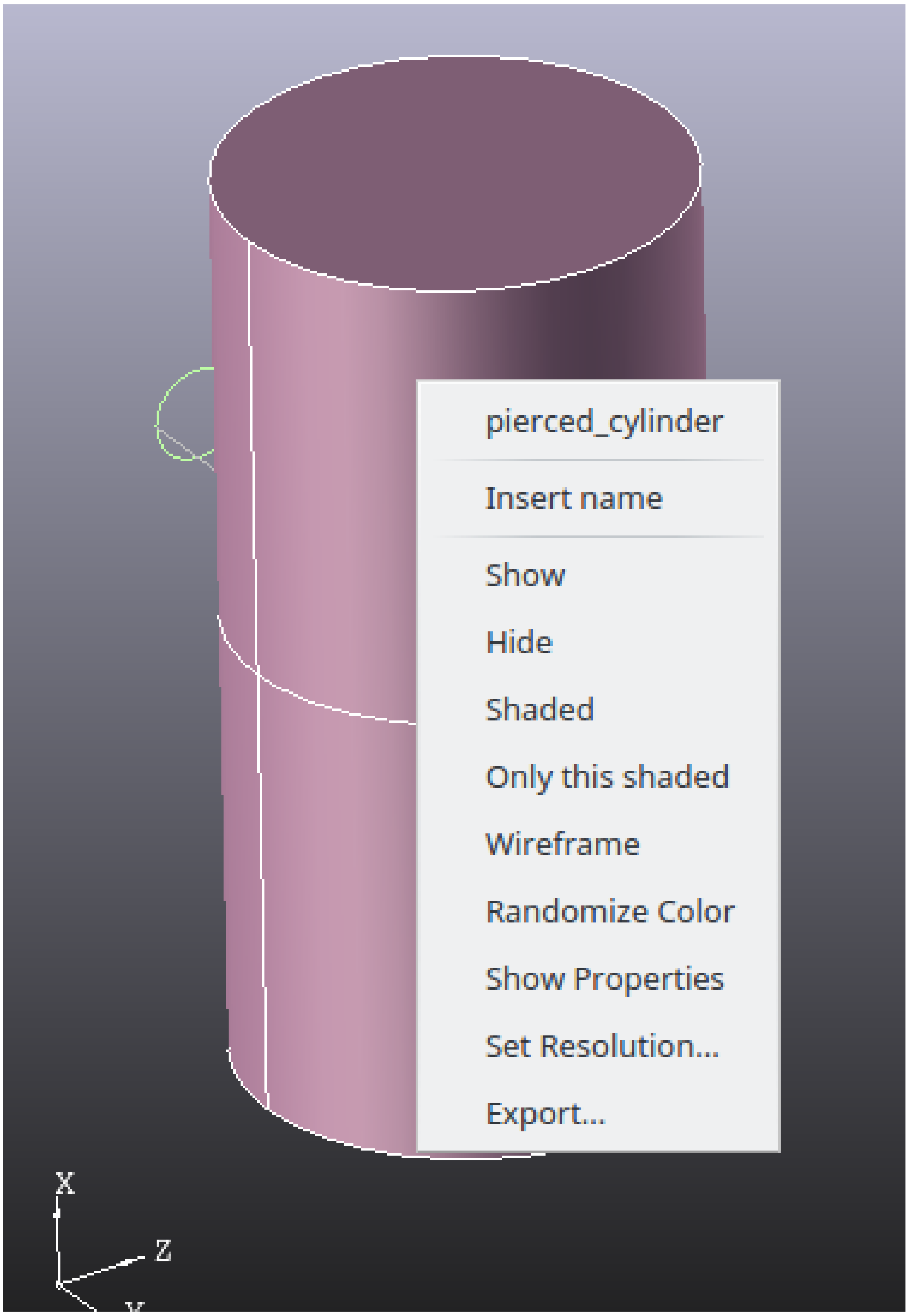
View Manipulation

- Dragging: Shift + mouse move
- Scaling: Ctrl + horizontal mouse move
- Rotating: Alt + mouse move

Model Navigation When hoovering the mouse pointer over a displayed feature in the 3D geometry window, it is highlighted. The highlighted feature is the one, which would be selected during subsequent mouse clicks.

When the left mouse button is pressed, the highlighted feature is selected and all its contained reference points are displayed.

When the right mouse button is pressed, a context menu for the selected feature is displayed.


pierced_cylinder
Insert name
Show
Hide
Shaded
Only this shaded
Wireframe
Randomize Color
Show Properties
Set Resolution...
Export...
X
Z
Y

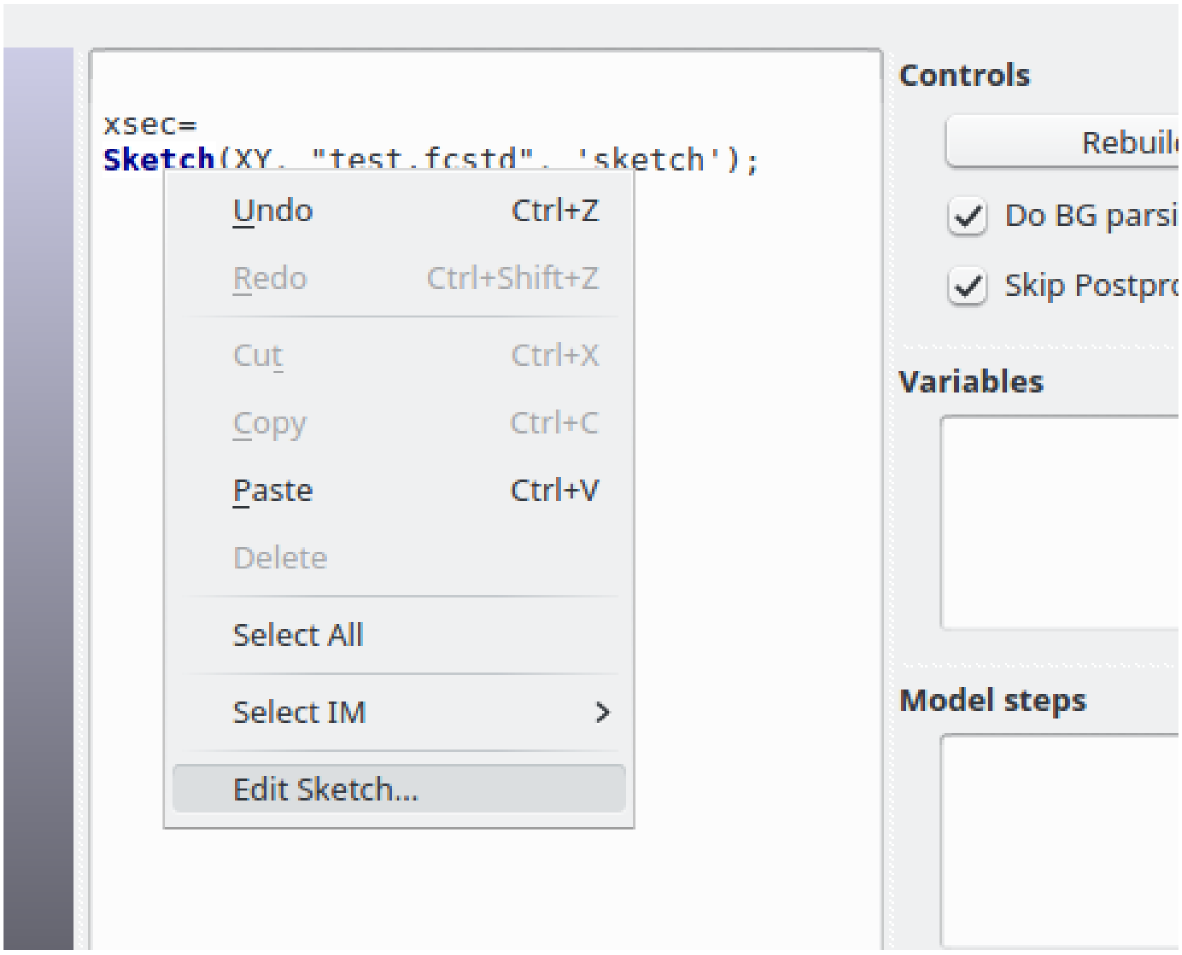
The context menu provides these functions:

- The first entry is the feature name. When it is selected, the definition in the script editor is highlighted and the cursor jumps to it.
- ”Insert name”: inserts the name of the feature symbol at the cursor location
- ”Export...”: Export the feature geometry to a file (BREP, STP, IGES, STL)

7.8.2 Text Editor

When script code is entered into the text editor window, it is parsed in the background. Once this has been done successfully, some extensions of the context menu is available:

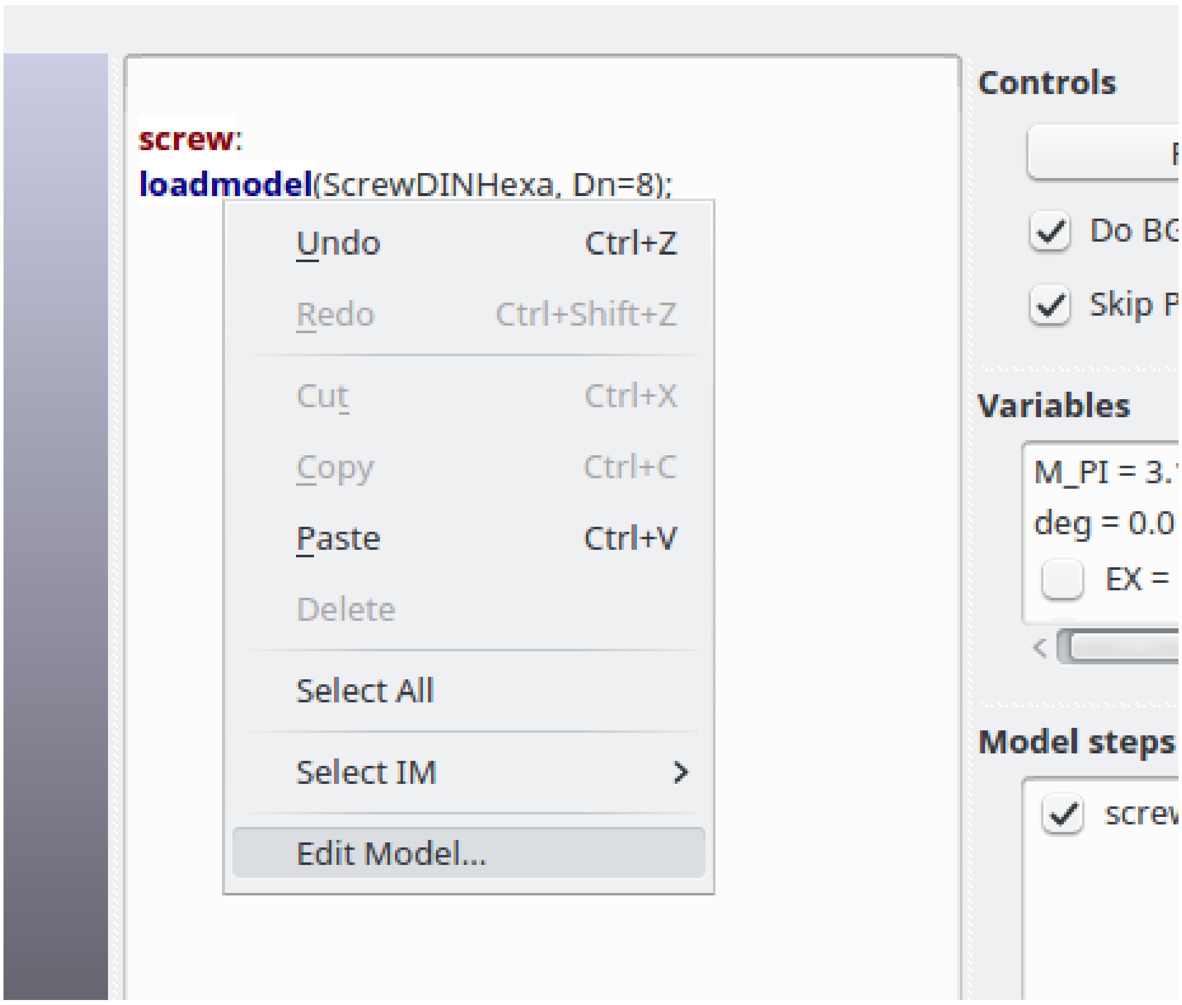
- Context menu on ”Sketch” command:



```
xsec=
Sketch(XY, "test.fcstd", 'sketch');
```

When selecting the ”Edit Sketch...” entry, FreeCAD is launched and the sketch editor opened. If the FreeCAD-file or sketch inside it is not yet existing, they are created.

- Context menu on ”loadmodel” command:



```
screw:
loadmodel(ScrewDINHexa, Dn=8);
```

When selecting the "Edit Model..." entry, another instance of iscad is launched with the specified model script loaded.

8 Developer Documentation

9 Getting Started, Tutorials