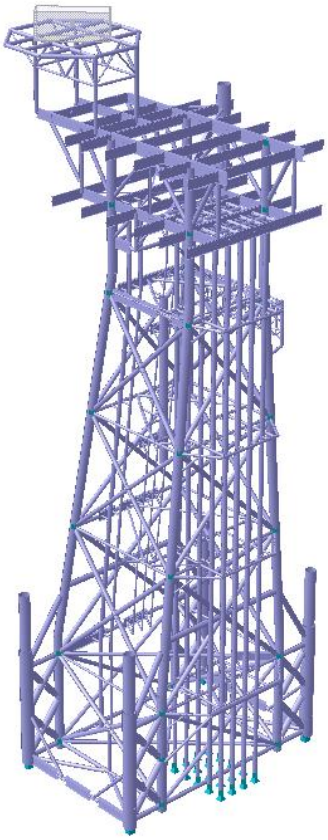
SESAM™
USER MANUAL

GENIE VOL. VI
MENU DESCRIPTION



Concept design and analysis
of offshore & maritime structures

DET NORSKE VERITAS

This page is blank for printing

Sesam™
User Manual
GeniE Vol. VI

Menu Description

Concept design and analysis
of offshore structures

11 April 2011

Valid from program version 5.3

Developed and Marketed by
DET NORSKE VERITAS

DNV Software Report No.: 00-000 / Revision 0, 11 April 2011

Published by:

Det Norske Veritas Software
Veritasveien 1
N-1322 HØVIK
Norway

Telephone: +47 67 57 99 00

Facsimile: +47 67 57 72 72

E-mail, sales: software.sesam@dnv.com

E-mail, support: software.support@dnv.com

Website: www.dnvsoftware.com

GeniE User Manual
Menu Description
Version 5.3

Table of Contents

1. INTRODUCTION

The purpose of this document is to list the available commands in GeniE. There are four alternative ways of accessing a command:

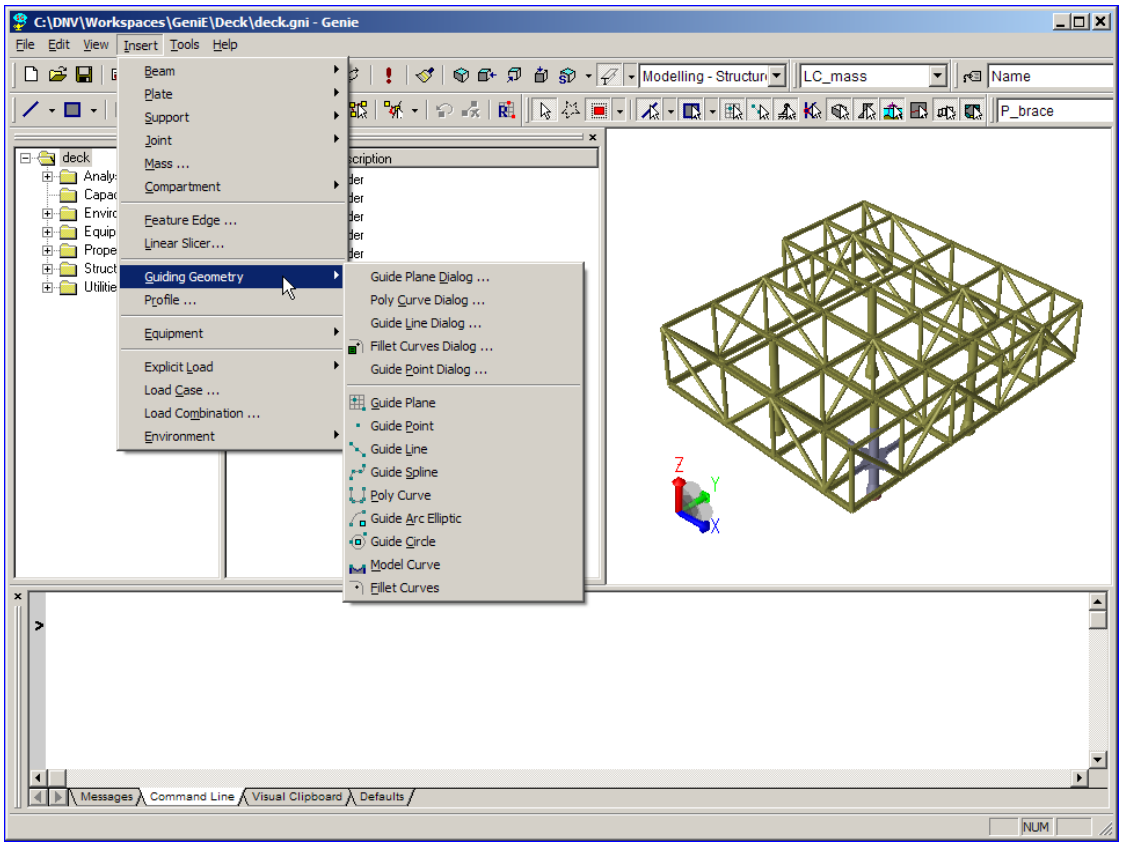
- From the pulldown menu
- From the tool button menu
- From the context sensitive menu when selecting an object(s) in the graphical window or from the browser
- From the command line interface window using the script language

The commands are listed in the following. For practical usage of the commands, reference is made to User Manuals Volume 1 through 5 as well as the tutorials found under GeniE's help site. You should have access to all the program extensions of GeniE to see all commands.

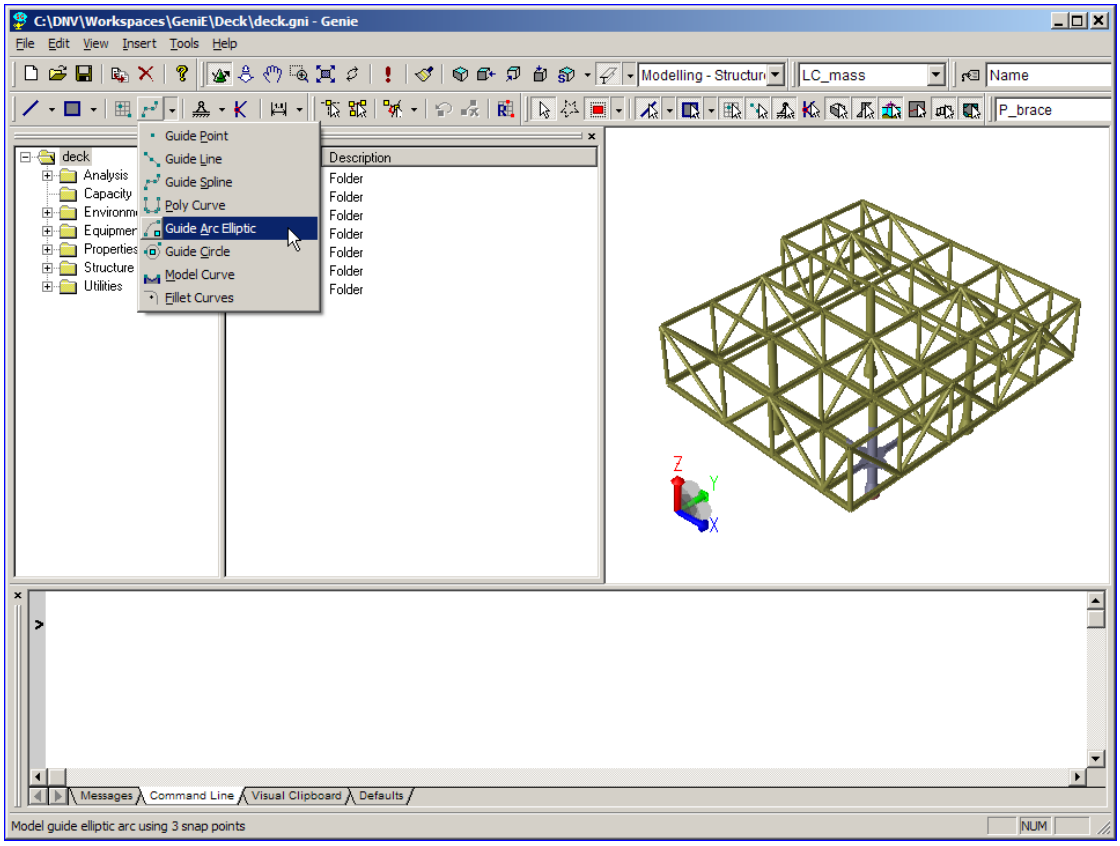
The script language is documented on GeniE's help site. For some of the commands, the relevant script command is given.

The program defaults for relevant menus are also shown.

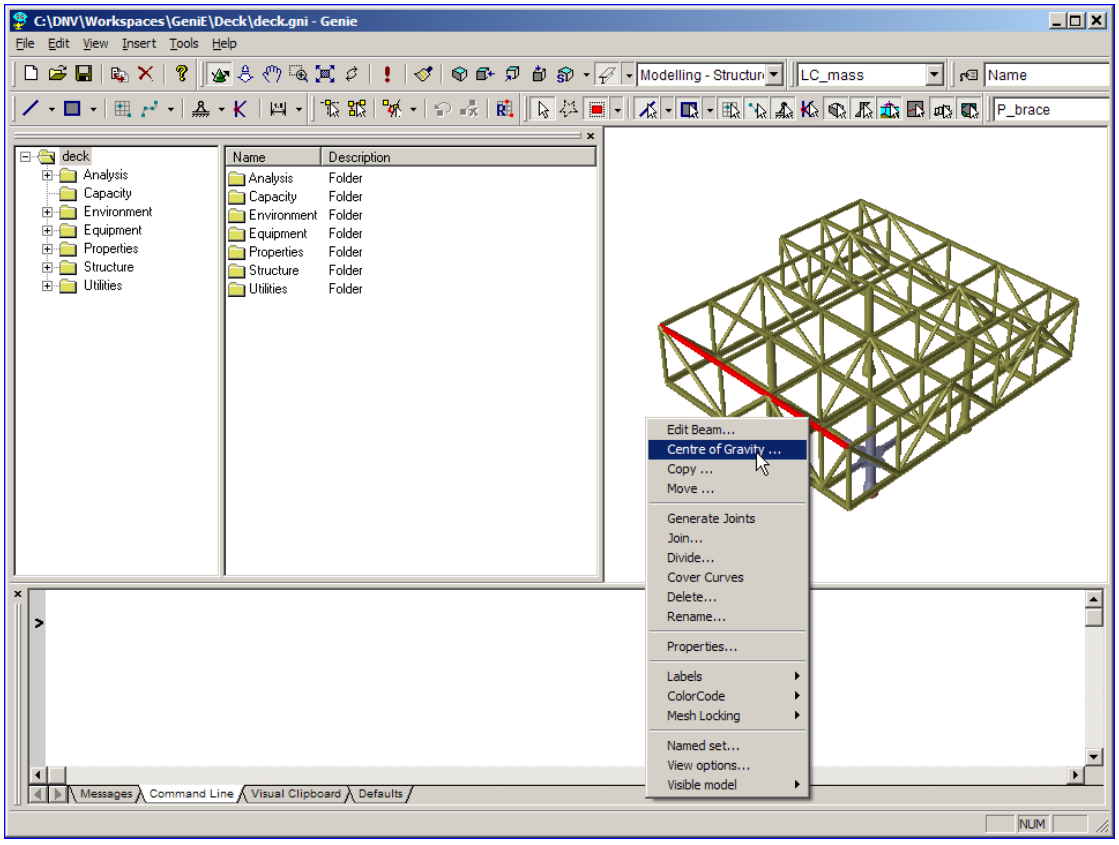
Example on a pulldown menu:



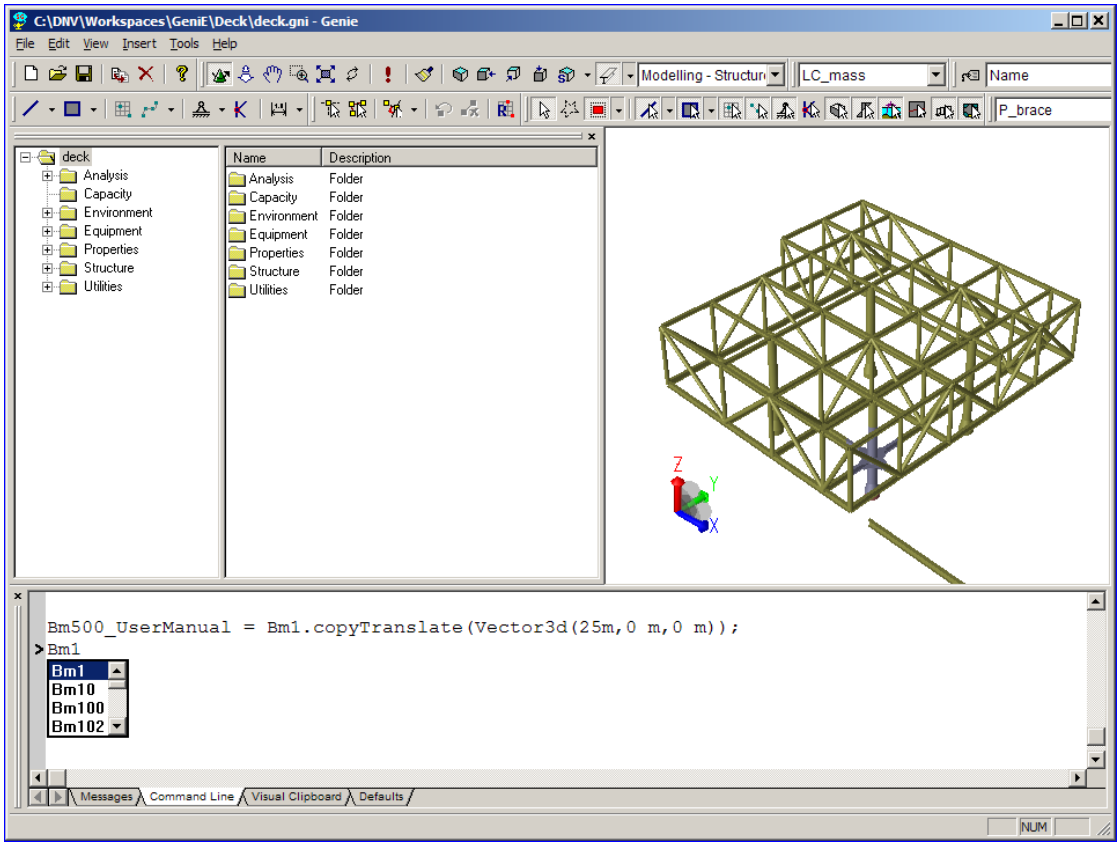
Example on a tool button menu



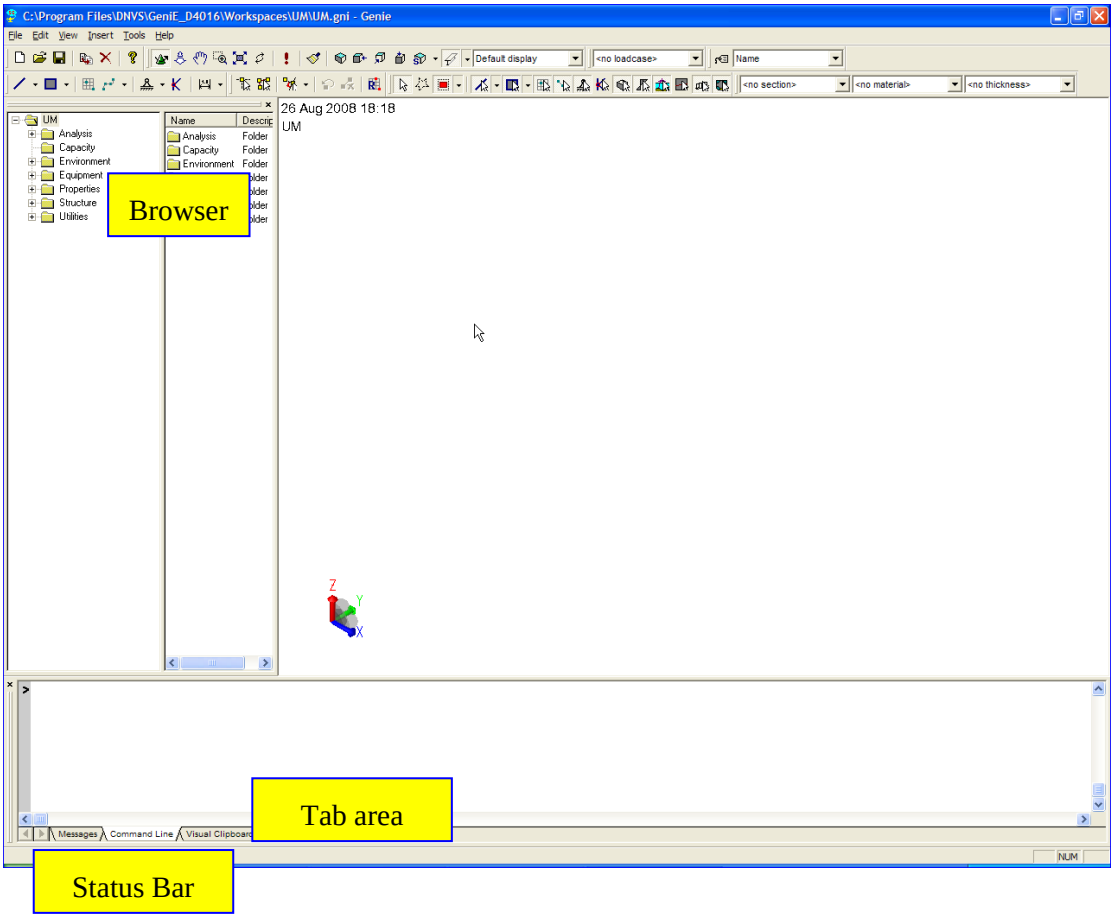
Example on a context sensitive menu (select object from graphic window or browser):



Example on command line interface window using the script language:



The user interface has a browser, a tab area, status bars in addition to the graphical window.



2. PULLDOWN MENUS

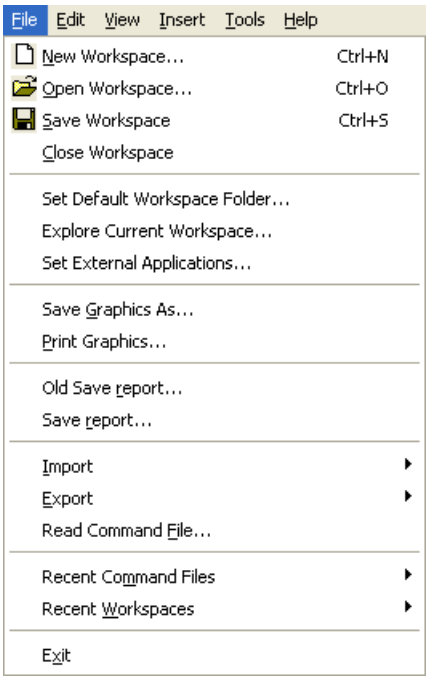
Pulldown menus are those menus available from the graphical window as shown to the right.

There are 6 categories of commands available from the pulldown menu. These are explained below each with a detailed description of their subcommands in the following Sections.

Command category	Subcommand	Explanation
File Commands	New workspace	Make a new project
	Open Workspace	Open an existing project
	Save Workspace	Save the active project session
	Close Workspace	Close the active project session
	Set Default Workspace Folder	Specify default storage location
	Explore Current Workspace	Explore your current workspace's folder
	Save Graphics As	Make a picture
	Print Graphics	Print a picture
	Old Save Report	Make a report using the previous template
	Save Report	Make a report
	Import	Import data from various file formats
	Export	Export data to various file formats
	Read Command File	Read and execute a journal file
	Recent Command Files	Read and execute a journal file from 10 last sessions
	Recent Workspaces	Open a project from 10 last sessions
	Exit	Exit the program
Edit Commands	Undo	Go one step backwards
	Redo	Go one step forward
	Undo/Redo Dialog	Specify the step forward or backward from a list
	Set UndoMark	Specify a user defined undo mark
	Copy	Copy an object
	Delete	Delete an object
	Properties	Define or edit properties
	Rules	Specify rules for a number of automatic program actions
	Licenses/features	Manage which licenses to be used by GeniE

Command category	Subcommand	Explanation
View Commands	Refresh Graphics	Clean up the graphics and remove all labels
	Browser	View browser or not
	Tabs	View the tabs or not
	Status Bar	View the status bar or not
	Toolbars	Decide which toolbars to see
	Restore Window Default Settings	Places the Genie window in the middle of the screen with default size and setup of the interface.
	Options	Create and modify view settings
Insert Commands	Beam	Define a beam
	Plate	Define a plate
	Support	Define a support point or line
	Joint	Define a joint concept
	Mass	Define a mass object
	Compartment	Automatically create compartments
	Feature Edge	Insert a mesh control line
	Linear Slicer	Define a tool used for computing and adjusting target forces and moments, e.g. in length direction of a floating vessel.
	Guiding Geometry	Insert guiding geometry
	Profile	Define a 2D profile used in punch operations
	Equipment	Define an equipment
	Explicit Load	Define explicit load like e.g. point/line/pressure load
	Load Case	Define a load case
	Load Combination	Create load combinations
	Environment	Define the environment for wind/wave/pile-soil analysis
Tools Commands	Analysis	Control analysis activities and look at results
	Equipment	Import weight lists
	Properties	Scale mass materials to target value
	Structure	Perform punching and validation of geometry
	Dimension	Find length and angles
	Customize	Set defaults for printing and naming conventions
Help Commands	Help Topics	Access to user documentation, tutorials and script language
	Status Lists on the Web	Access to on-line status list
	DNV Software on the Web	Access to DNV Software web site
	About GeniE	Lists sub-contracted software used in GeniE

2.1 The File pulldown menu



The File pulldown menu is where you can create, save and work from existing projects.

You can also import and export data from your computer or from the internet, i.e. GeniE's help pages.

See also Section 3.1 of User Manual Volume 1 for more details.

2.1.1 New workspace

Purpose: Make a new project

This command is not scripted. If non-default database units are selected these will be logged. Typically when selecting "mm" as database units:

GenieRules.Units.setDatabaseUnits("mm", "N", "delC");

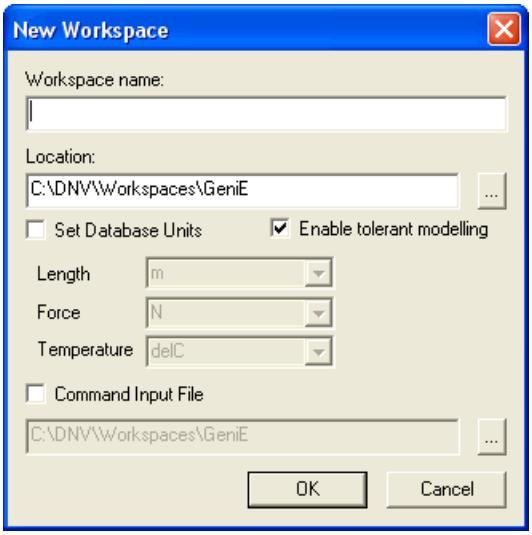
GenieRules.Units.setInputUnit(Length, "mm");

GenieRules.Units.setInputUnit(Force, "N");

GenieRules.Units.setInputUnit(TempDiff, "delC");

GenieRules.Tolerances.useTolerantModelling = true;

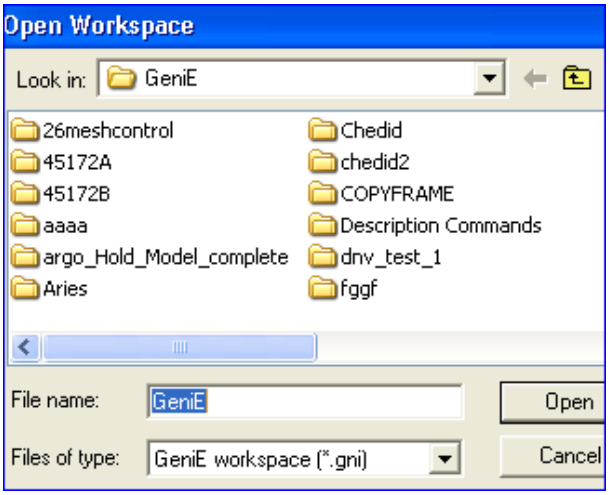
GenieRules.Compatibility.version = "V4.0-14";



2.1.1 Open workspace

Purpose: Open up existing project

This command is not scripted.



2.1.2 Save workspace

Purpose: Save model to database

This command is not scripted.

2.1.3 Close workspace

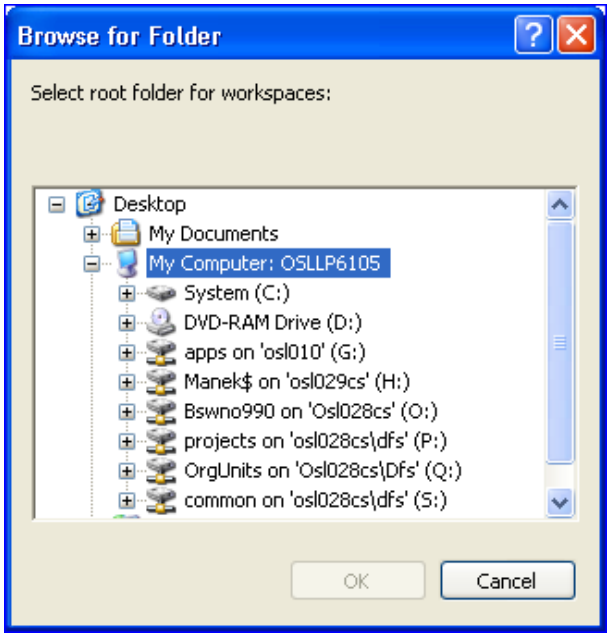
Purpose: Close a workspace

This command is not scripted.

2.1.4 Set Default workspace folder

Purpose: Choose were you will save your projects

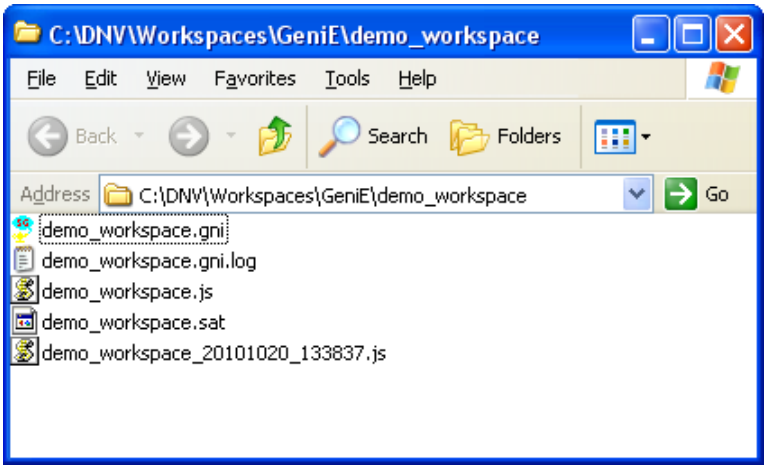
This command is not scripted. The working directory is stored in the Registry for the current GeniE version (in the user profile for the current user)



2.1.5 Explore Current Workspace

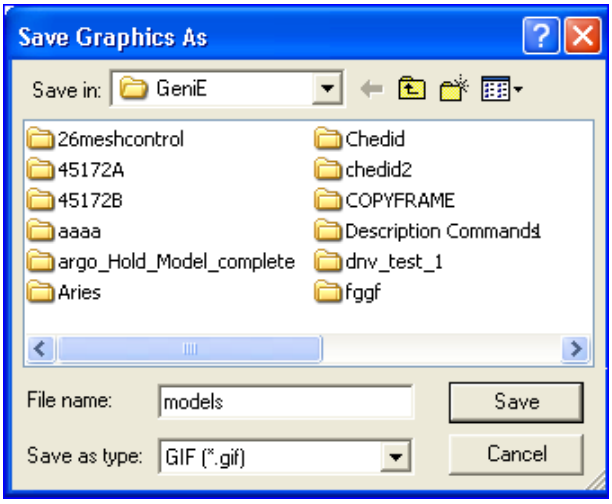
Purpose: Opens a new file explorer window showing the content of the folder where your current workspace is located.

This command is not scripted.



2.1.6 Save graphics as

Purpose: Save a graphics file of the current display



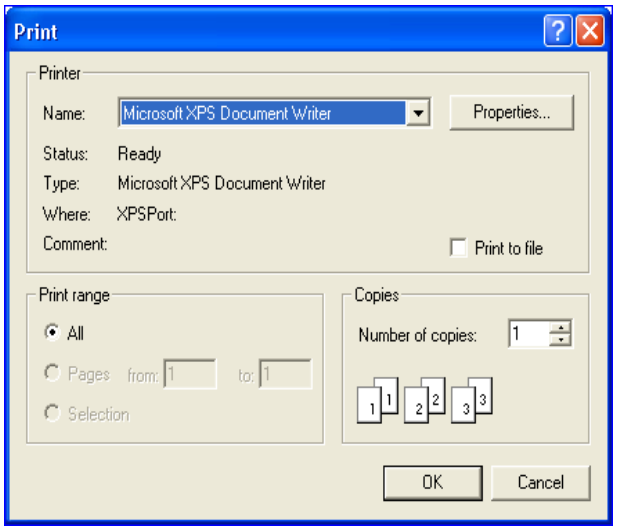
This command is not scripted. To include picture generation in a journal file, see the last Chapter of this User Manual. Typically:

Graphics.saveImage(name,width,height);

2.1.7 Print graphics

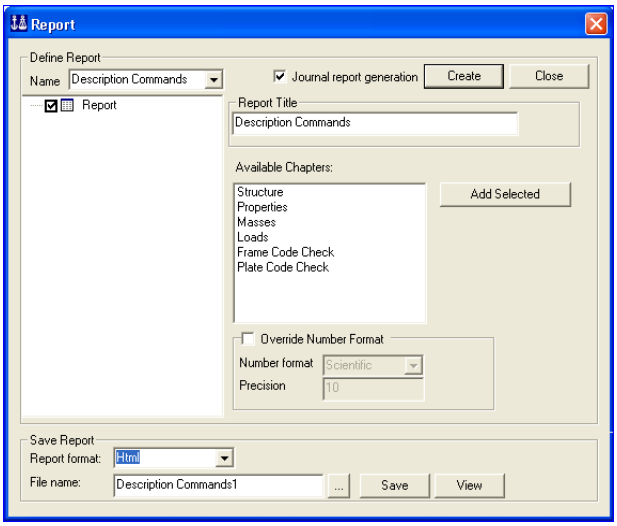
Purpose: Print a graphics file directly on printer

This command is not scripted.



2.1.8 Save report

Purpose: Save a print of your model to various formats for viewing in a notepad editor, an internet explorer, MS Excel or MS Word. The formats are text, HTML and XML.



This command is scripted and the scripting is dependent on your choice for which parts to include in the report. Typically parts of it may be:

jacket_cc1 = Report("jacket_cc");

jacket_cc1.add(ChapterLoads());

jacket_cc1.element(1).analysis = PSI;

jacket_cc1.element(1).add(TableLoadCase());

jacket_cc1.element(1).add(TableLoadSummary());

2.1.8.1 Structure

Purpose: To document the structural parts as shown to the right. The tick off marks indicates program default.

This command is scripted and the scripting is dependent on your choice for which parts to include in the report. Typically parts of it may be:

a1 = Report("a");

a1.add(ChapterStructure());

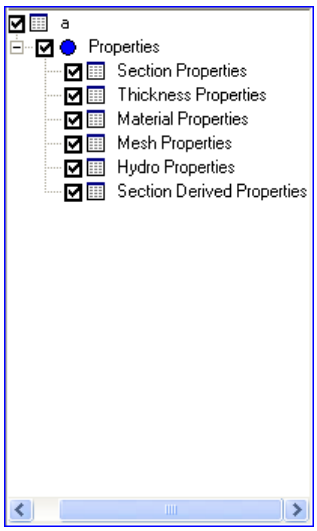
a1.element(1).add(TableBeamCoordinate());

a1.element(1).add(TableBeamProperty());

a1.element(1).add(TableBeamHydroProperty());

2.1.8.2 Properties

Purpose: To document the properties as shown to the right. The tick off marks indicates program default.



This command is scripted. Typical parts of it when saving it as a Word document:

a1 = Report("a");

a1.add(ChapterProperties());

a1.element(1).add(TablePropertySection());

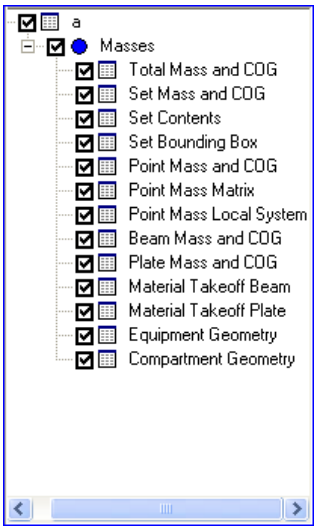
a1.element(1).add(TablePropertyHydro());

a1.element(1).add(TableSectionProperty());

a1.saveAs("a1.doc", mrWordXML);

2.1.8.3 Masses

Purpose: To document the mass elements as shown to the right. The tick off marks indicates program default.



This command is scripted. Typical parts of it:

a1 = Report("a");

a1.add(ChapterMasses());

a1.element(1).add(TableTotalMassAndCOG());

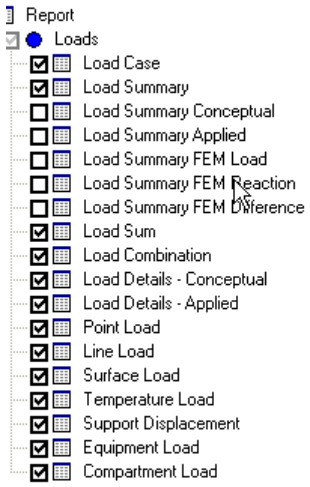
a1.element(1).add(TableSetMassAndCOG());

a1.element(1).add(TableSetContents());

a1.element(1).add(TableSetBoundingBox());

2.1.8.4 Loads

Purpose: To document the load parts as shown to the right. The tick off marks indicates program default.



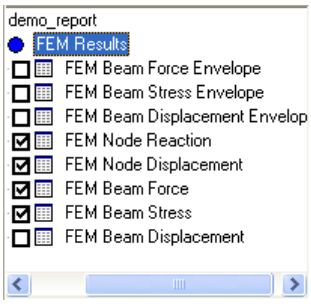
.

This command is scripted. Typically when documenting the "Load Summary", the "FEM Node Reaction" and the "FEM Node Displacement" for Analysis1:

```
a1 = Report("a");
a1.add(ChapterLoads());
a1.element(1).analysis = Analysis1;
a1.element(1).add(TableLoadSummary());
a1.saveAs("a1.doc", mrWordXML);
```

2.1.8.5 FEM Results

Purpose: To document the FEM Results as shown to the right. The tick off marks indicates program default.

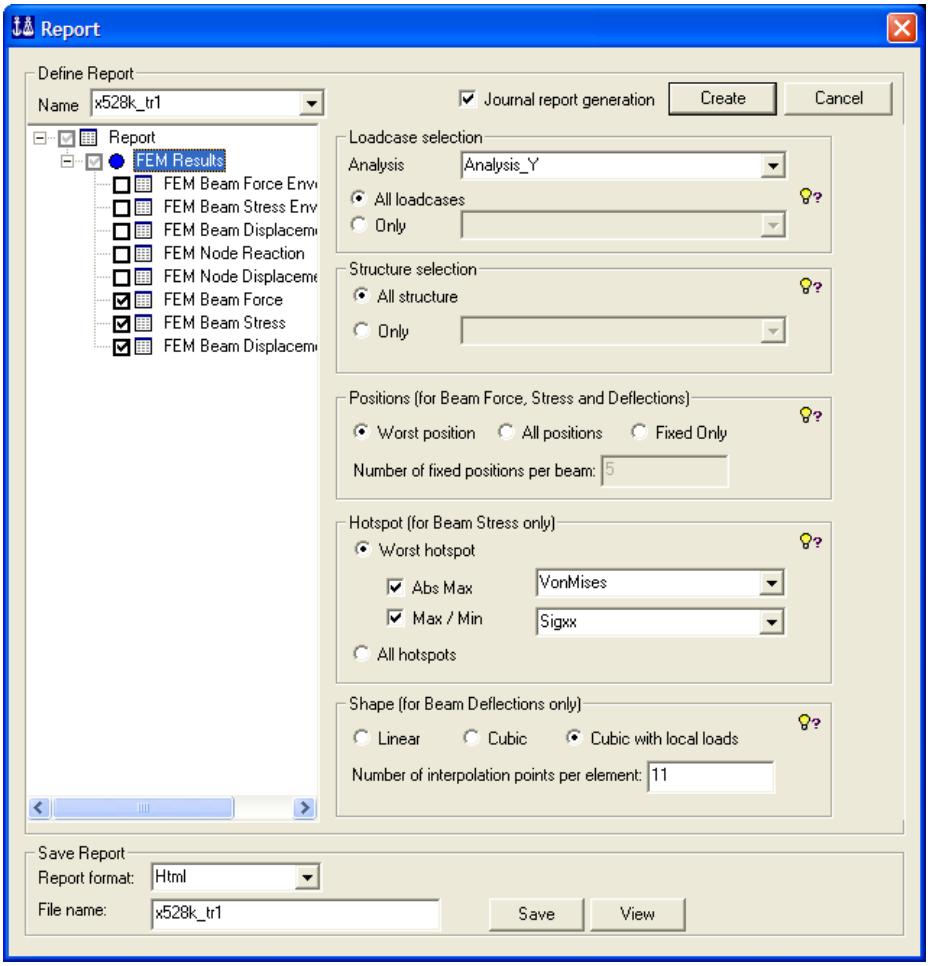


Notice that the reaction forces will be printed for all finite nodes with a boundary condition with reference to the finite element number. In case you specify a joint for a boundary condition, the joint name is also used as a reference. Finite element node displacements are listed for specified joints only

This command is scripted. Typically when documenting the "Beam Force Envelope", the "FEM Beam Stress Envelope", the "FEMNodeReaction", the "FEM Node Displacement", the "FEM Beam Force" and the "FEM Beam Stress" for Analysis1:

```
demo_report = Report("demo_report");
demo_report.add(ChapterFEMResults());
demo_report.element(1).analysis = Analysis1;
demo_report.element(1).setNoLimit();
demo_report.element(1).add(TableFEMBeamForceEnvelope());
demo_report.element(1).add(TableFEMBeamStressEnvelope());
demo_report.element(1).add(TableFEMNodeReaction());
demo_report.element(1).add(TableFEMNodeDisplacement());
demo_report.element(1).add(TableFEMBeamForce());
demo_report.element(1).add(TableFEMBeamStress());
demo_report.saveAs("demo_report.xml", mrWordXML);
```

2.1.8.6 Explanation of the FEM Results settings



Loadcase selection

Analysis:

Select the analysis you want to report. The Beam Force Envelope table and Beam Stress Envelope table "scan" through all the loadcases in the analysis and show the worst results. For the other tables in this chapter, you only get the results individually from each loadcase of the selected analysis.

All loadcases:

Shows the results for all loadcases in the selected analysis.

Only:

Shows the results for the selected loadcase only.

Structure selection

All structure

Shows the results for all the structure in your model. Note that generating the report can be quite time consuming for a large model if you select all structure.

Only

Shows the results for the selected named set only.

Positions (for Beam Force, Stress and Deflections)

These settings are only influencing the Beam Force and Beam Stress table.

Worst position

Shows the results for the position on the beams having the largest moments for the Beam Force table, or the highest stresses as specified under Hotspot for the Beam Stress table.

All position

Shows the results in both the specified fixed positions and in the worst positions.

Fixed only

Shows the results only in the specified fixed positions.

Number of fixed positions per beam

Here you specify the number of positions along a beam that you want to check. The minimum is 2, meaning only the two end points, the maximum is 100. Default is 5. Note that if you are specifying a high number of fixed positions on a large model, it will be time consuming to generate the report.

Hotspot

Worst hotspot, Abs Max

Shows the absolute max value of the stress component given in the combo box.

Worst hotspot, Max/Min

Shows the maximum and minimum values of the stress component given in the combo box.

All hotspots

Shows the stresses in all hotspots.

Shape (for Beam Deflections only)

Linear

Implies that each finite element is approximated by a straight line, meaning that the two end points of each element are calculated.

Cubic

Implies that each finite element is approximated by a cubic spline function that matches the beam end rotations. Specified number of interpolation points per element is calculated along the spline.

Cubic with local loads

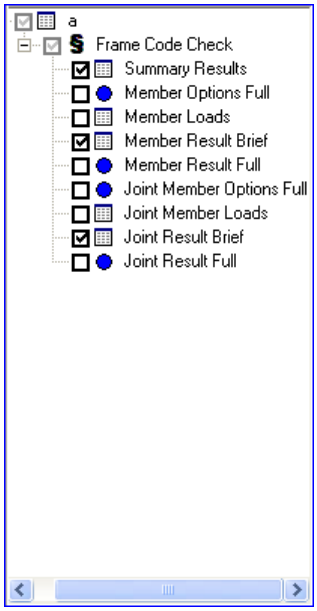
Implies that an analytical solution of deflections due to local element loads is superimposed on the cubic spline function. Specified number of interpolation points per element is calculated along the spline.

Number of interpolation points per element

Number of interpolation points must be between 5 and 25 and is only relevant for the cubic options.

2.1.8.7 Frame code check

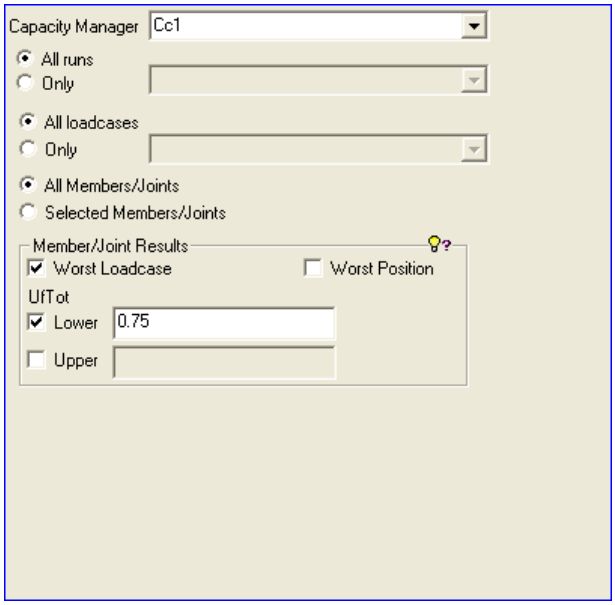
Purpose: To document the results from a frame code check as shown to the right. The tick off marks indicates program default.



This command is scripted. Typically when documenting the "Summary Results", "Member Options Full" and "Member Result Brief" for all utilisation factors above 0.75 for all available code check runs:

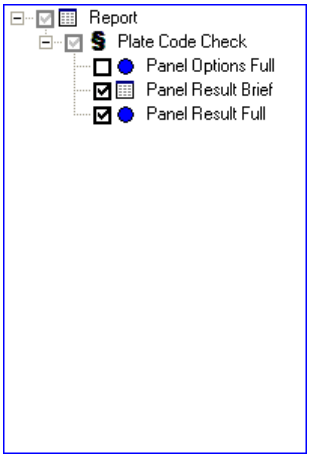
```
a1 = Report("a");
a1.add(FrameCodeCheckChapter());
a1.element(1).run = Cc1.allRuns;
a1.element(1).worstLoadCase = true;
a1.element(1).limit = LimitLower("UfTot", 0.75);
a1.element(1).add(TableSummaryResult());
a1.element(1).add(TableMemberOptionsFull());
a1.element(1).add(TableMemberResultBrief());
a1.saveAs("a1.doc", mrWordXML);
```

In this case the filter options are set from the dialogue as shown to the right:



2.1.8.8 Plate code check

Purpose: To document the results from a plate code check as shown to the right. The tick off marks indicates program default.



This command is scripted. Typically when saving as an Excel document using the filter options as shown below:

a1 = Report("a");

a1.add(PlateCodeCheckChapter());

a1.element(1).run = Cc1.allRuns;

a1.element(1).worstPosition = true;

a1.element(1).worstLoadCase = true;

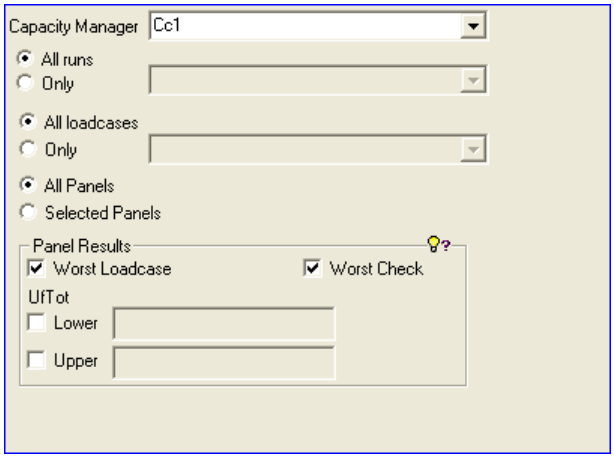
a1.element(1).add(TablePanelResultBrief());

a1.element(1).element(2).setPrimarySortColumn("UfTot", false);

a1.element(1).element(2).setInheritParentLimits(true);

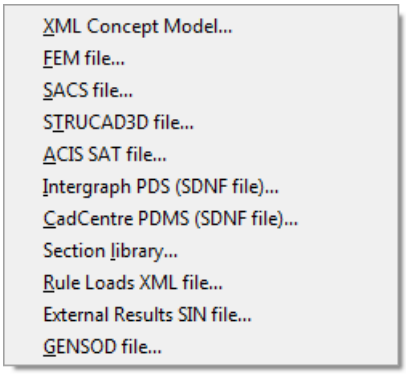
a1.element(1).add(TablePanelResultFull());

a1.saveAs("a1.xls", mrExcelXML);



2.1.9 Import

To import data from the following sources:



2.1.9.1 XML Concept Model

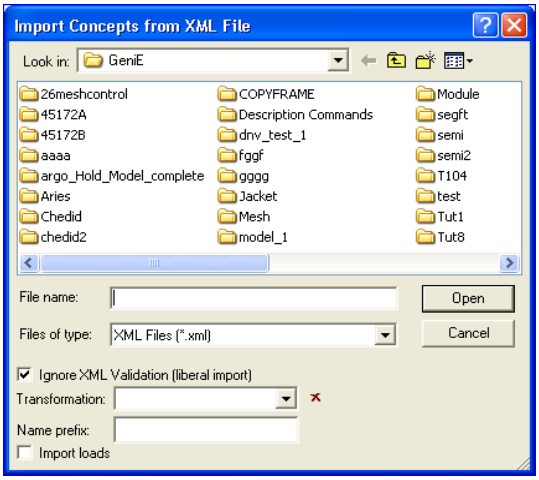
Purpose: Import a structural concept model.

This command is scripted and a typical example is given below.

XmlImporter = ImportConceptXml();

XmlImporter.importLoads = true;

XmlImporter.DoImport("C:/Location/yourname.xml");



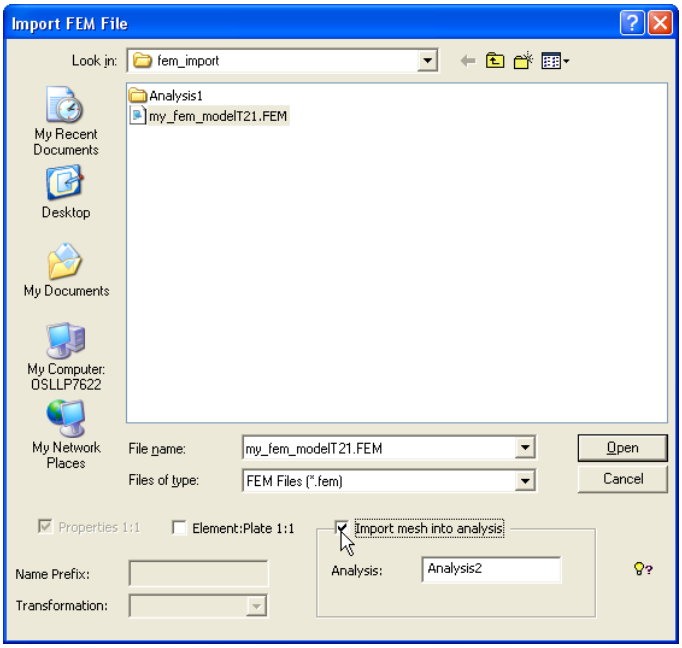
2.1.9.2 FEM file

Purpose: Import geometry and loads from FEM file and create concept model.

This command is scripted and a typical example is given below.

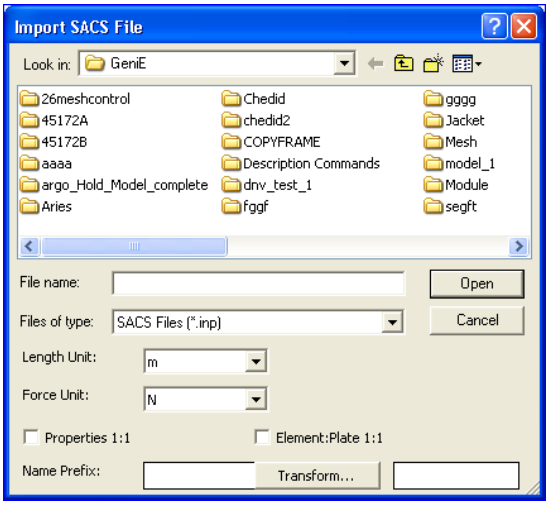
FemImporter = ImportMeshFem();

FemImporter.DoImport("C:/Location/T1.FEM");



2.1.9.3 SACS file

Purpose: Import geometry and loads from a Sacs input file and create concept model. The items that are converted are documented on the Reference Document section on GeniE's help pages.



This command is scripted and a typical example is given below.

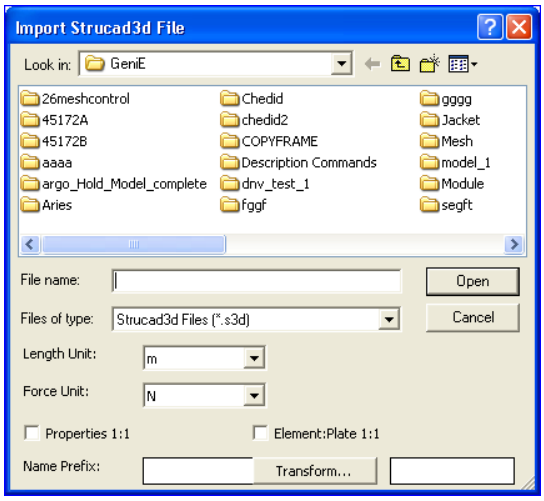
SacsImporter = ImportMeshSacs();

SacsImporter.setOutputUnits("m","N");

SacsImporter.DoImport("C:/Location/yourname.inp");

2.1.9.4 STRUCAD 3D file

Purpose: Import geometry and loads from a Strucad input file and create concept model. The items that are converted are documented on the Reference Document section on GeniE's help pages.



This command is scripted and a typical example is given below.

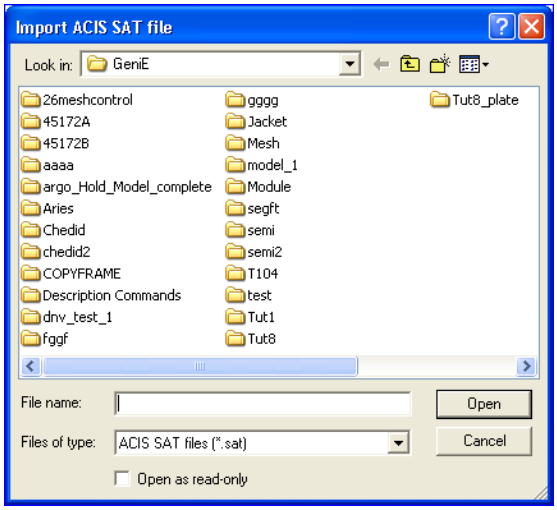
Strucad3dImporter = ImportMeshStrucad3d();

Strucad3dImporter.setOutputUnits("m","N");

Strucad3dImporter.DoImport("C:/Location/yourname.s3d");

2.1.9.5 ACIS SAT file

Purpose: Import geometry and create concept model



This command is scripted and a typical example is given below.

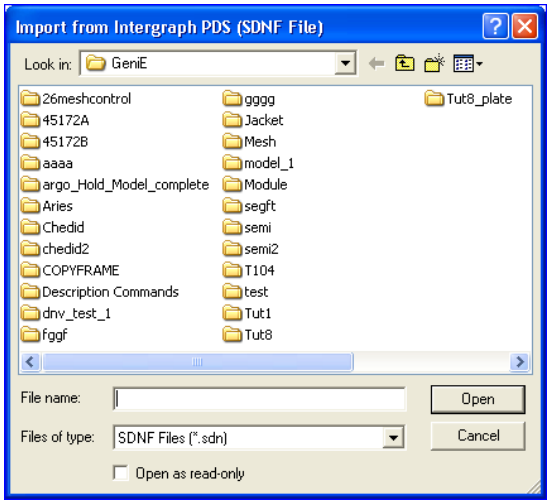
SatImporter = ImportGeometrySat();

SatImporter.DoImport("C:/Location/yourname.sat");

2.1.9.6 Intergraph PDS (SDNF file)

Purpose: Import geometry and create concept model

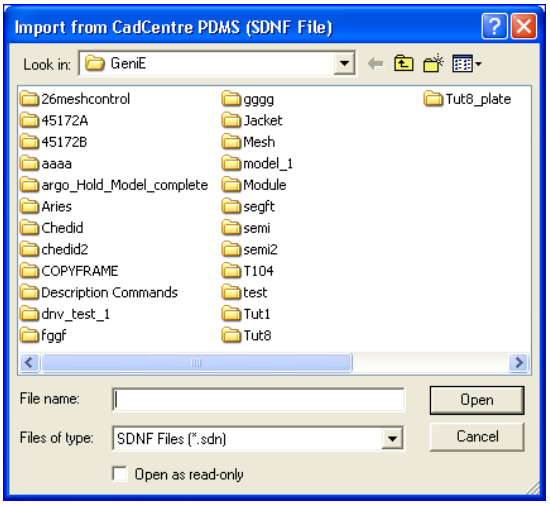
This command is not scripted.



2.1.9.7 CadCentre PDMS (SDNF file)

Purpose: Import geometry and create concept model

This command is not scripted.



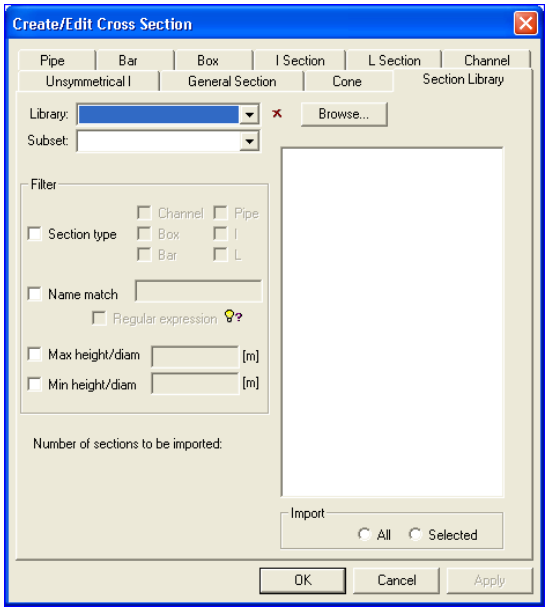
2.1.9.8 Section Library

Purpose: Import AISC, Euronorm/Norwegian Standard Libraries or common bulb, angle or tee cross sections.

This command is scripted. The scripting depends on how many profiles that are selected. A typical example is given below where one pipe section OD101_6X10 has been selected.

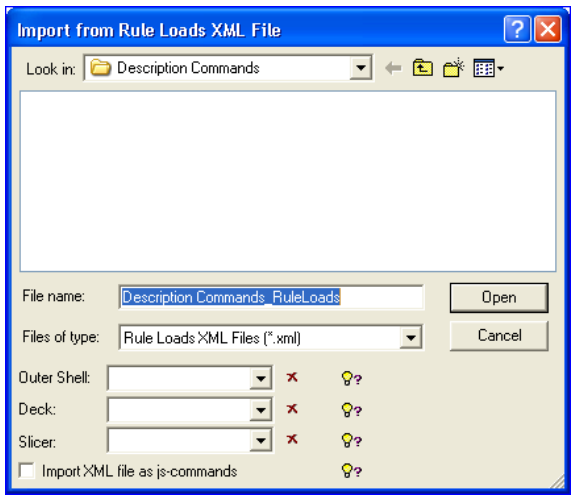
OD101_6X10 = PipeSection(101.6mm,10mm);

OD101_6X10.description = "NVS lib : 101,6x10 NS-EN 10210/10219";



2.1.9.9 Rule Loads XML file

Purpose: To import loads and boundary conditions as defined by Nauticus Hull using the CSR rules for bulk ships.



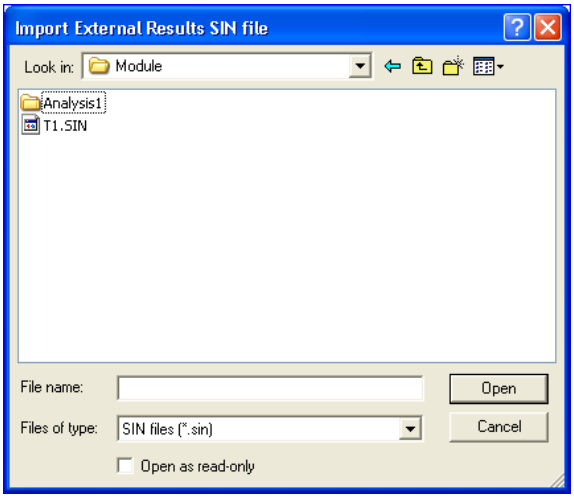
This command is scripted and a typical example is given below.

RuleLoadsImporter = ImportRuleLoadsXml();

RuleLoadsImporter.DoImport("C:/Location/yourname.xml");

2.1.9.10 External Results SIN file

Purpose: To import a results file on SIN format. There are strict limitations on this functionality. See User Manual Volume 4 for documentation.



This command is scripted and a typical example is given below.

SinImporter = ImportResultsSin();

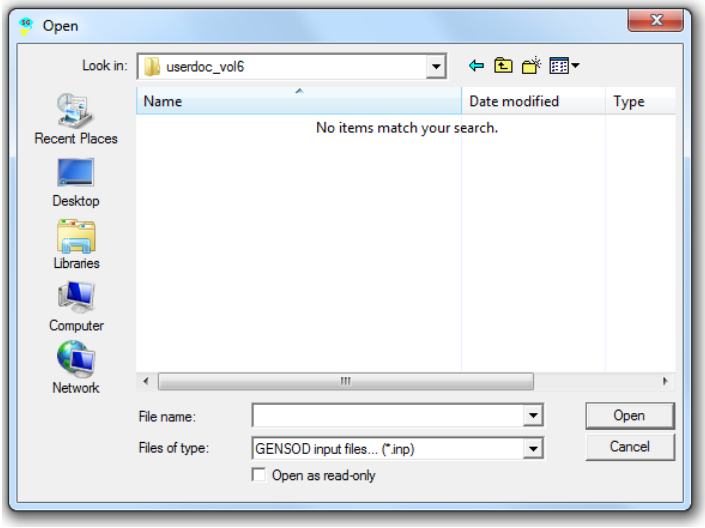
SinImporter.DoImport("C:/Location/T1.SIN");

2.1.9.11 GENSOD file

The stand-alone application GENSOD generates the soil data needed by SPLICE, storing them in GENSOD.inp.

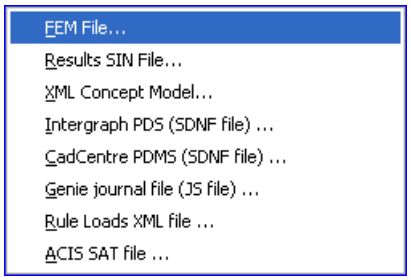
Purpose : To import a GENSOD.inp file.

GeniE restores the information from the GENSOD.inp file by reproducing the js-commands, thus supplying the folders “Environment/Location” and “Environment/Soil” of the GeniE browser with the relevant data.



2.1.10 Export

To export data to the following sources:



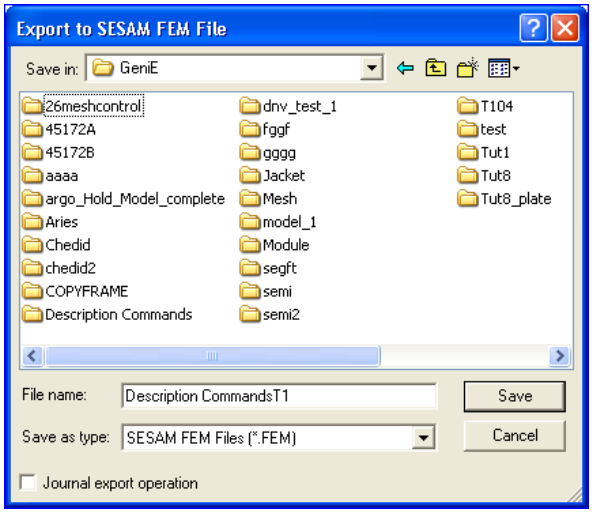
2.1.10.1 FEM file

Purpose: To export a FEM file with a user given name.

This command is not scripted unless you tick off for such option. To include FEM export in a journal file, see the last Chapter of this User Manual. Typically:

FemExporter = ExportMeshFem();

FemExporter.DoExport("C:/Location/T1.FEM");



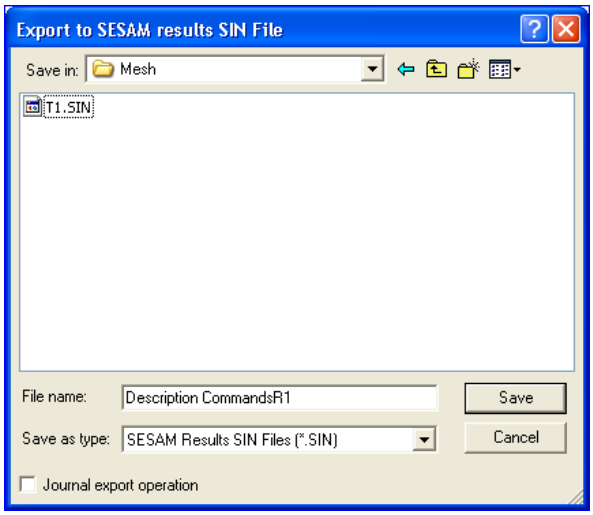
2.1.10.2 Results SIN file

Purpose: To export a SIN results file with a user given name.

This command is not scripted unless you tick off for such option. To include SIN export in a journal file, see the last Chapter of this User Manual. Typically:

SinExporter = ExportResultsSin();

SinExporter.DoExport("C:/Location/R1.SIN");



2.1.10.3 XML Concept file

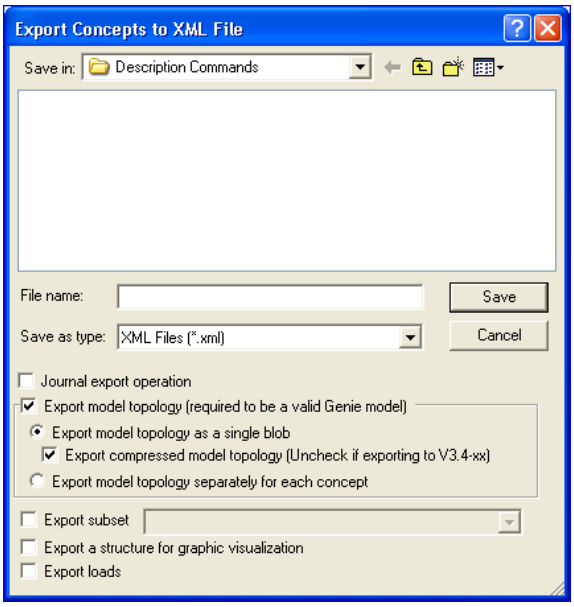
Purpose: Export concept model to XML format. The parts of the concept model that are exported to xml format are documented in User Manual Volume 3.

This command is not scripted unless you tick off for such option. To include XML export in a journal file, see the last Chapter of this User Manual. Notice that export of loads is a separate tick off option. Typically:

XmlExporter = ExportConceptXml();

XmlExporter.exportLoads = true;

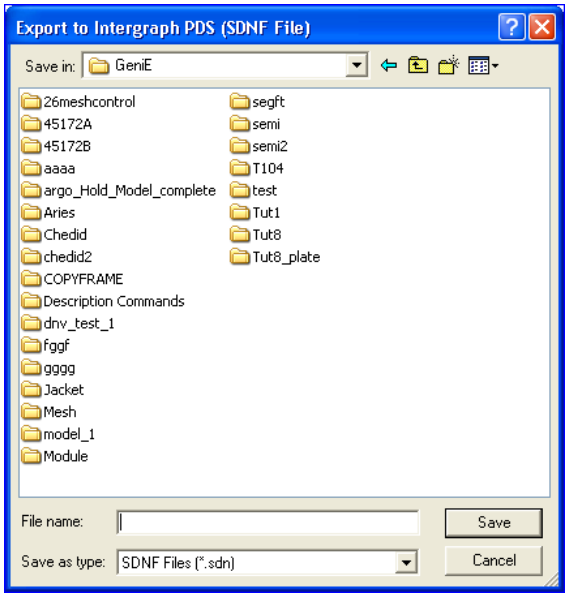
XmlExporter.DoExport("C:/Location/yourname.xml");



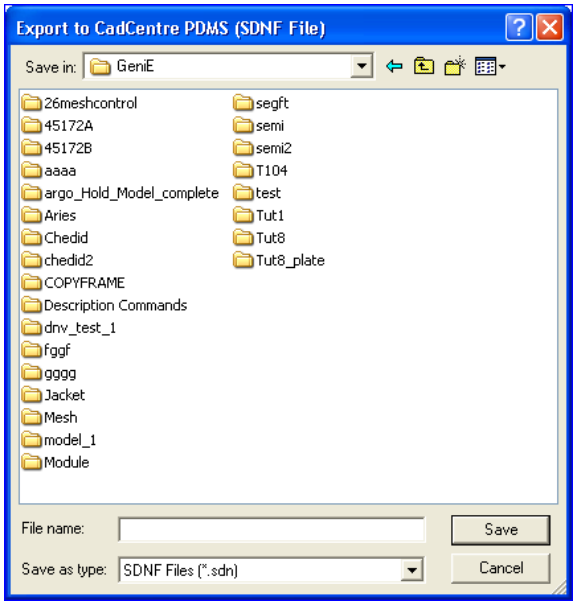
2.1.10.4 Intergraph PDS (SDNF file)

Purpose: Export geometry to PDS format

This command is not scripted.



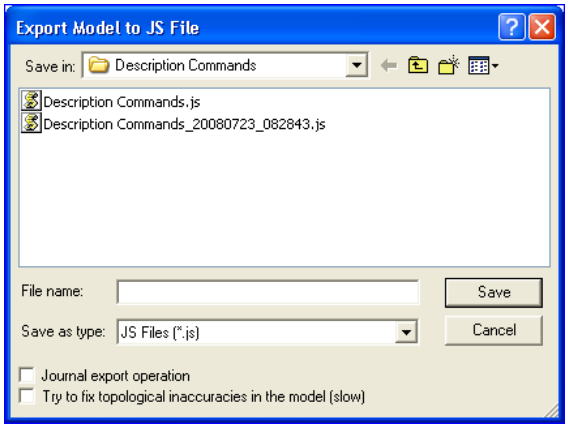
2.1.10.5 CadCentre PDMS (SDNF file)



Purpose: Export geometry to PDMS format

This command is not scripted.

2.1.10.6 GeniE journal file (JS)



Purpose: Create and export a clean journal file for frame and plate structures. The parts of the concept model that are exported to js format are documented in User Manual Volume 3.

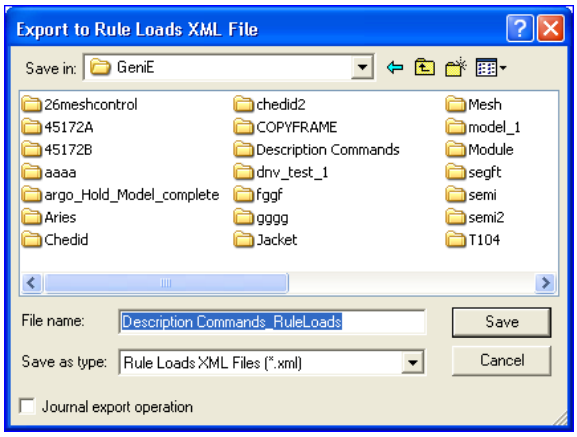
This command is not scripted unless you tick off for such option. To include js export in a journal file, see the last Chapter of this User Manual. Notice that export of loads is a separate tick off option. Typically:

JsExporter = ExportModelJS();

JsExporter.DoExport("C:/Location/yourname.js");

2.1.10.7 Rule Loads XML file

Purpose: Export compartment and some structure information for use by Nauticus Hull to create CSR rule loads.



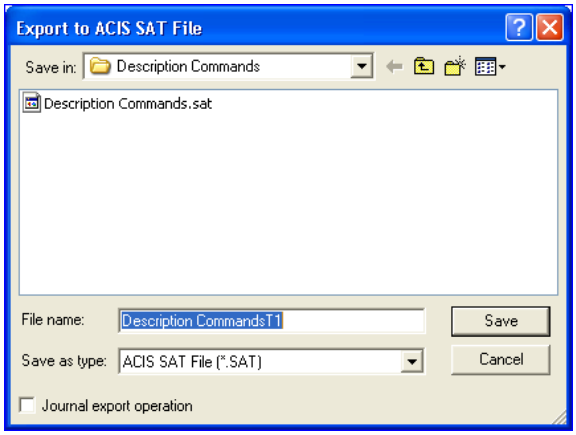
This command is not scripted unless you tick off for such option. Typically:

RuleLoadsExporter = ExportRuleLoadsXml();

RuleLoadsExporter.DoExport("C:/Location/yourname.xml");

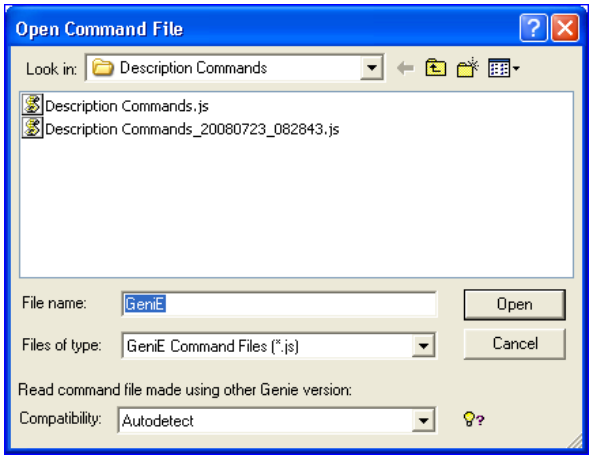
2.1.10.8 ACIS SAT file

Purpose: Export geometry to Acis Sat file format.



This command is not scripted.

2.1.11 Read command File

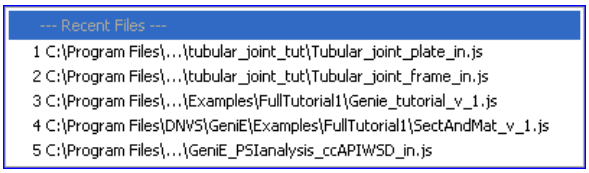


Purpose: Read in and execute a journal file (.js file).

This command is not scripted directly, but the content of the .js file is scripted. You may edit the journal file to include the operation. Typically:

ReadCommandFile("C:/Location/yourname.js");

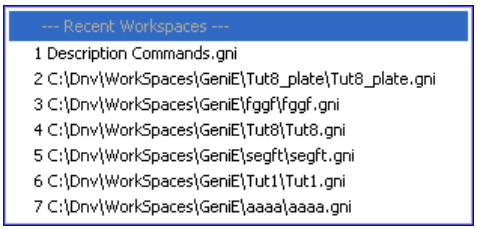
2.1.12 Recent command file



Purpose: Read in and execute one of the 10 last used journal files

This command is not scripted directly, but the content of the .js file is scripted.

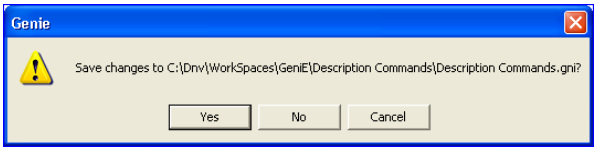
2.1.13 Recent workspaces



Purpose: Read in and open one of the 10 last used workspaces

This command is not scripted.

2.1.14 Exit



Purpose: Exit program and save workspace

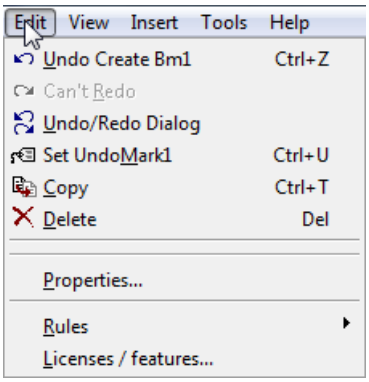
This command is not scripted.

2.2 The Edit pulldown menu

The Edit pulldown menu is where you can copy, delete, undo or redo, and change the rules and properties for your working material like i.e. materials, thicknesses, meshing and joints.

2.2.1 Undo/Redo

Purpose: You can go back and redo recent work you are not satisfied with or things you have done wrong. The history log is deleted when you save the work.



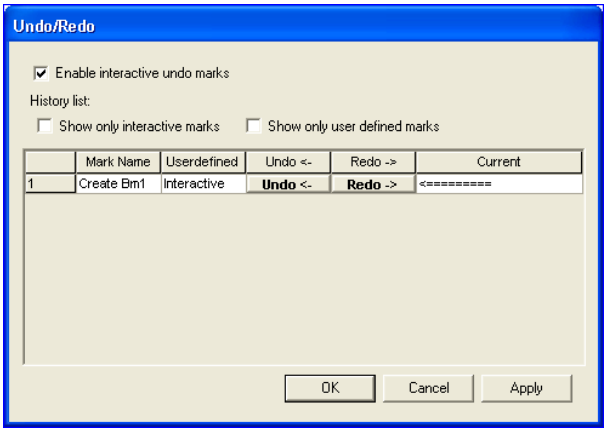
This command is not scripted. This command will modify the script.

When doing undo, you will remove recently added lines from the journal.

When doing redo, you will add these lines back to the journal.

2.2.2 Undo/Redo Dialog

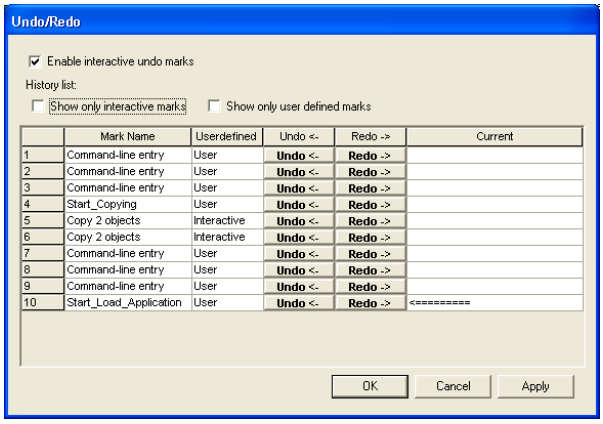
Purpose: Specify the step forward or backward from a list. You can go back or forward to undo or redo recent work you are not satisfied with or things you have done wrong. The history log is deleted when you save the work.



This command is not scripted.

2.2.3 Set Undo Mark

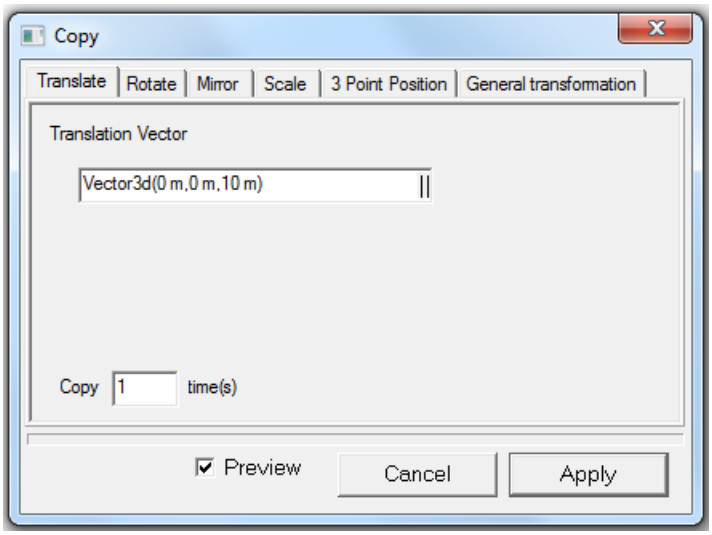
Purpose: To include a user defined undo mark in the undo/redo dialogue.



This command is not scripted. You may include undo marks in the journal file so that it is easy for you to find locations for undo or redo operations. Typically:

setUndoMarker("yourname");

2.2.4 Copy



Purpose: Copy a selected object using translate, rotate, mirror, 3 point position or a general transformation.

This command is scripted. The scripting depends on which operation that is carried out. Typically for a copy translate operation of beam Bm1 (10 meters in global z-direction):

Bm2 = Bm1.copyTranslate(Vector3d(0 m,0 m,10 m));

2.2.5 Delete

Purpose: Mark a selected object and delete it.

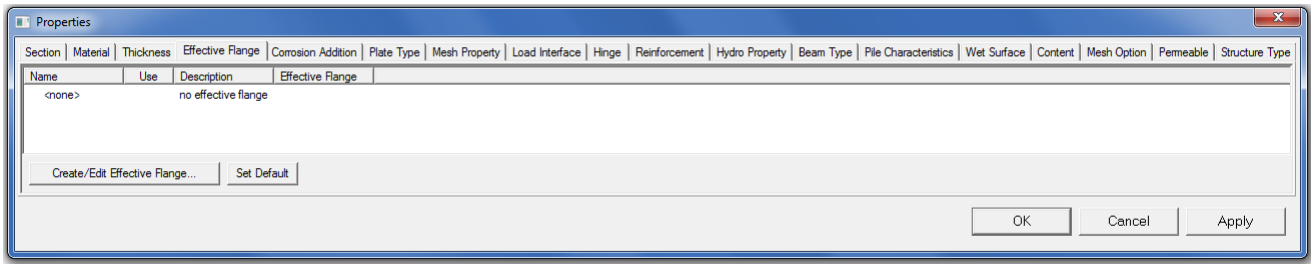
This command is scripted. The scripting depends on which object is deleted. Typically for a delete operation of plate Pl1:

Delete(Pl1);

2.2.6 Properties

Purpose: Edit and modify a property library

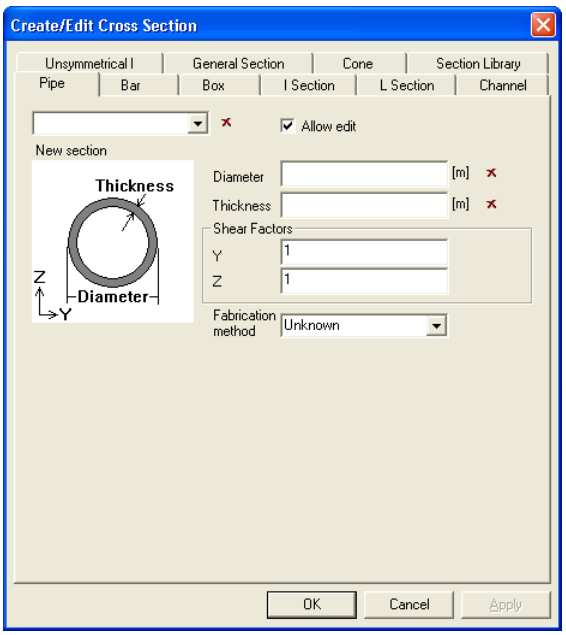
The commands for each of the available properties are shown in the following. They are all accessed from the following menu:



More details may be found on GeniE's help pages under Reference Documents -> Beam Section Parameters.

2.2.6.1 Section

Purpose: To define or edit a section property. The picture to the right shows the dialogue for definition of a pipe section. It is also possible to define section properties of type bar, box, I, L, Channel, Unsymmetrical I (also used to define T and bulb), general section and cones.



This command is scripted. The scripting depends on which property is created. Typically for a pipe section MyPipe where shear factors and fabrication method differ from default values:

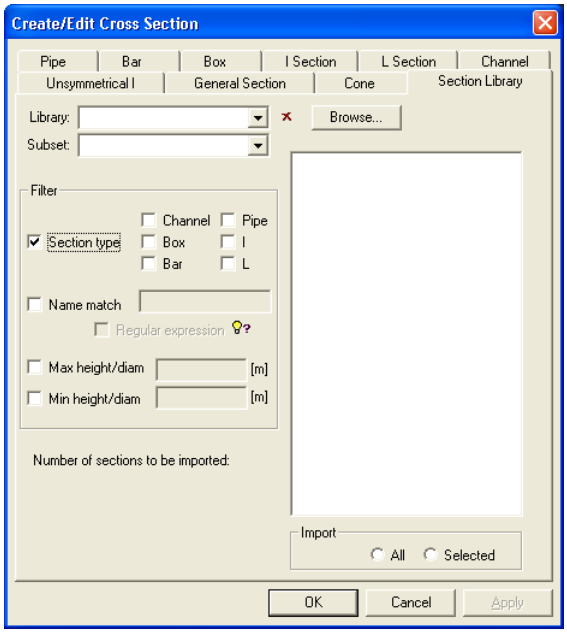
MyPipe = PipeSection(1000mm, 25mm);

MyPipe.shearFactorY = 0.8;

MyPipe.shearFactorZ = 0.9;

MyPipe.fabricationMethod = fmRolled;

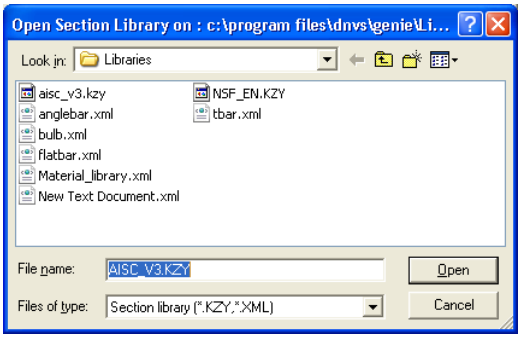
It is also possible to import section profiles from section libraries. The available section libraries are accessed when you push the *Browse* button.



This command is scripted. The scripting depends on which section property that is imported. Typically for a HE200A section from the NSF_EN library:

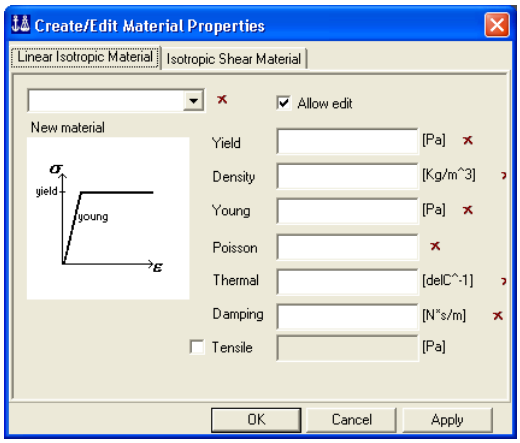
HE200A = ISection(190mm,200mm,6.5mm,10mm);

HE200A.description = "NVS lib : HE 200 A NS-EN 10034";



2.2.6.2 Material

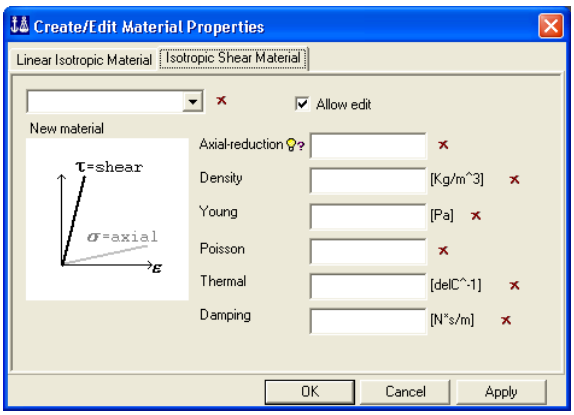
Purpose: To create and edit a material property



This command is scripted. Typically for a linear isotropic material named MyMat:

MyMat = MaterialLinear(275000, 7.85, 210000000, 0.3, 1.2e-5, 3.0e-5, 430000);

It is also possible to define an isotropic shear material.



This command is scripted. Typically for an isotropic shear material named MyShearMat with axial reduction of 75:

MyShearMat = MaterialShear(75, 7.85, 210000000, 0.3, 1.2e-5, 3.0e-5);

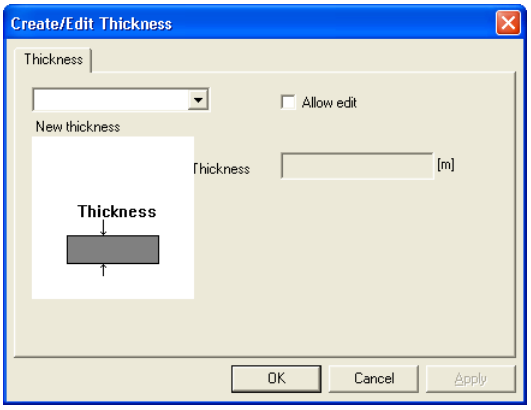
Notice that it is also possible to import a material library from *C:\ProgramFiles\DNVS\GeniE\Libraries\Material_library.xml*.

The path name assumes you have installed the program using the default installation set-up. The library is imported by using the command ***File|Import|XML Concept Model***.

The library contains 71 linear isotropic materials; these are documented in User Manual Volume 3 Appendix B.

2.2.6.3 Thickness

Purpose: To create and edit a thickness property

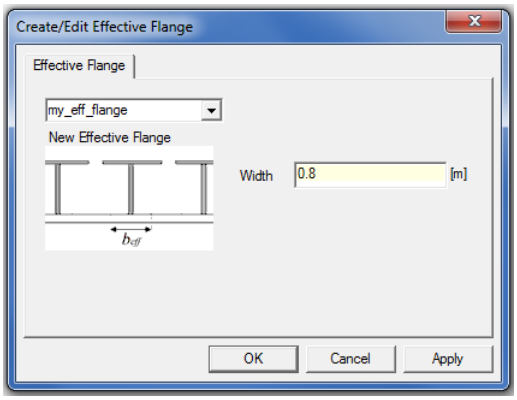


This command is scripted. Typically for a thickness 20mm:

My_thickness = Thickness(20mm);

2.2.6.4 Effective flange

Purpose: To create and edit an effective flange property

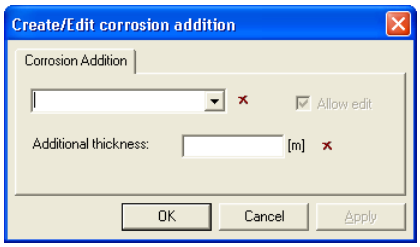


This command is scripted. Typically for an effective flange of 0,8 m:

my_eff_flange = EffectiveFlange(0.8);

2.2.6.5 Corrosion addition

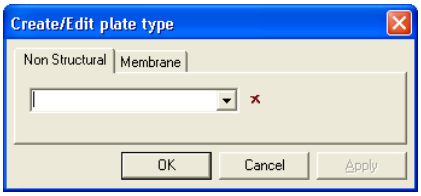
Purpose: To create and edit a corrosion addition used when creating a finite element model using the CSR rules for bulk ships.



This command is scripted. Typically for a corrosion addition of 8 mm:

MyCorrosion = CorrosionAddition(8mm);

2.2.6.6 Plate type



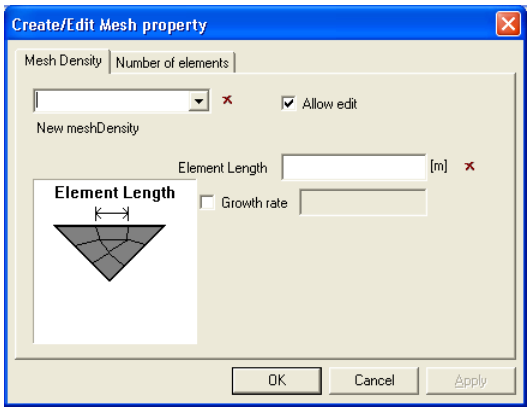
Purpose: To create a non-structural or membrane plate type. Notice that the non-structural plate is used solely to close a volume like a compartment and that the non-structural plate does not carry any loads or contributes to the structural stiffness. The membrane plate will ensure that a plate will be represented as a membrane finite element when creating a FE model.

This command is scripted. Typically:

MyNonStructPlate = PlateTypeNonstruct();
MyMembranePlate = PlateTypeMembrane();

2.2.6.7 Mesh Property

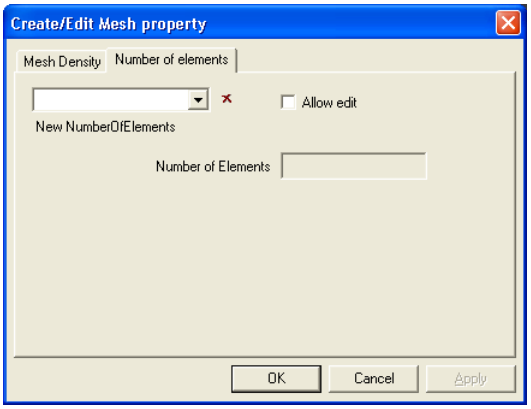
Purpose: To create a mesh density that can be applied to a plate, a beam or a feature edge.



This command is scripted. Typically:

MyMeshDensity = MeshDensity(2.5m);

It is also possible to define number of elements that can be applied to a beam or a feature edge.



This command is scripted. Typically:

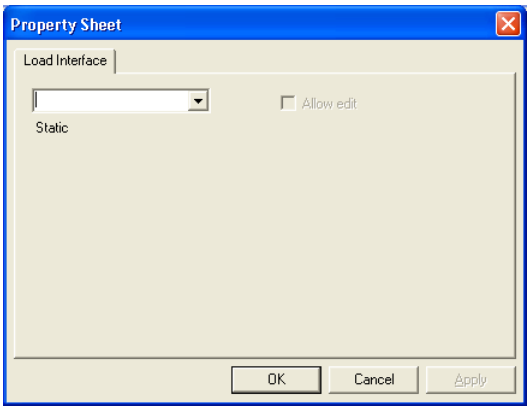
MyMeshNoElements = NumberOfElements(8);

2.2.6.8 Load interface

Purpose: To create a property to decide which beams to receive loads from equipments.

This command is scripted. Typically:

MyLoadInterface = LoadInterface();

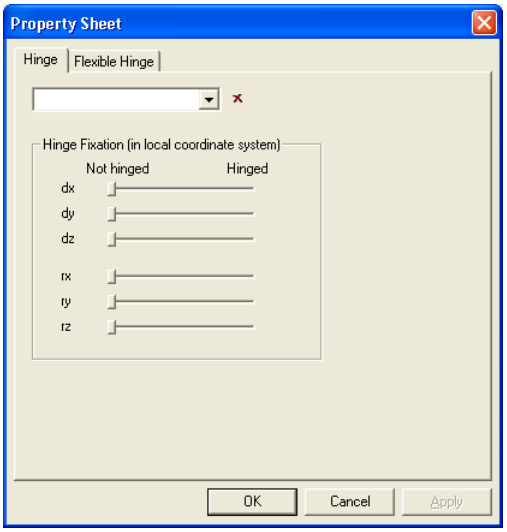


2.2.6.9 Hinge

Purpose: To create hinges to specify which degrees of freedom for a beam end that is connected to another beam.

This command is scripted. Typically for a hinge where all the rotational degrees of freedom are not connected:

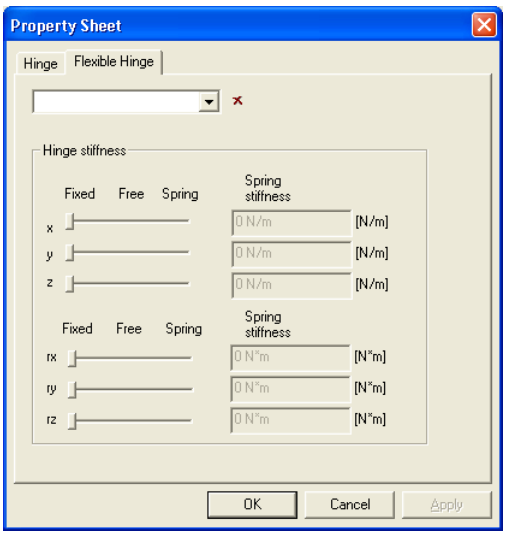
MyHinge = Hinge(1,1,1,0,0,0);



It is also possible to specify a flexible hinge whereby stiffness for each degree of freedom is specified.

This command is scripted. Typically for a flexible hinge where 4 different spring stiffnesses are used (fixed in x and y):

MyFlexHinge = FlexibleHinge(Fixed,Fixed, Stiffness(500),Stiffness(12500), Stiffness(25000),Stiffness(75000));

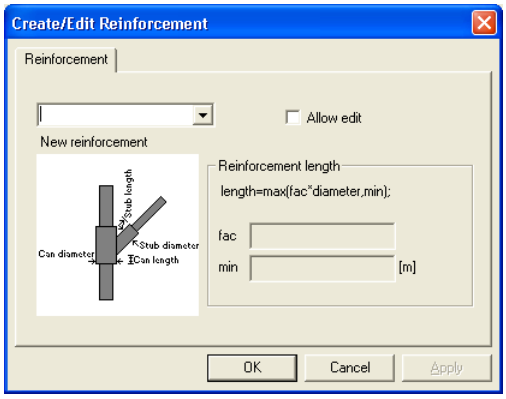


2.2.6.10 Reinforcement

Purpose: To create a reinforcement to be applied to a can or stub.

This command is scripted. Typically:

MyReinforcement = Reinforcement(0.5,0.45m);



2.2.6.11 Hydro property

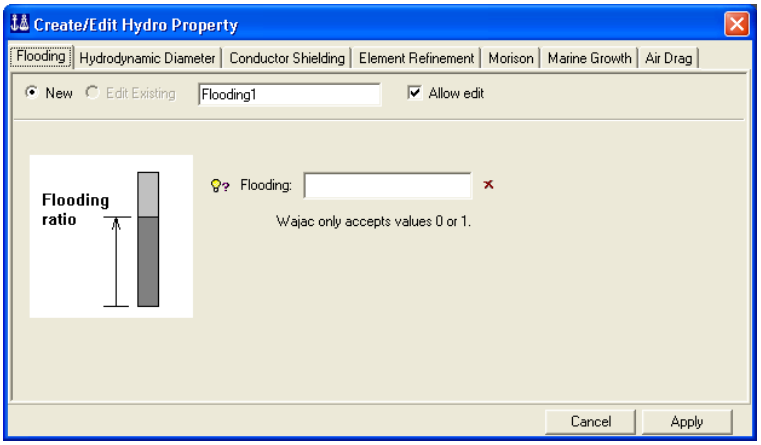
Purpose: To define hydro properties to be applied to beams for use in hydrodynamic analysis based on the Morison theory (Wajac). For more details, please see User Manual Volume 2 and the Wajac User Manual.

2.2.6.11.1 Flooding

Purpose: To define a flooding ratio

This command is scripted. Typically:

MyFlooding = Flooding(1);

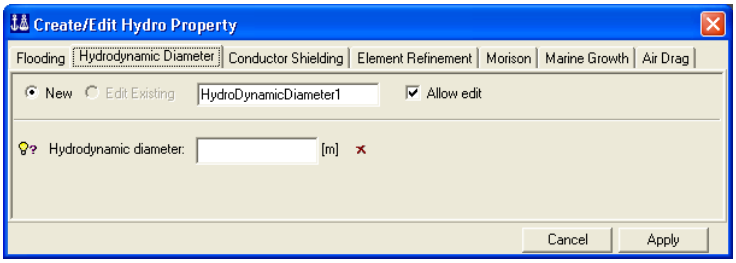


2.2.6.11.2 Hydrodynamic diameter

Purpose: To define a hydrodynamic diameter

This command is scripted. Typically:

MyHydroDiamater = HydroDynamicDiameter(0.75m);

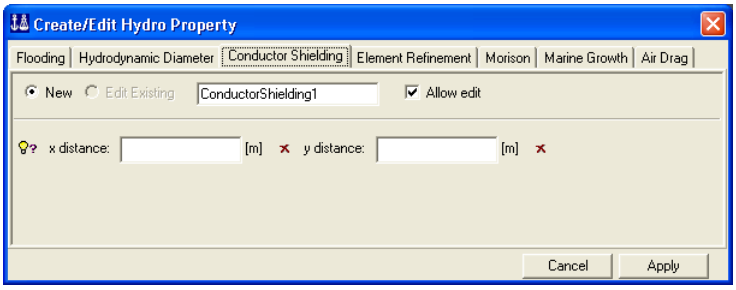


2.2.6.11.3 Conductor Shielding

Purpose: To define a conductor shielding

This command is scripted. Typically:

MyShielding = ConductorShielding(1m,2m);

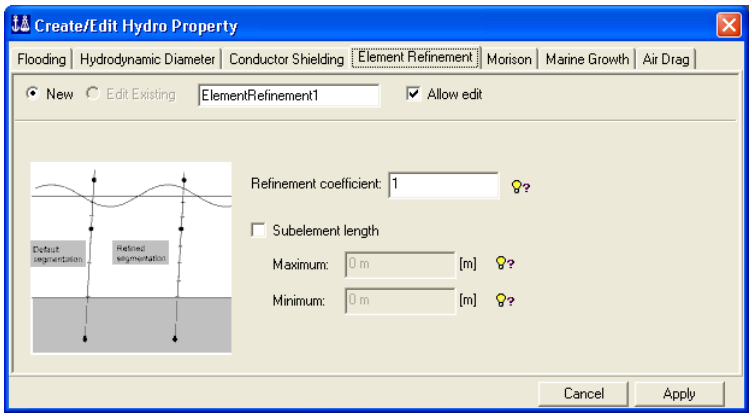


2.2.6.11.4 Element refinement

Purpose: To define an element refinement

This command is scripted. Typically:

MyRefinement = ElementRefinement(5);

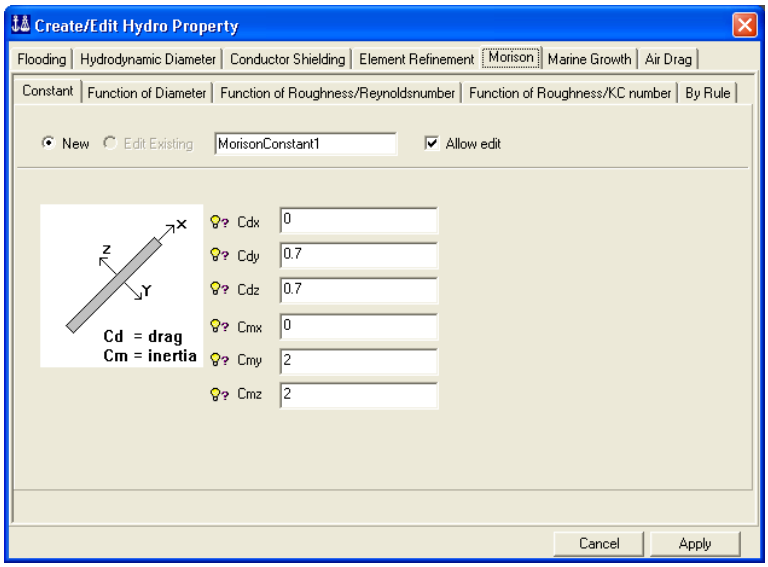


2.2.6.11.5 Morison

Purpose: To define constant Morison coefficients

This command is scripted. Typically:

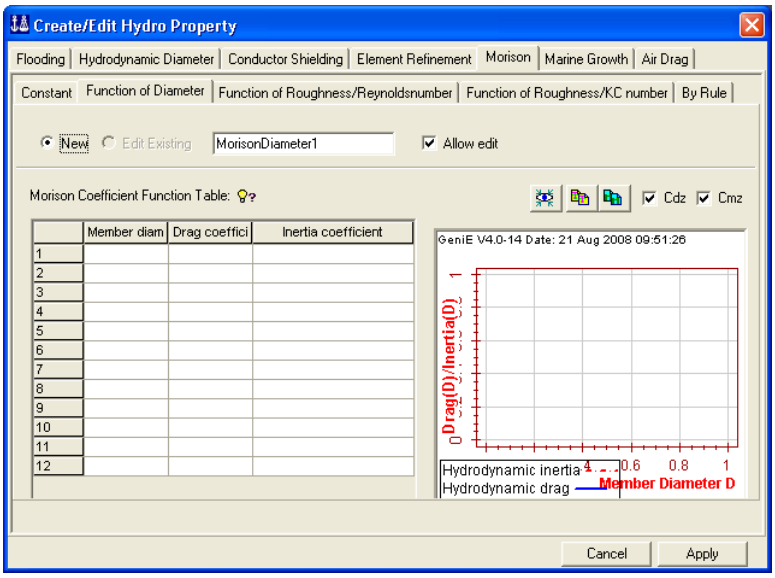
MyMorison = MorisonCoefficients(0,0.7,0.7,0,2,2);



Purpose: To define Morison coefficients as a function of diameter

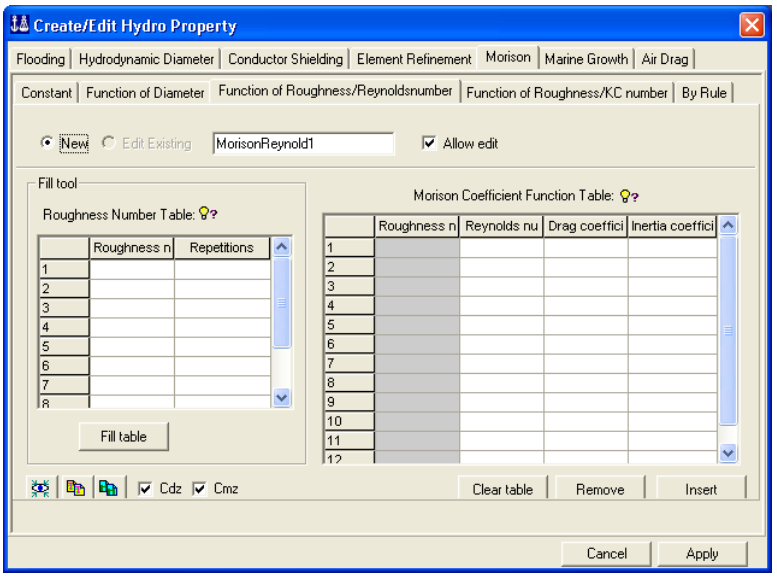
This command is scripted. Typically (for 2 different diameters):

MyMorisonDiameter = MorisonDiameterFunction(Array(2,3), Array(1,0.9),Array(2,2.1));



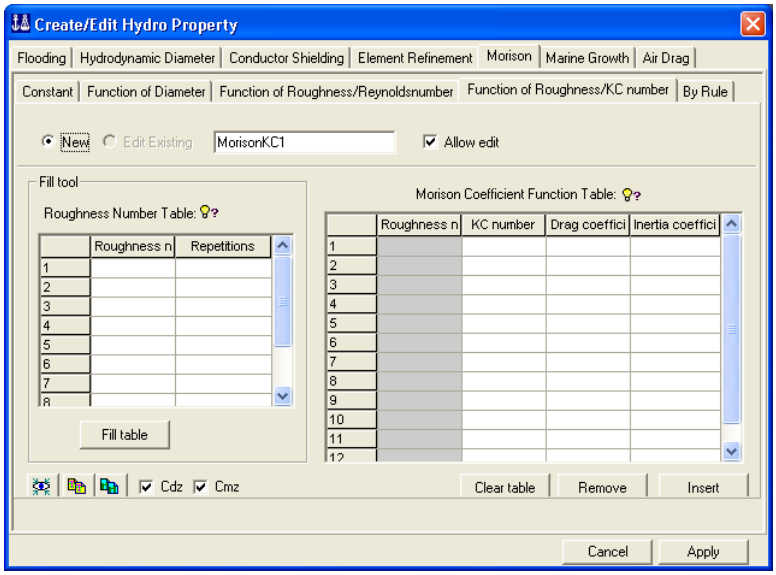
Purpose: To define Morison coefficients as a function of Roughness and Reynolds number

This command is scripted.



Purpose: To define Morison coefficients as a function of Roughness and Keulegan Carpenter (KC) number

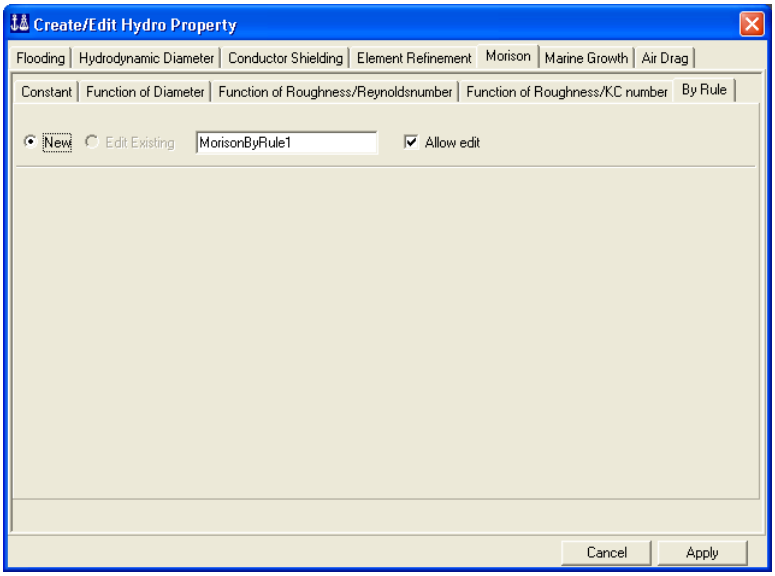
This command is scripted.



Purpose: To define Morison coefficients according to the API recommendations. See Wajac User Manual for a further definition

This command is scripted. Typically:

MyAPIMorison = MorisonByRule();

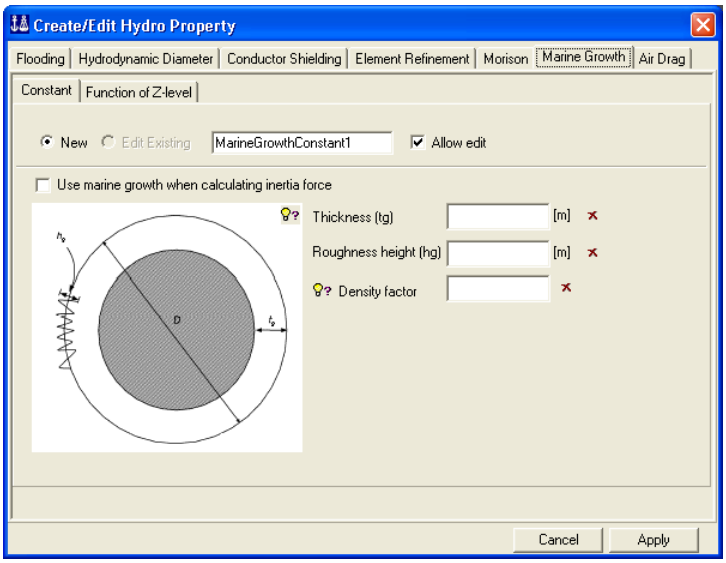


2.2.6.11.6 Marine growth

Purpose: To define a constant value for marine growth

This command is scripted. Typically:

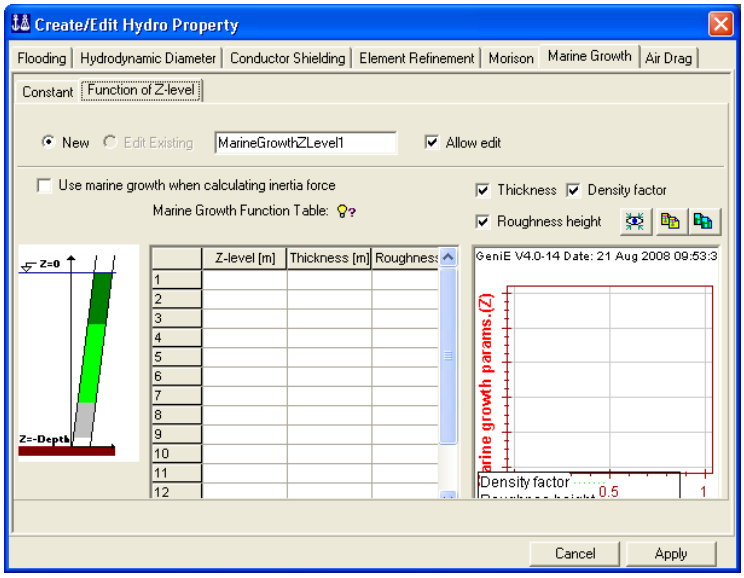
MyMarineGrowth = MarineGrowthConstant(20cm,1cm,2);
MyMarineGrowth.use InForceCalculations(false);



Purpose: To define a linearly varying value for marine growth

This command is scripted. Typically:

MyMarineVarGrowth = MarineGrowthZLevelFunction();
MyMarineVarGrowth.add (0m,0.2m,0.03m,2);
MyMarineVarGrowth.add (-50m,0.1m,0.02m,1);
MyMarineVarGrowth.use InForceCalculations(false);

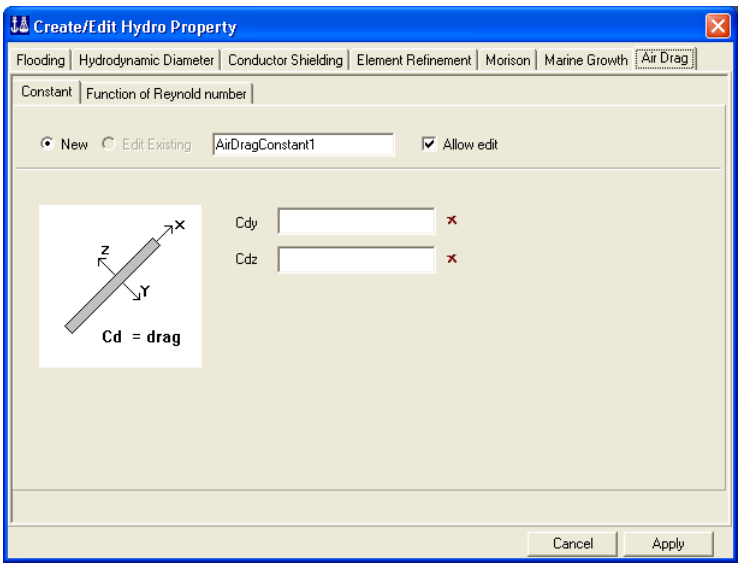


2.2.6.11.7 Air drag

Purpose: To define a constant value for air drag

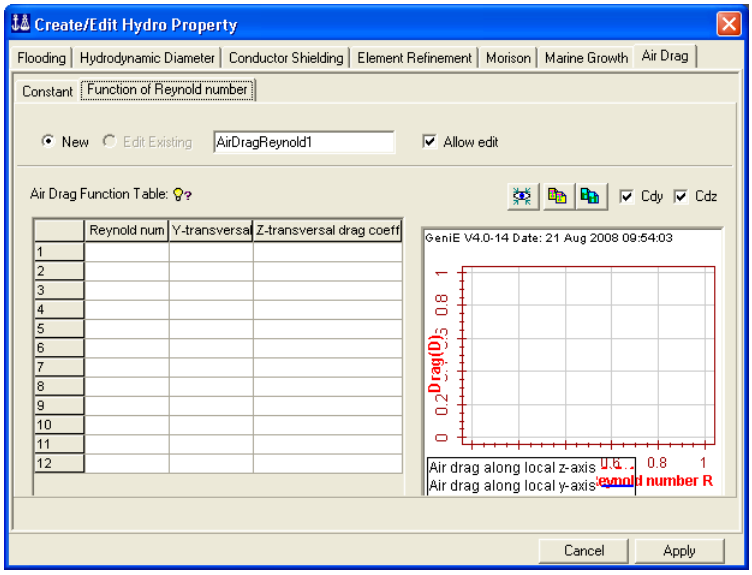
This command is scripted. Typically:

MyAirDrag = AirDragConstant(1,0.65);



Purpose: To define a value for air drag as function of the Reynold number.

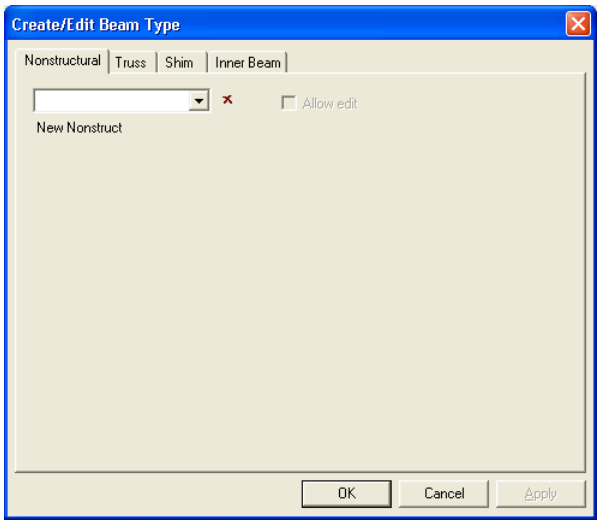
This command is scripted.



2.2.6.12 Beam Type

Purpose: To define various beam classifications to enable pre-defined rules.

2.2.6.12.1 Nonstructural

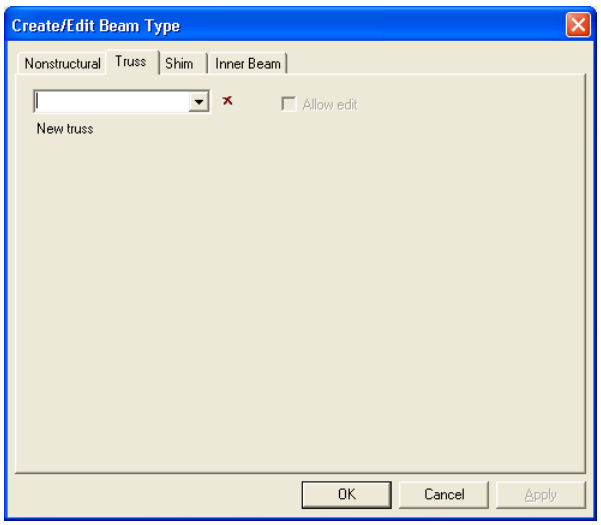


Purpose: To classify a beam as non-structural in a Morison analysis. The beam will contribute to the wave load analysis but not in a structural analysis.

This command is scripted. Typically:

MyNonStruct = BeamTypeNonstruct();

2.2.6.12.2 Truss

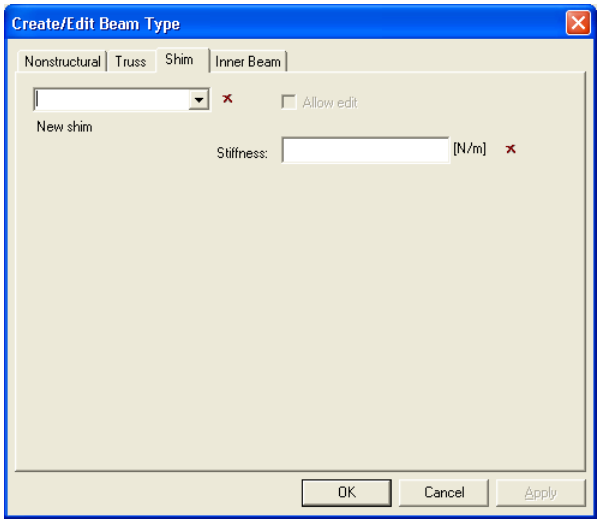


Purpose: To classify a beam as truss member. In this case the bending stiffness of the member is neglected.

This command is scripted. Typically:

MyTruss = BeamTypeTruss();

2.2.6.12.3 Shim



Purpose: To classify a beam as shim member. The shim elements are normally special connections between conductors and conductor frames whereby the conductor is free to move in vertical direction.

This command is scripted. Typically:

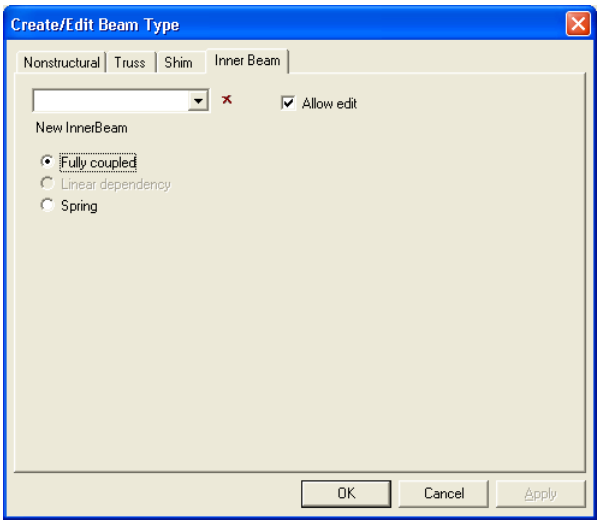
MyShim = BeamTypeShim(1250);

2.2.6.12.4 Inner beam

Purpose: To specify the connections between a beam and an inner beam.

This command is scripted. Typically:

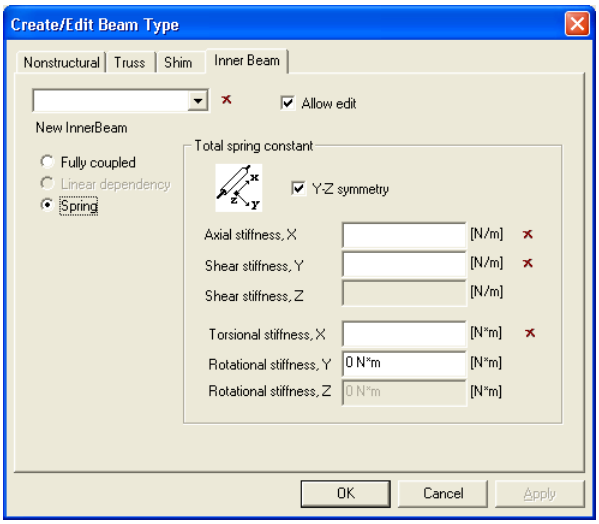
MyFullCoupling = InnerBeamSolid();



It is also possible to specify a spring connection between the beams.

This command is scripted. Typically:

*MySpringConnection = InnerBeamSpring(250,350,350,2500,0 kN*m,0 kN*m);*

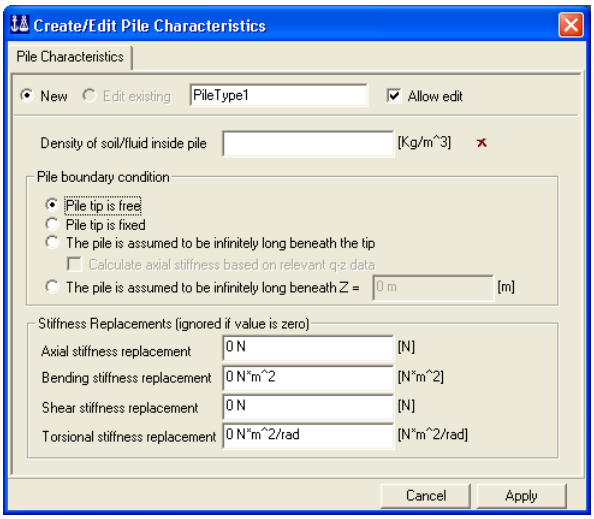


2.2.6.13 Pile Characteristics

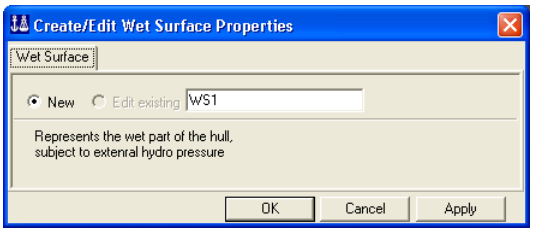
Purpose: To define specific pile properties to be used in a non-linear pile – soil analysis.

This command is scripted. Typically:

MyPileChar = PileCharacteristics(2.5, tcFree);



2.2.6.14 Wet Surface

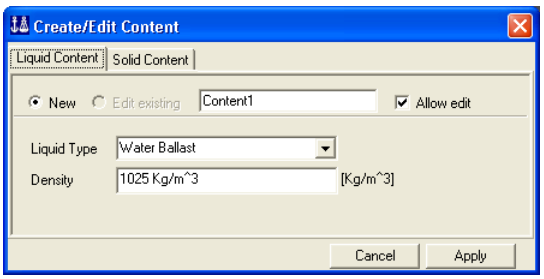


Purpose: To define a wet surface to be used by a panel model (outside wetted surface and compartment definitions) and for load application to plates and shells.

This command is scripted. Typically:

MyWetSurface = WetSurface();

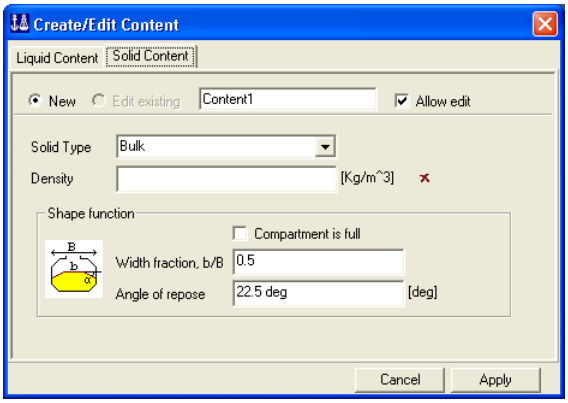
2.2.6.15 Content



Purpose: To define a liquid or solid content for use when filling a compartment.

This command is scripted. Typically:

MyLiquidContent =
LiquidContent(ctFuelOil, 900 tonne/m^3);



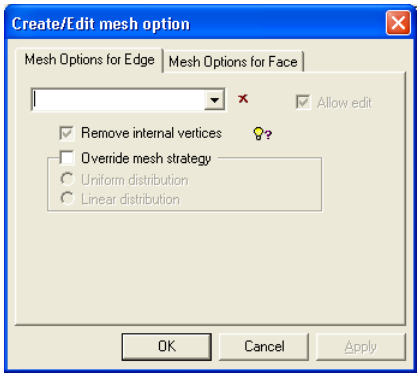
It is also possible to define a solid content.

This command is scripted. Typically:

MySolidContent = SolidContent(ctBulk, 2500,
ContentCSRBulk(0.5, 22.5 deg));

2.2.6.16 Mesh Option

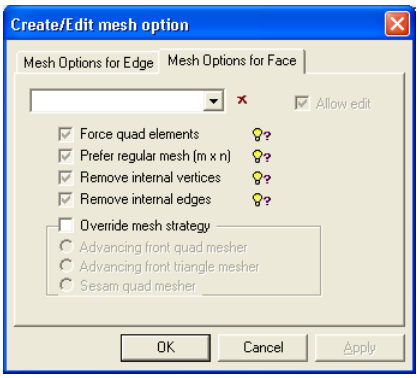
Purpose: To define a property that over-rules the default mesh settings.



This command is scripted. Typically for an edge:

MyMeshEdge = MeshOptionEdge();
MyMeshEdge.eliminateInternalVertices = true;
MyMeshEdge.meshStrategy = LinearDistributionEdge;

It is also possible to specify mesh options for a face.

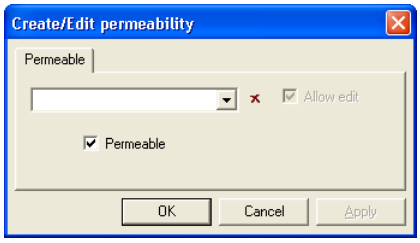


This command is scripted. Typically for a face:

MyMeshFace = MeshOptionFace();
MyMeshFace.forceQuadElements = true;
MyMeshFace.eliminateInternalEdges = true;
MyMeshFace.eliminateInternalVertices = true;
MyMeshFace.meshStrategy = AdvancingFrontQuadMesher;

2.2.6.17 Permeable

Purpose: To define a plate or shell as permeable even though the plate or shell is continuous (i.e. no holes). This is typically used when defining compartments. Loads are applied to the plate or shell.

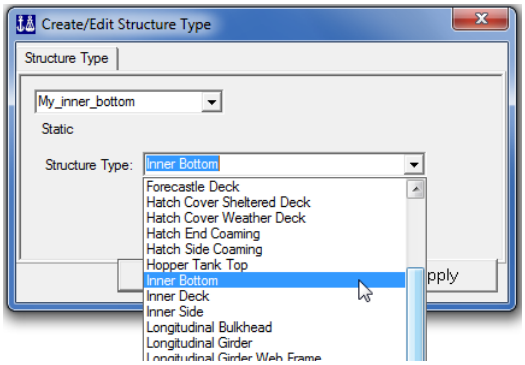


This command is scripted. Typically:

MyPermeable = Permeable(true);

2.2.6.18 Structure type

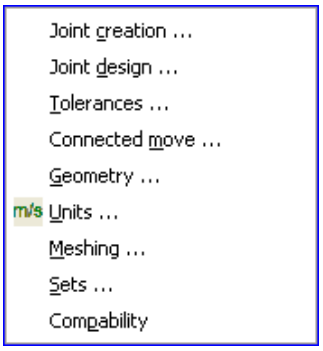
Purpose: To define or edit a structure type. Typically the structure type is applied to a part of the structure. When running a codecheck it is sometimes necessary to apply different structure types to different parts of the structure.



This command is scripted. Typically:
My_inner_bottom = StructureType(stInnerBottom);

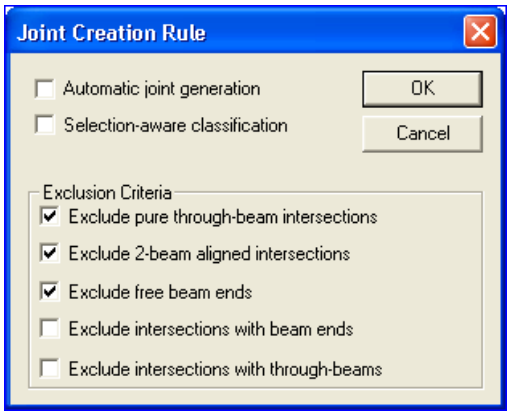
2.2.7 Rules

Purpose: Change or edit the rules for creation of joints, automatic design of tubular joints, tolerant modelling, connected move of beams, definition of groups (not sets), input units, meshing, scripting of sets and compatibility checking of log files between various program versions. Each option is described in the following:



2.2.7.1 Joint creation

Purpose: Rules for automatic creation of joints. The joints will be defined and named according to the rules specified as well as the naming convention.



This command is scripted. Typically, when activating the automatic joint generation and the exclusion criteria as shown:

GenieRules.JointCreation.exclude(geFreeBeamEnds,true);

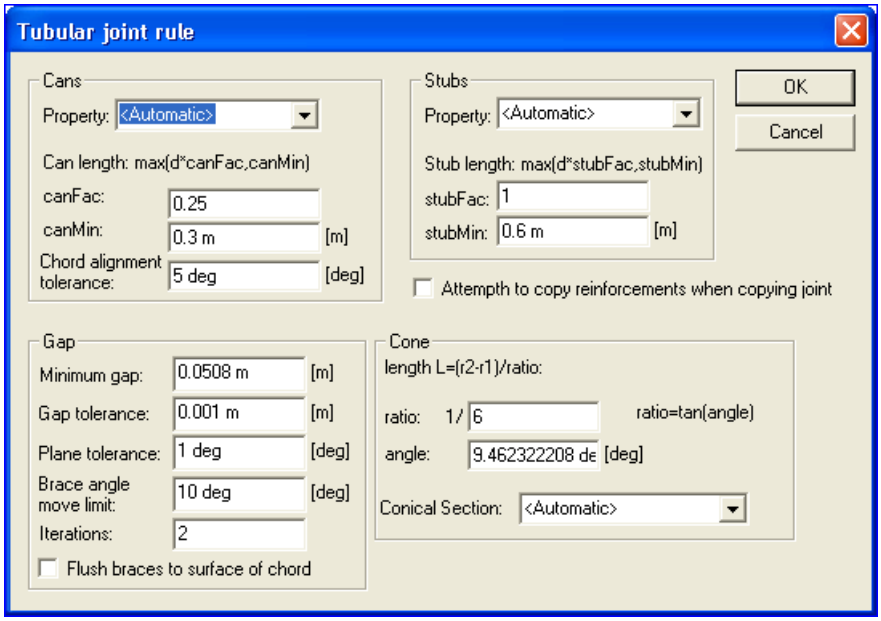
GenieRules.JointCreation.exclude(ge2BeamAligned,true);

GenieRules.JointCreation.exclude(geThroughBeamPure,true);

GenieRules.JointCreation.autoGenerate = true;

2.2.7.2 Joint design

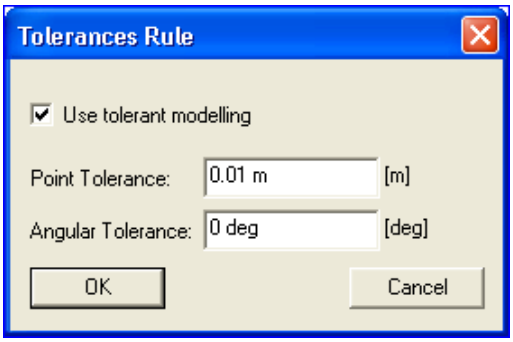
Purpose: Rules for calculating cans, stubs, cones and planewise gap in a tubular joint.



Command	Script
canFac “0.3” and canMin “0.4m”	*GenieRules.JointDesign.setDefaultCanRule(0.3,0.4 m);*
Chord alignment tolerance “3 deg”	*GenieRules.JointDesign.chordAlignmentTolerance = 3 deg;*
stubFac “0.9” and stubMin “0.5m”	*GenieRules.JointDesign.setDefaultStubRule(0.9,0.5 m);*
Minimum gap “0.075 m”	*GenieRules.JointDesign.minimumGap = 0.075 m;*
Gap tolerance “0.002 m”	*GenieRules.JointDesign.gapTolerance = 0.002 m;*
Plane tolerance “2 deg”	*GenieRules.JointDesign.planeTolerance = 2 deg;*
Brace angle move limit “11 deg”	*GenieRules.JointDesign.braceAngleMoveLimit = 11 deg;*
Iterations “3”	*GenieRules.JointDesign.iterations = 3;*
Flush braces to surface of chord	*GenieRules.JointDesign.flushBraces = true;* ***or*** *false*
Cone ratio 1/”8”	*GenieRules.JointDesign.coneAngle = 7.125016349 deg;*
Cone angle “10 deg”	*GenieRules.JointDesign.coneAngle = 10 deg;*

2.2.7.3 Tolerances

Purpose: Set tolerances when working with tolerant modelling.



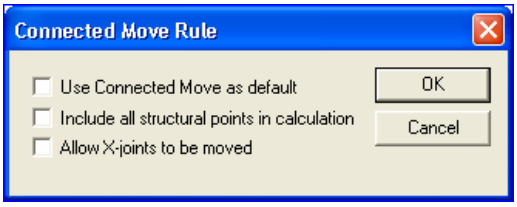
This command is scripted. Typically, when modifying the point tolerance to 0.05 m and angular tolerance to 3 degrees:

GenieRules.Tolerances.pointTolerance = 0.05;

GenieRules.Tolerances.angleTolerance = 3 deg;

2.2.7.4 Connected move

Purpose: Rules for connected move of beams.



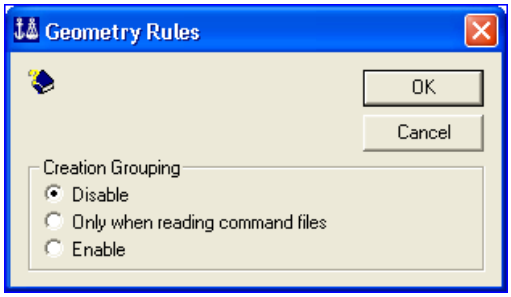
This command is scripted. Typically, when setting connected move as default and including all structural points in calculation:

GenieRules.ConnectedMove.defaultConnected = true;

GenieRules.ConnectedMove.useStructuralPoints = true;

2.2.7.5 Geometry

Purpose: To specify whether GeniE shall make groups when e.g. copying or moving structural members. This can lead to a better performance when reading a command file or during modelling. The command must be used with care.



This command is scripted. Typically, when enabling grouping when reading a command file:

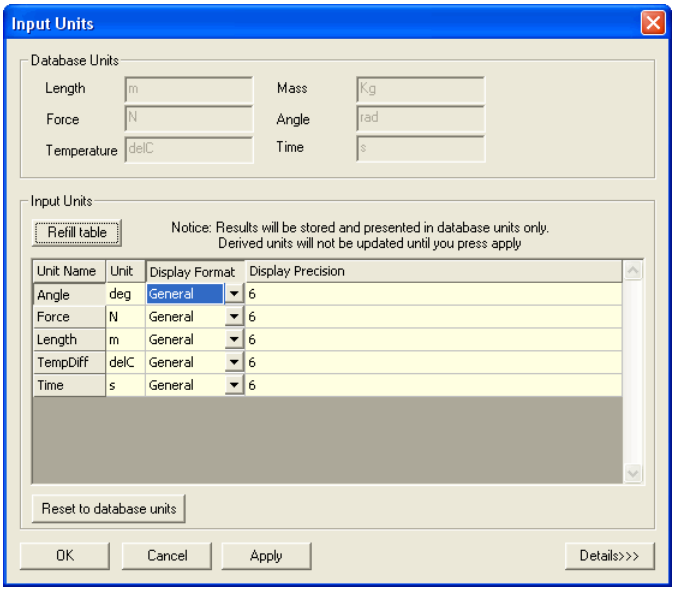
GenieRules.Geometry.creationGrouping = cgGroupingJournal;

2.2.7.6 Units

Purpose: Specify default input units and how labels etc, will appear on the desktop.

This command is scripted. Typically, when changing input unit to mm with fixed format display:

GenieRules.Units.setInputUnit (Length,"mm");
GenieRules.Units.setDisplay Format(Length,6,mcFixed);

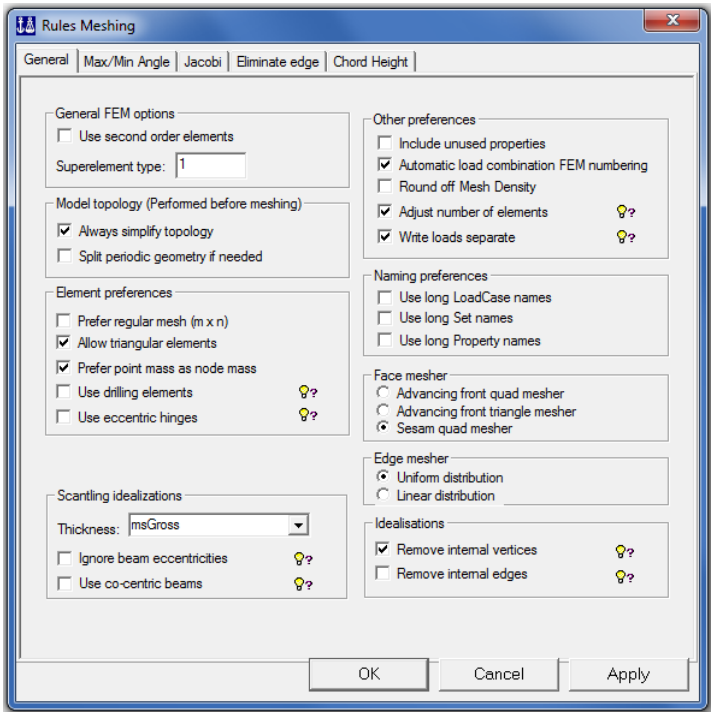


2.2.7.7 Meshing

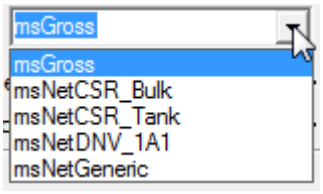
Purpose: Set global rules for creation of mesh. These may be override by local settings.

There are five categories of global mesh settings. For guidance in how these are used, please see Chapter 6 of User Manual Volume 3.

This Section lists the commands and the associated script command.



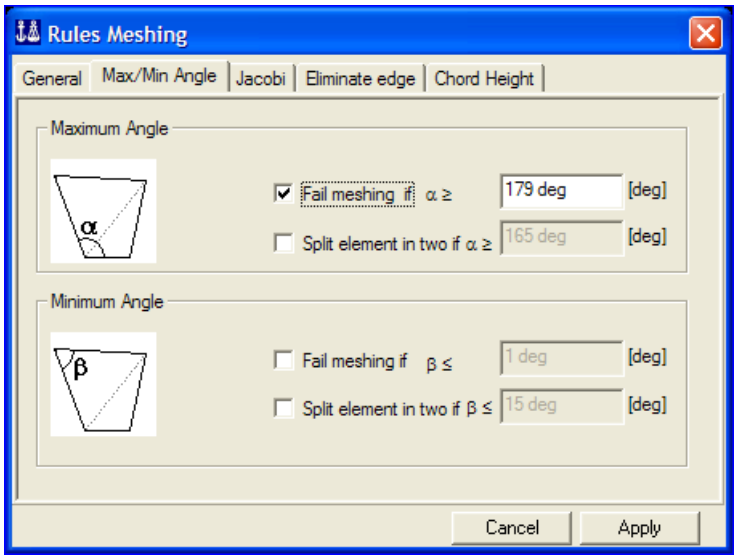
2.2.7.7.1 General

Command	Script
Use second order elements	*GenieRules.Meshing.elementType = mp2ndOrder;* ***or*** *mp1stOrder;*
Specify superelement type to "2"	*GenieRules.Meshing.superElementType = 2;*
Always simplify topology	*GenieRules.Meshing.autoSimplifyTopology = true;* ***or*** *false*
Split periodic geometry if needed	*GenieRules.Meshing.autoSplitPeriodicGeometry = true;* ***or*** *false*
Prefer rectangular mesh (m x n)	*GenieRules.Meshing.preference(mpPreferRectangularMesh, true);* ***or*** *false*
Allow triangular mesh	*GenieRules.Meshing.preference(mpAllowTriangularElements, true);* ***or*** *false*
Prefer point mass as node mass	*GenieRules.Meshing.preference(mpPreferPointMassAsNodeMass, true);* ***or*** *false*
Use drilling elements	*GenieRules.Meshing.preference(mpUseDrillingElements, true);* ***or*** *false*
Use eccentric hinges	
Specify scantling idealization according to the choices in the combo box: 	*Examples* *GenieRules.Meshing.scantlings = msGross;* *GenieRules.Meshing.scantlings = msNetCSR_Bulk;*
Ignore eccentricities	*GenieRules.Meshing.ignoreEccentricities = true;);* ***or*** *false*
Use co-centric beams	*GenieRules.Meshing.useCocentricBeams = true;*
Include unused properties	*GenieRules.Meshing.preference(mpIncludeUnusedProperties, true);* ***or*** *false*
Automatic load combination FEM numbering	*GenieRules.Meshing.FemLoadcaseNumbering = mmCombinationsAutomatic;);* ***or*** *Manual*
Round off mesh density	*GenieRules.Meshing.preference(mpMeshDensityRounded, true);* ***or*** *false*
Adjust number of elements	*GenieRules.Meshing.preference(mpAdjustNumberOfElements, true);* ***or*** *false*
Write loads separate	*GenieRules.Meshing.preference(mpWriteLoadsSeparate, true);* ***or*** *false*
Use long LoadCase names	GenieRules.Meshing.preference(mpUseLongLoadcaseNames, true); ***or*** *false*
Use long Set names	GenieRules.Meshing.preference(mpUseLongSetNames, true); ***or*** *false*
Use long Property names	GenieRules.Meshing.preference(mpUseLongPropertyNames, true); ***or*** *false*

Advancing front quad mesher	*GenieRules.Meshing.faceMeshStrategy = AdvancingFrontQuadMesher;*
Advancing front triangle mesher	*GenieRules.Meshing.faceMeshStrategy = AdvancingFrontTriangleMesher;*
Sesam quad mesher	*GenieRules.Meshing.faceMeshStrategy = SesamQuadMesher;*
Uniform distribution	*GenieRules.Meshing.edgeMeshStrategy = UniformDistributionEdge;*
Linear distribution	*GenieRules.Meshing.edgeMeshStrategy = LinearDistributionEdge;*
Remove internal vertices	*GenieRules.Meshing.eliminateInternalVertices = true;* ***or*** *false*
Remove internal edges	*GenieRules.Meshing.eliminateInternalEdges = true;* ***or*** *false*

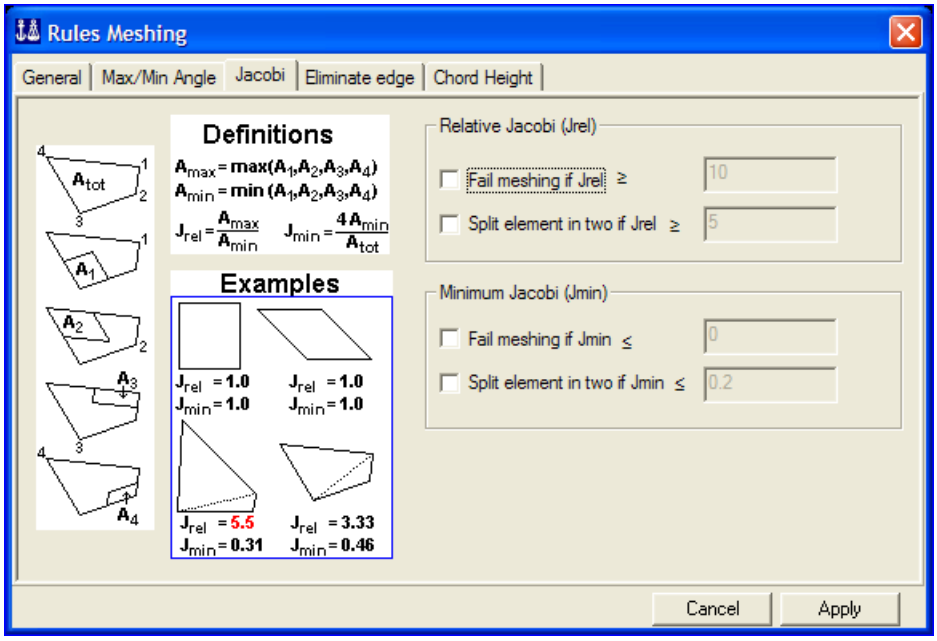
2.2.7.7.2 Max/min angle

Command	Script
Fail meshing if α ≥ "145"	*GenieRules.Meshing.activate(mpMinAngle,mpFail,true);* *GenieRules.Meshing.setLimit(mpMaxAngle,mpFail,145);*
Split element in two if α ≥ "130"	*GenieRules.Meshing.activate(mpMaxAngle,mpSplit,true);* *GenieRules.Meshing.setLimit(mpMaxAngle,mpSplit,130);*
Fail meshing if β ≤ "15"	*GenieRules.Meshing.activate(mpMinAngle,mpFail,true);* *GenieRules.Meshing.setLimit(mpMinAngle,mpFail,15);*
Split element in two if β ≤ "30"	*GenieRules.Meshing.activate(mpMinAngle,mpSplit,true);* *GenieRules.Meshing.setLimit(mpMinAngle,mpSplit,30);*



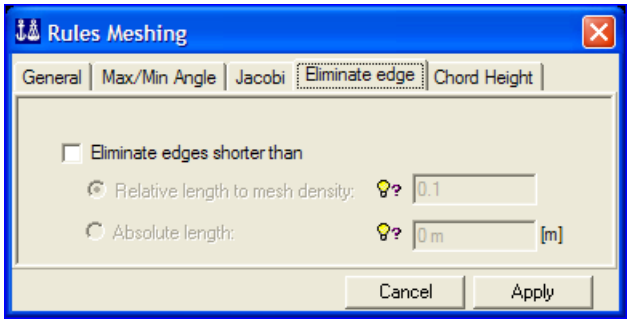
2.2.7.7.3 Jacobi

Command	Script
Fail meshing if Jrel ≥ "8"	*GenieRules.Meshing.activate(mpMaxRelativeJacobi,mpFail,true);* *GenieRules.Meshing.setLimit(mpMaxRelativeJacobi,mpFail,8);*
Split element in two if Jrel ≥ "6"	*GenieRules.Meshing.activate(mpMaxRelativeJacobi,mpSplit,true);* *GenieRules.Meshing.setLimit(mpMaxRelativeJacobi,mpSplit,6);*
Fail meshing if Jrel ≤ "0.2"	*GenieRules.Meshing.activate(mpMinNormalizedJacobi,mpFail,true);* *GenieRules.Meshing.setLimit(mpMinNormalizedJacobi,mpFail,0.2);*
Split element in two if Jrel ≤ "0.4"	*GenieRules.Meshing.activate(mpMinNormalizedJacobi,mpSplit,true);* *GenieRules.Meshing.setLimit(mpMinNormalizedJacobi,mpSplit,0.4);*



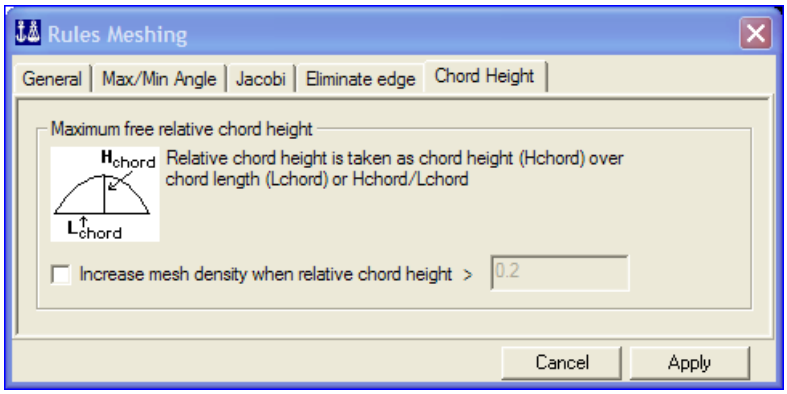
2.2.7.7.4 Eliminate edge

Command	Script
Relative length to mesh density "0.2"	*GenieRules.Meshing.activate(mpMinEdge,true);* *GenieRules.Meshing.setLimit(mpMinEdge,0.2);*
Absolute length "0.05 m"	*GenieRules.Meshing.activate(mpMinEdgeByLength,true);* *GenieRules.Meshing.setLimit(mpMinEdgeByLength,0.05m);*



2.2.7.7.5 Chord height

Command	Script
Increase mesh density when relative chord height > “0.25”	*GenieRules.Meshing.activate(mpMaxChord,true);* *GenieRules.Meshing.setLimit(mpMaxChord,0.25);*



2.2.7.8 Sets

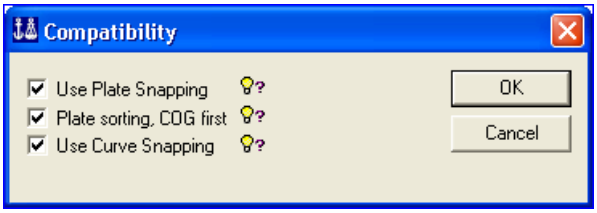
Purpose: Set rules for compact or verbose scripting of sets.

This command is scripted. Typically, when disabling compact scripting of sets:

GenieRules.Sets.scriptCompact = false;

2.2.7.9 Compability

Purpose: To specify criteria when reading in command files created in previous program releases.



Command	Script
Use Plate Snapping	*GenieRules.Compatibility.enable(PlateSnapping, true);* ***or*** *false*
Plate sorting, COG first	*GenieRules.Compatibility.enable(PlateSortingCOGFirst, true);* ***or*** *false*
Use Curve Snapping	*GenieRules.Compatibility.enable(CurveSnapping, true);* ***or*** *false*

2.2.8 Licenses/features

Here you can select which licenses should be used when GeniE is running on your computer. It can be useful if you are several persons sharing one license for one of the features listed.

2.2.8.1 CurvedGeometry

This should be checked if you are using curved geometry.

2.2.8.2 FrameCodecheck

This should be checked if you are using frame code checks.

2.2.8.3 PlateCodecheck

This should be checked if you are using plate code checks.

2.2.8.4 Enforce use of GeniE.lite license

This should be checked if you want to enforce using a GeniE.lite license, even if a full GeniE license is available.

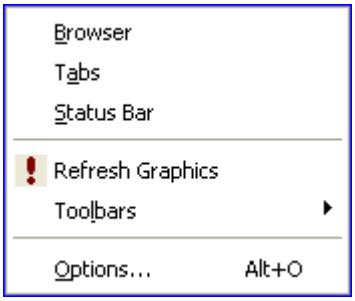
2.2.8.5 Do not show this dialog again

This should be checked to stop this dialog appearing each time you start GeniE.

2.3 The View pulldown menu

Here you can modify the settings of what you want to see, and you can define the graphics of what you want to see.

None of the commands are scripted. The actions are persistent, i.e. the changes are permanent for the ongoing and new jobs until you modify the view settings again.



2.3.1 View – Browser

Tick off for browser if you want to see it or if you want to remove it from your view

2.3.2 View – Tab

Tick off for viewing command lines interface windows or if you want to remove it

2.3.3 View – Status bar

Show or hide the status bar.

2.3.4 View – Refresh graphics

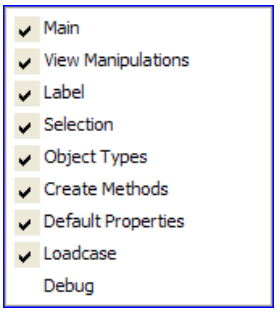
Redraws the current graphic views and remove any label.

2.3.5 View – Toolbars

Tick off option for customizing how many toolbars you want on the desktop.

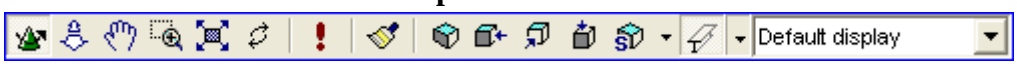
When an option is ticked off it is part of the desktop.

For a description of each toolbar menu, see following Chapters in this User Manual.

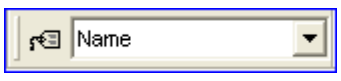


2.3.5.1 Main

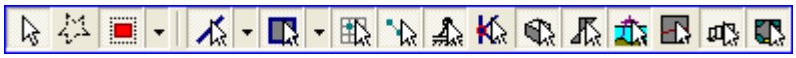
2.3.5.2 View manipulation



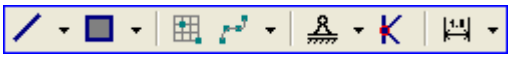
2.3.5.3 Label



2.3.5.4 Selection

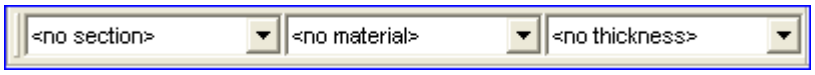


2.3.5.5 Object Types

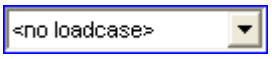


2.3.5.6 Create Methods

2.3.5.7 Default Properties



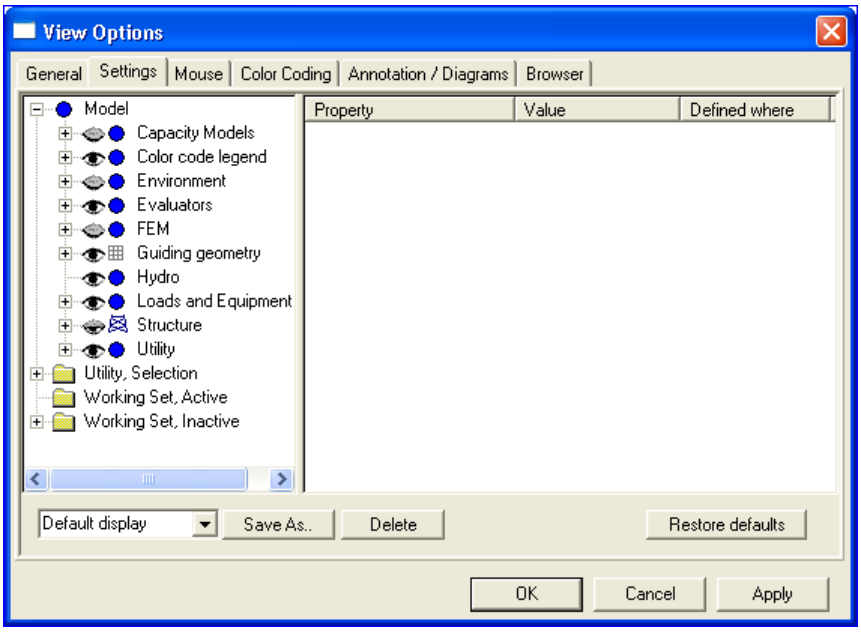
2.3.5.8 Loadcase



2.3.6 View – Options

Access to a number of options controlling the view (Display settings, Category settings, Cursor feedback, Load and result, Presentation). Consult User Manual Volume 1 for further details.

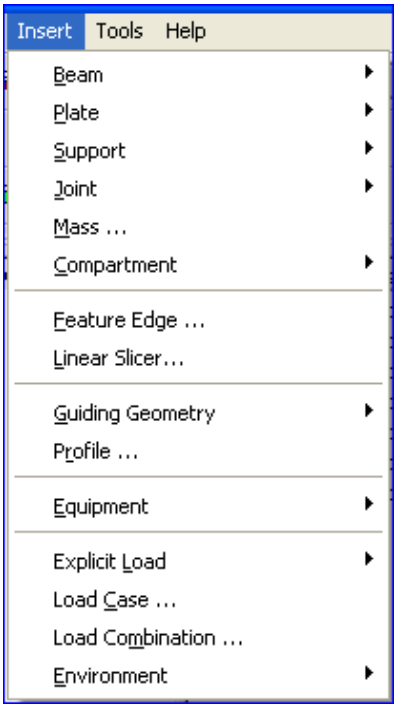
Most of the data specified under View|Options is stored in the Registry for the current GeniE version (in the user profile for the current user). The exception is the creation of colour coding maps which is logged to the journal file and stored in the program database.



2.4 The Insert pulldown menu

From the Insert pulldown menu you can define guiding geometry, feature edges, structural parts and loads (including equipments). In addition you can define compartments environments.

Each of these commands is listed in the following with a typical script command.

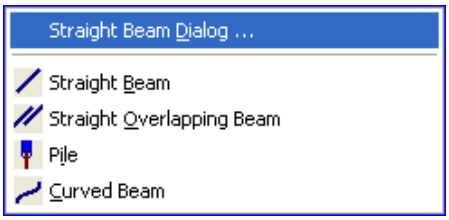


2.4.1 Beam

Here you can choose between different types of beams and specify the end points manually or snapping to points. See Section 3.4 of User Manual Volume 1 for details.

2.4.1.1 Straight Beam Dialog

Purpose: Insert a beam and specify end points manually. You can also specify section and material properties manually if these differ from default settings.

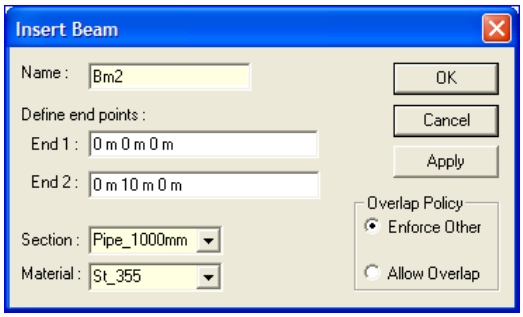


This command is scripted. Typically, when inserting Bm2 between (0, 0, 0) and (0, 10, 0) and using section type Pipe_1000mm and material type St_355:

Bm2 = Beam(Point(0 m,0 m,0 m),Point(0 m,10 m,0 m));

Bm2.Material = St_355;

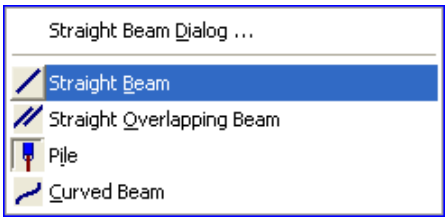
Bm2.Section = Pipe_1000mm;



See Chapter 3 for a definition of reference point modelling.

2.4.1.2 Straight Beam

Purpose: Insert a beam graphically by snapping to 2 points.

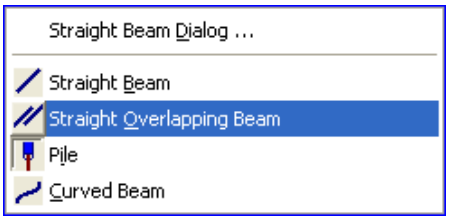


This command is scripted. Typically, when inserting Bm3 (automatically name) between Point1 and Point 2 when using reference point modelling:

Bm3 = Beam(Point1, Point2);

2.4.1.3 Straight Overlapping Beam

Purpose: To insert an additional beam between two points where there is a straight beam(s) from before.

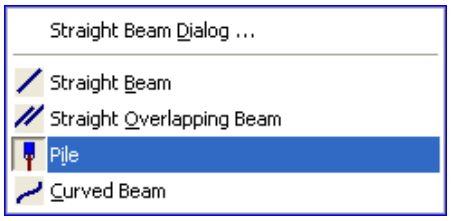


This command is scripted. Typically, when inserting Bm4 on top of another beam (automatically name) between Point1 and Point 2 when using reference point modelling:

Bm4 = Beam(Point1,Point2,geAllowOverlap);

2.4.1.4 Pile

Purpose: To insert a pile (a special variant of a beam concept) whereby the top coordinate is defined by a point, the bottom is defined by an elevation and the direction of a guiding beam. The pile may be a combination of a straight beam and an overlapping beam (i.e. the pile is inside a pile guide or a leg).

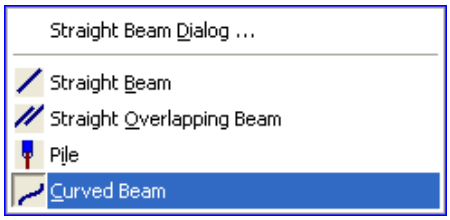


This command is scripted. Typically, when inserting Pile1 from Point1 and down to elevation z=-50m using the direction of Bm3 (reference point modelling has been used):

Pile1 = Pile(Point1,GuideLine(Point1, Point1 + Bm3.localSystem.xVector, 0).intersect(ZPlane3d(-50m)));

2.4.1.5 Curved Beam

Purpose: Insert a curved beam graphically by snapping to 3 points or more



This command is scripted. Typically, when inserting Bm6 by referring to Point1, Point2 and Point3 (shown with both reference point modelling as well as regular coordinate modelling):

Bm6 = CurvedBeam(Point1,Point2,Point3);

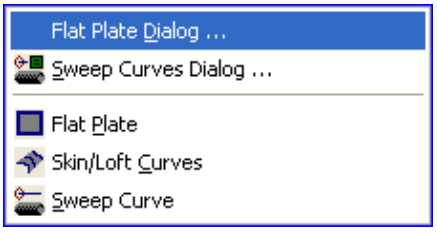
Bm6 = CurvedBeam(Point(10 m,10 m,10 m),Point(10 m,20 m,20 m),Point(10 m,10 m,30 m));

2.4.2 Plate

From the insert plate menu you can insert plates and shells using different modelling techniques. For more details see Section 3.5 of User Manual Volume 1 and Section 3.3 of User Manual Volume 3.

2.4.2.1 Flat Plate Dialog

Purpose: Insert a plate and specify corner points manually.

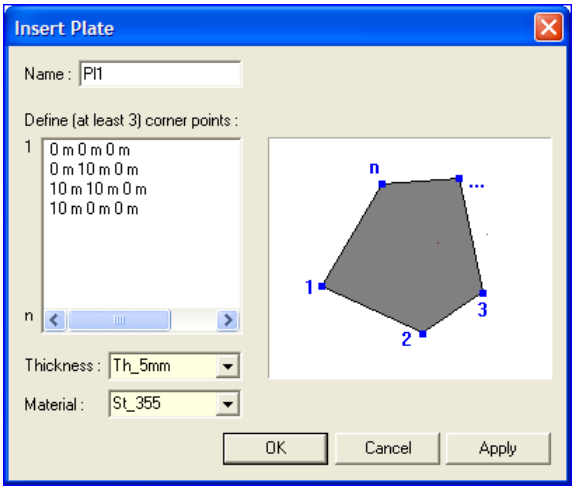


This command is scripted. Typically, when inserting Pl1 using the coordinate values as shown to the right:

Pl1 = Plate(Point(0 m,0 m,0 m),Point(0 m,10 m,0 m), Point(10 m,10 m,0 m),Point(10 m,0 m,0 m));

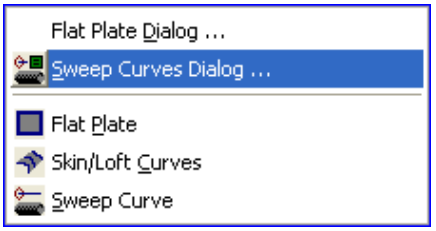
Pl1.Material = St_355;

Pl1.Thickness = Th_5mm;



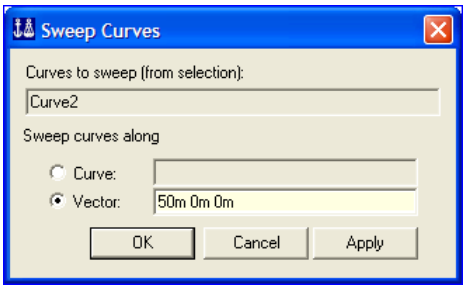
2.4.2.2 Sweep Curves Dialog

Purpose: To sweep (or extrude) a curve(s) along another curve or vector to form a shell.

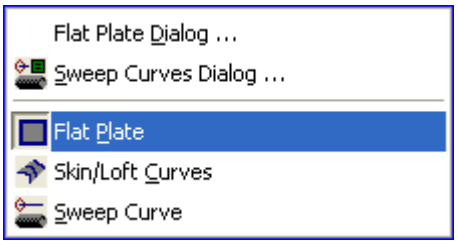


This command is scripted. Typically, when inserting Pl2 by sweeping Curve2 50 m in x-direction:

Pl2 = SweepCurve(Curve2, Vector3d(50m,0m,0m));



2.4.2.3 Flat Plate

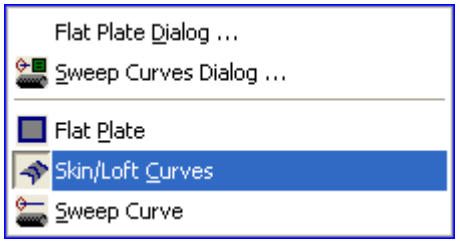


Purpose: Insert a plate graphically by snapping to 4 points or more. A triangular plate is made when 1st and 4th snap point is the same.

This command is scripted. Typically, when inserting Pl3 that has 5 corners (notice that 1st and 6th snap point must be the same – remember to use the tool bar Snap Point Loop):

Pl3 = Plate(Point(10 m,0 m,50 m),Point(10 m,8.75 m,50 m),Point(8.75 m,10 m,50 m),Point(3.75 m,7.5 m,50 m),Point(3.75 m,0 m,50 m));

2.4.2.4 Skin/Loft Curves



Purpose: To use a skin operation in between two curves or more. It is also possible to include existing shells or plates to guide the shell surface tangent (this is called lofting).

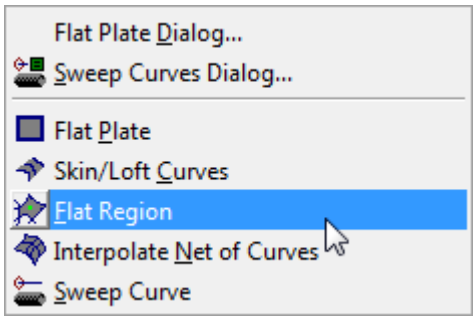
This command is scripted. Typically, when defining Pl4 from a skin operation between Curve1, Curve2 and Curve3:

Pl4 = SkinCurves(Array(Curve2, Curve3, Curve4));

Similarly, when using plate Pl2 as guidance to control the initial surface of Pl4 (this is a lofting operation):

Pl4 = LoftCurves(Pl2, Array(Curve2, Curve3, Curve4));

2.4.2.5 Flat region

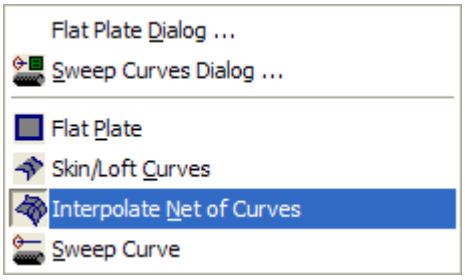


Purpose: To create a flat region consisting of one or several plates. The region is created based on a plane and existing structure forming one or several enclosed areas.

This command is scripted. Typically, when defining Pl5 from a plane at z=4 m, using the area enclosed by the plates Pl1, Pl2, Pl3 and Pl5:

Pl5 = Plate(ZPlane3d(4 m), Array(Pl1, Pl2, Pl3, Pl4),
Point(1.5 m, 0.6 m, 4 m));

2.4.2.6 Curve-Net Interpolation



Purpose: To construct a surface that interpolates two arrays of N and M curves, which form a set of (N-1)x(M-1) rectangular patches. Each patch is formed by any two consecutive curves of both arrays.

This command is scripted. Typically, when defining P15 from such an operation between the curve arrays Array(Curve1, Curve2, Curve3) and Array(Curve4, Curve5):

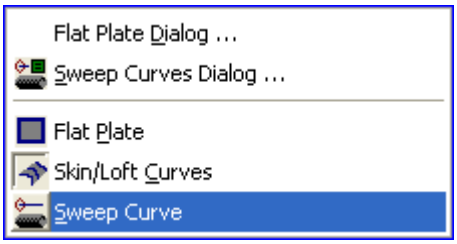
Pl5 = InterpolateCurveNet(Array(Curve1, Curve2, Curve3), Array(Curve4, Curve5));

Note that one of the curves may be a single point, provided that the same intersection restrictions hold for this point as well. For instance the command

Pl5 = InterpolateCurveNet(Array(Point1,Curve1, Curve2, Curve3), Array(Curve4, Curve5));

will give a valid result, provided that Point1 is the intersection point Curve4 and Curve5, thus forming a triangular patch (which is a degenerate rectangular patch).

2.4.2.7 Sweep Curve



Purpose: To insert a plate or shell by graphically referring to two curves. The first curve defines the width of the object while the second curve specifies the direction.

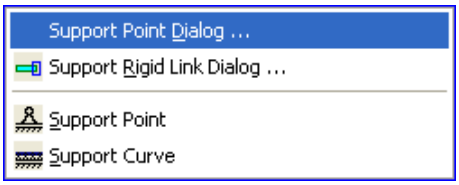
This command is scripted. Typically, when defining Pl5 by referring to Curve2 and Curve 5:

Pl5 = SweepCurve(Curve2, Curve5);

2.4.3 Support

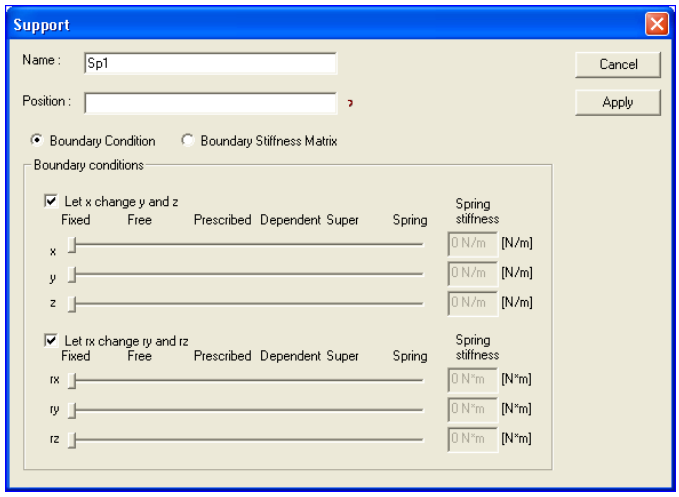
Here you can insert support and specify the support conditions graphically or manually

See Section 3.12 of User manual Volume 1 and Chapter 5 of User Manual Volume 3 for more details on how to define support conditions.



2.4.3.1 Support Point Dialog

Purpose: Insert a support and specify location manually. You may choose between a boundary condition fixed, free, prescribed (used by a load case), dependent (used together with rigid link support), super (used when defining a superelement) and a spring. It is also possible to define a boundary stiffness matrix.

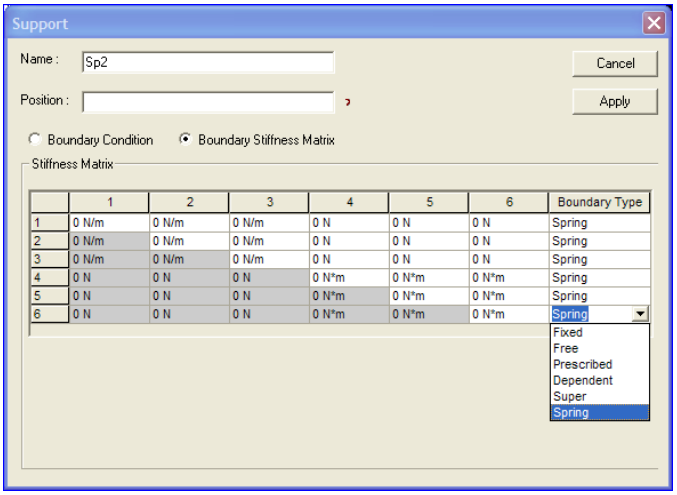


This command is scripted. Typically, when defining a boundary condition Sp1 with fixed translations and free rotations:

Sp1 = SupportPoint(Point(15,0,0));

Sp1.boundary = BoundaryCondition(Fixed, Fixed, Fixed, Free, Free, Free);

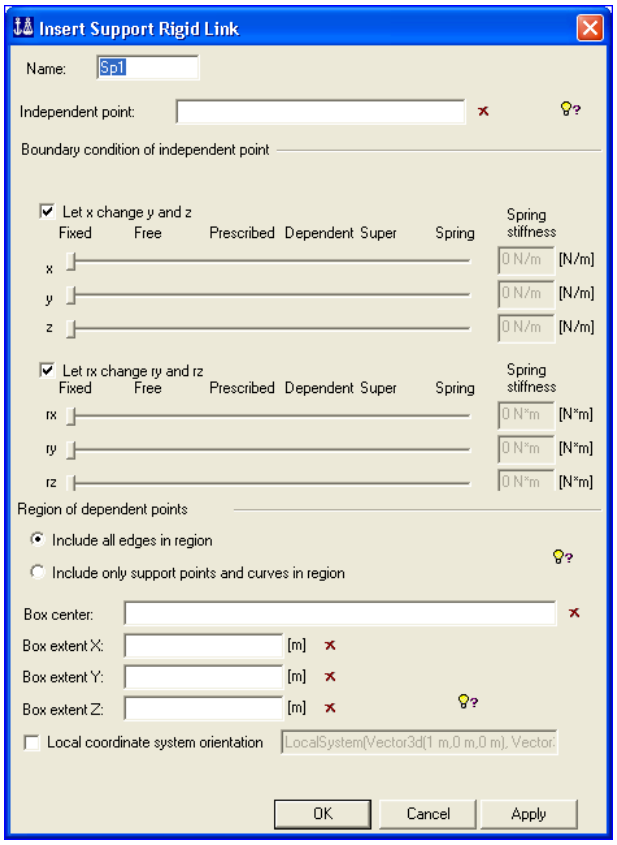
Alternatively when using the boundary stiffness matrix option to define Sp2. In this case values along the diagonal have been inserted:



*Sp2.boundary = BoundaryStiffnessMatrix(Stiffness(100 N/m), Stiffness(200 N/m), Stiffness(300 N/m), Stiffness(4000 N*m), Stiffness(5000 N*m), Stiffness(6000 N*m));*

2.4.3.2 Rigid Link Dialog

Purpose: To insert boundary conditions so that they are dependent upon a point.



This command is scripted. Typically, when defining a rigid link Sp1 where the input parameters are:
Independent point is Point7
- Box centre is Point8
- Box extent is δx = 0.2m, δy = 0.6m and δz = 1m
- Include all edges in region
- Boundary condition: all degrees of freedom are dependent

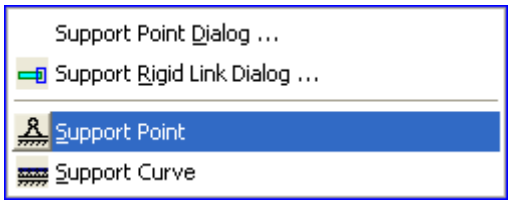
Sp6 = SupportRigidLink(Point7, FootprintBox(Point8, Vector3d(0.2,0.6,1), LocalSystem(Vector3d(1 m,0 m,0 m), Vector3d(0 m,0 m,1 m))));

Sp6.includeAllEdges = true;

Sp6.boundary = BoundaryCondition(Dependent, Dependent, Dependent, Dependent, Dependent, Dependent);

2.4.3.3 Support Point

Purpose: Insert a support point graphically by snapping to 1 point. Per default the support point will be fixed in all six degrees of freedom. To modify these you need to select the support point(s) and edit the conditions.

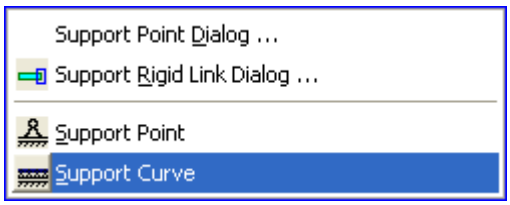


This command is scripted. Typically, when defining a support point at (0m, 0m, 0m) and modify the rotational degrees of freedom

Sp1 = SupportPoint(Point(0 m,0 m,0 m));

Sp1.boundary = BoundaryCondition(Fixed, Fixed, Fixed, Free, Free, Free);

2.4.3.4 Support Curve



Purpose: Insert support conditions along a line; curved or straight. Per default all degrees of freedom are fixed. When making a finite element model all finite element nodes belonging to the support curve will receive same boundary condition.

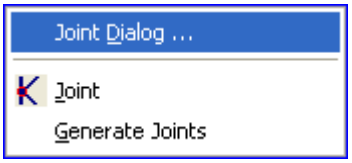
This command is scripted. Typically, when defining a support curve SC5 and modify some of the degrees of freedom:

Sc5 = SupportCurve(ModelCurve(Point(24 m,5 m,0 m), Point(24.5 m,5 m,0 m), Point(25 m,5.5 m,0 m)));

Sc5.localSystemRule = ConstantLocalSystem(LocalSystem(Vector3d(1 m,0 m,0 m), Vector3d(0 m,0 m,1 m)));

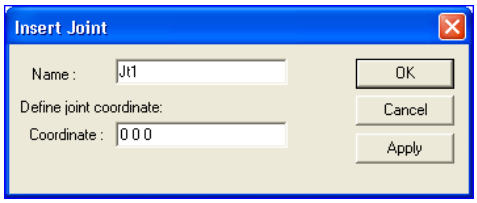
Sc5.boundary = BoundaryCondition(Fixed, Free, Free, Free, Fixed, Fixed);

2.4.4 Joint



Insert single joints manually or graphically. Alternatively by using the joint creation rules as defined from ***Edit|Rules|Joint Creation***. For more details on how to do it, see Section 3.6 of User Manual Volume 1.

2.4.4.1 Joint Dialog

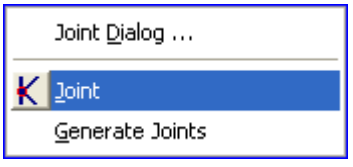


Purpose: Insert a joint by specifying its coordinates manually. Notice that a joint can only be created at a beam intersection.

This command is scripted. Typically, when defining a joint Jt1 at a beam intersection at (4m, 11.5m, 10m):

Jt1 = Joint(Point(4 m,11.5 m,10 m));

2.4.4.2 Joint



Purpose: Insert a joint by snapping to 1 point. Same conditions as above apply.

This command is scripted. Typically, when defining a joint Jt1 at a beam intersection at (4m, 11.5m, 10m):

Jt1 = Joint(Point(4 m,11.5 m,10 m));

2.4.4.3 Generate Joint

Purpose: Insert joints according to the joint creating rules by selecting a beam or beams. Number of joints to be defined depends on the filter settings defined in ***Edit|Rules|Joint Creation***.

This command is scripted. Typically, when referring to a beam with 6 intersections a total of 6 joints are created. In this case the rules include that joints shall be created at each beam intersection:

Jt2 = Joint(Point(0 m,28.5 m,10 m));

Jt3 = Joint(Point(4 m,28.5 m,10 m));

Jt4 = Joint(Point(12 m,28.5 m,10 m));

Jt5 = Joint(Point(18 m,28.5 m,10 m));

Jt6 = Joint(Point(24 m,28.5 m,10 m));

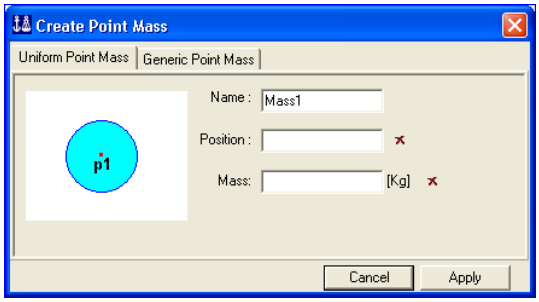
Jt7 = Joint(Point(27 m,28.5 m,10 m));

2.4.5 Mass

There are two alternatives for adding a specific point mass. They are described in the following. See Section 4.6.2 for further references.

2.4.5.1 Uniform Point Mass

Purpose: To define a same mass contribution in global x, y and z directions.

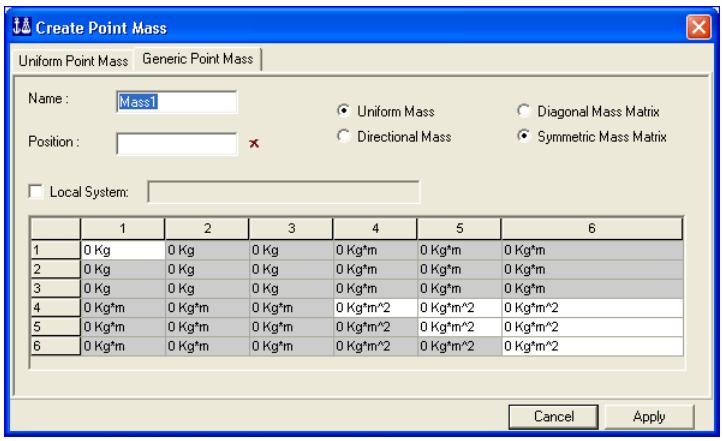


This command is scripted. Typically, when adding 1000 tonnes at a specific position:

Mass1 = PointMass(Point(10,10,10), 1000 tonne);

2.4.5.2 Generic Point Mass

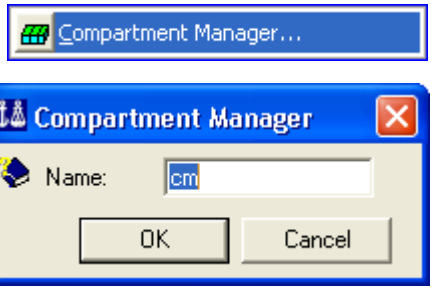
Purpose: To define a mass with different contribution in global translation and rotation x, y and z directions.



This command is scripted. Typically, when adding different mass elements using directional and a diagonal mass matrix:

*Mass2 = PointMassMatrix(Point(25,25,25), 5000 Kg, 4000 Kg, 3000 Kg, 10000 Kg*m^2, 8000 Kg*m^2, 6000 Kg*m^2);*

2.4.6 Compartment Manager

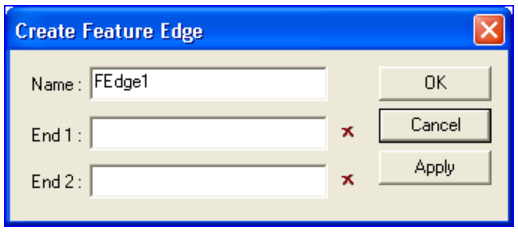


Purpose: To automatically create compartments for use when creating loads from content or to define a tank subjected for filling in HydroD. The compartment information is also used by Nauticus Hull to automatically generate rules based loading conditions according to the CSR Bulk Rules. All closed voids will form a compartment. For more details, see Section 4.4.1 of User Manual Volume 1.

This command is scripted. Typically:

My_Comp_Manager = CompartmentManager();

2.4.7 Feature Edge

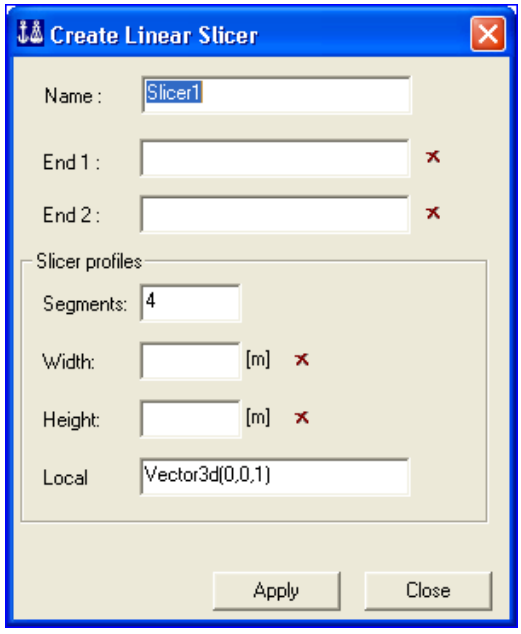


Purpose: To insert an edge (or line) for controlling the quality of your mesh. Consult Section 3.12.2 of User Manual Volume 1 or Chapter 6 of User Manual Volume 3 to see how this feature can be used.

This command is scripted. Typically:

FEdge1 = FeatureEdge(Point(24 m,5 m,1 m), Point(24 m,5. m,7 m));

2.4.8 Linear Slicer



Purpose: A linear slicer is used to calculate correction moment and shear forces to achieve target values according to CSR rules for bulk ships. The correction moments are automatically applied to the model. To do this it is necessary to first define the linear slicer and then select the slicer to see the moment and shear forces before and after corrections have been applied. This feature is further documented in the training material for Nauticus Hull.

The linear slicer can also be used to compute forces and moments along a defined direction for a given load case.

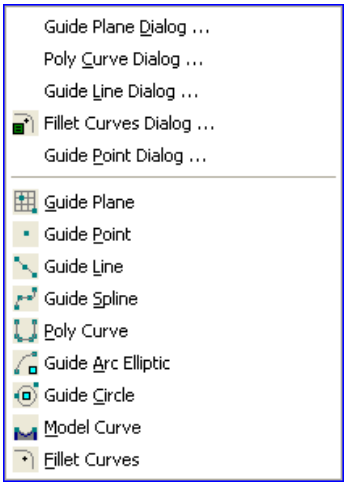
This command is scripted. Typically:

Slicer1 = LinearSlicer(Point(18 m,-1 m,10 m), Point(12 m,30 m,10 m), Vector3d(0,0,1), 50, 30);

Slicer1.segments = 10;

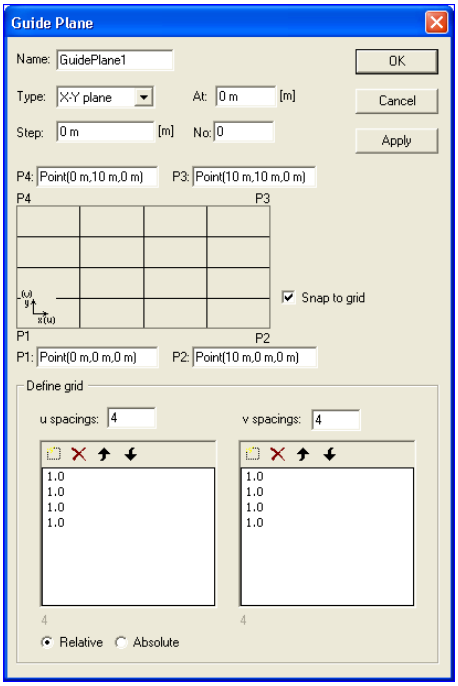
2.4.9 Guiding Geometry

Guiding geometry is often used as a starting position when creating structural parts like beams, plates, stiffeners and shells. There are a number of different guiding geometries that can be used. They are listed in the following. For a detailed explanation on how to create and use guiding geometries, reference is made to Section 3.3 of User Manual Volume 1 and Section 3.2 of User Manual Volume 3.



2.4.9.1 Guide Plane Dialog

Purpose: Insert a guide plane and specify corner points manually, elevations (x, y, or z), length of spacing (constant or varying) in two directions.



This command is scripted. Typically when using different relevant values for u (1, 2, 2, 1) and v (1, 2, 3, 4) and creating 3 guide planes simultaneously by using a step length of 5 meters 2 times:

GuidePlane1 = GuidePlane(Point(0 m,0 m,0 m),Point(10 m,0 m,0 m),Point(10 m,10 m,0 m),Point(0 m,10 m,0 m),4,4,1,2,2,1,1,2,3,4);

GuidePlane1.snapmode = true;

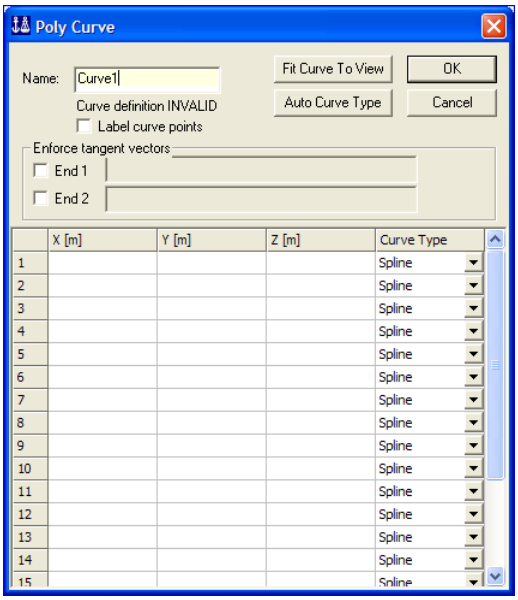
GuidePlane2 = GuidePlane(Point(0 m,0 m,5 m),Point(10 m,0 m,5 m),Point(10 m,10 m,5 m),Point(0 m,10 m,5 m),4,4,1,2,2,1,1,2,3,4);

GuidePlane2.snapmode = true;

GuidePlane3 = GuidePlane(Point(0 m,0 m,10 m),Point(10 m,0 m,10 m),Point(10 m,10 m,10 m),Point(0 m,10 m,10 m),4,4,1,2,2,1,1,2,3,4);

GuidePlane3.snapmode = true;

2.4.9.2 Poly Curve Dialog



Purpose: Insert a guide curve and specify coordinates manually. The options "Enforce tangent vectors" is used to control the vectors of the poly curve at the start and end position.

This command is scripted. Typically when defining 6 reference points to construct the poly curve using the spline option only and controlling the line tangent at start (direction 1,1,0) and end (direction 0,1,0) positions:

Curve1 = PolyCurve();

Curve1.clear();

Curve1.addPoint(Point(1, 1, 1), ggSpline);

Curve1.addPoint(Point(2, 2, 2), ggSpline);

Curve1.addPoint(Point(3, 3, 3), ggSpline);

Curve1.addPoint(Point(4, 4, 4), ggSpline);

Curve1.addPoint(Point(2, 4, 2), ggSpline);

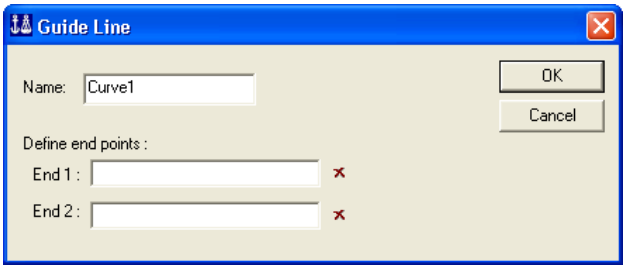
Curve1.addPoint(Point(1, 5, 7), ggSpline);

Curve1.startDeriv(Vector3d(1,1,0));

Curve1.endDeriv(Vector3d(0,1,0));

Curve1.rebuild();

2.4.9.3 Guide Line Dialog



Purpose: Insert a straight guide line and specify end points manually

This command is scripted. Typically between the two points:

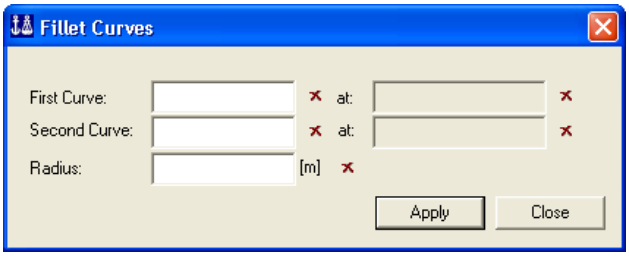
Curve1 = GuideLine(Point(1,1,1), Point(2,2,2), 4);

Notice that the number "4" in the expression above is used to divide the guide line into 5 equal parts so that there are 4 internal snap points in addition to the 2 end snap points. This means that if you use "6" the guide line will have 6 internal snap points. To do this you can write in the command input window (remember to click the Enter button):

Curve1 = GuideLine(Point(1,1,1), Point(2,2,2), 6);

2.4.9.4 Fillet Curves Dialog

Purpose: Fillet curves are used to define the curvature between two straight lines. When fillet curves are used, the two straight lines in question and the fillet curves are automatically joined to a composite curve.

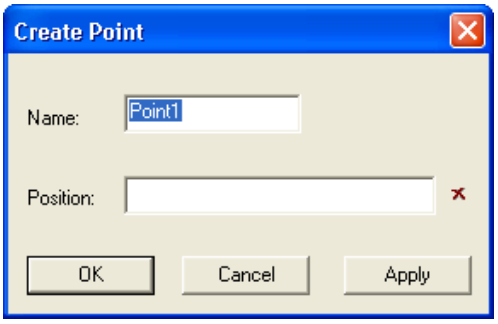


This command is scripted. Typically when selecting two curves and specifying a radius of 2 m. The coordinate values are automatically proposed by GeniE):

Curve4 = filletCurves(Curve2, Point(3 m,0 m,0 m), Curve3, Point(5 m,1 m,0 m), 2);

2.4.9.5 Guide Point Dialog

Purpose: Insert a guide point and specify coordinates manually.

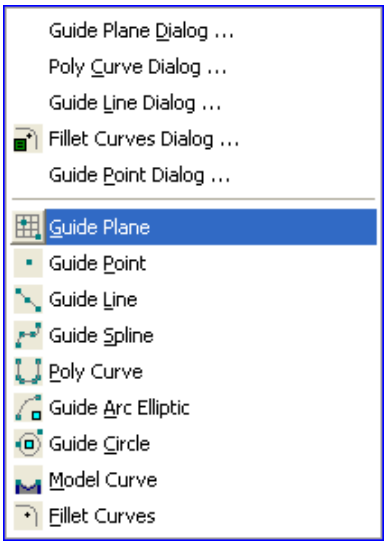


This command is scripted. Typically:

Point1 = Point(7.5 m,5 m,0 m);

2.4.9.6 Guide Plane

Purpose: Insert a guide plane graphically by snapping to 4 points.
Per default the guide plane will receive 4 equal spacing lengths in u and v directions.

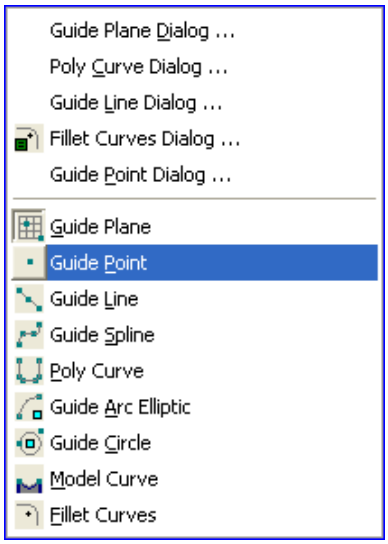


This command is scripted. Typically:

GuidePlane3 = GuidePlane(Point(10 m,0 m,0 m),Point(10 m,10 m,0 m),Point(10 m,10 m,10 m),Point(10 m,0 m,10 m),4,4,1,1,1,1,1,1,1,1);

2.4.9.7 Guide Point

Purpose: Insert a guide point graphically by snapping to 1 point.

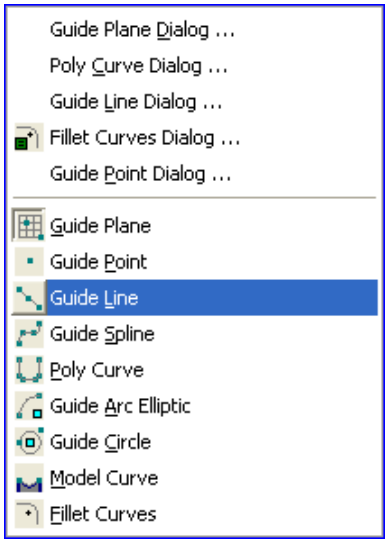


This command is scripted. Typically:

Point3 = Point(2.5 m,0 m,10 m);

2.4.9.8 Guide Line

Purpose: Insert a straight guide line graphically by snapping to 2 points.



This command is scripted. Typically (notice that the default value for internal snap points is "3"):

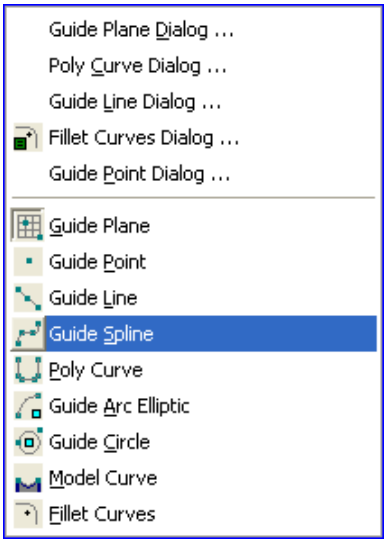
Curve5 = GuideLine(Point(0 m,0 m,0 m), Point(2.5 m,2.5 m,0 m), 3);

If you want to change number of internal snap points to 4 you can type the following in the command line window:

Curve5 = GuideLine(Point(0 m,0 m,0 m), Point(2.5 m,2.5 m,0 m), 4);

2.4.9.9 Guide Spline

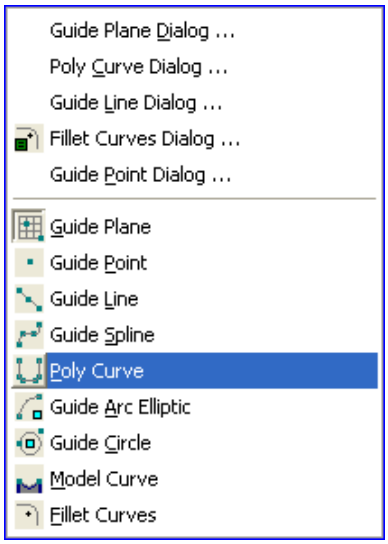
Purpose: Insert a guide spline graphically by snapping to 3 points or more.



This command is scripted. Typically when using 5 snap points (remember to double click the final snap point):

Curve12 = GuideSpline(Array(Point(5 m,0 m,0 m),Point(7.5 m,0 m,0 m),Point(10 m,2.5 m,0 m),Point(10 m,5 m,2.5 m),Point(10 m,5 m,5 m)),3);

2.4.9.10 Poly Curve



Purpose: Insert a poly curve graphically by snapping to 3 points or more.

This command is scripted. Typically when using 6 snap points (remember to double click the final snap point):

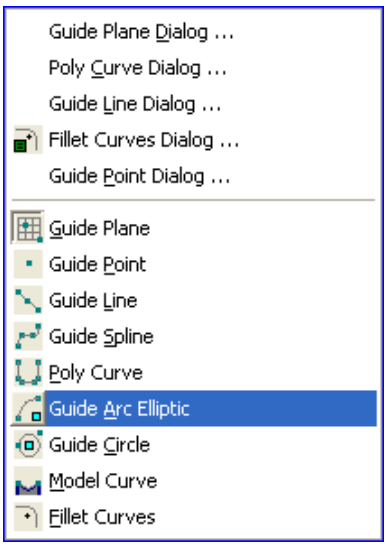
Curve14 = PolyCurve(Array(Point(5 m,10 m,10 m), Point(7.5 m,10 m,10 m), Point(10 m,7.5 m,10 m), Point(10 m,5 m,10 m), Point(7.5 m,2.5 m,10 m), Point(5 m,2.5 m,10 m)));

Curve14.modifyCurveType(2, ggStraight);

Curve14.modifyCurveType(4, ggStraight);

Notice that in this case there are two parts that are straight as a result of an automatic curve fitting process. This can be changed from editing the poly curve.

2.4.9.11 Guide Arc Elliptic



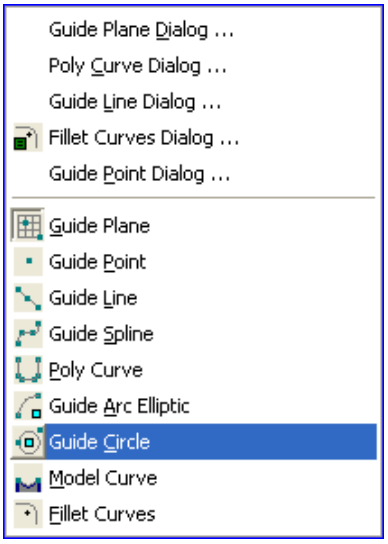
Purpose: Insert a guide arc elliptic graphically by snapping to 3 points (origin, start and end of ellipse)

This command is scripted. Typically:

Curve20 = GuideArcElliptic(Point(10 m,2.5 m,7.5 m), Point(10 m,5 m,5 m), Point(10 m,2.5 m,10 m), true);

2.4.9.12 Guide Circle

Purpose: Insert a guide circle graphically by snapping to 3 points (origin, radius, plane). The plane is a result of the three points.

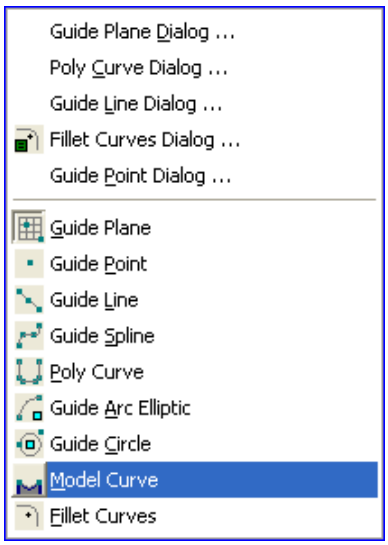


This command is scripted. Typically:

Curve24 = GuideCircle(Point(10 m,2.5 m,2.5 m), Point(10 m,5 m,2.5 m), Point(10 m,0 m,5 m));

2.4.9.13 Model Curve

Purpose: Insert a model curve by snapping between two points. The model curve will follow the surface curvature (if any) between the two points. Per default there are 3 internal snap points- these can not be altered.

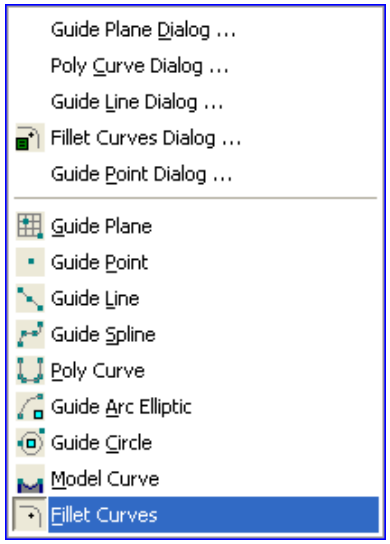


This command is scripted. Typically:

Curve25 = ModelCurve(Array(Point(5 m,2.5 m,10 m),Point(6.327271284 m,2.174330704 m,14.9859724 m),Point(7.5 m,2.5 m,20 m)),3);

2.4.9.14 Fillet Curves

Purpose: To create a fillet curve by clicking on two adjacent lines. GeniE will propose a minimum fillet radius in between; this can be altered by typing a new radius followed by Enter.

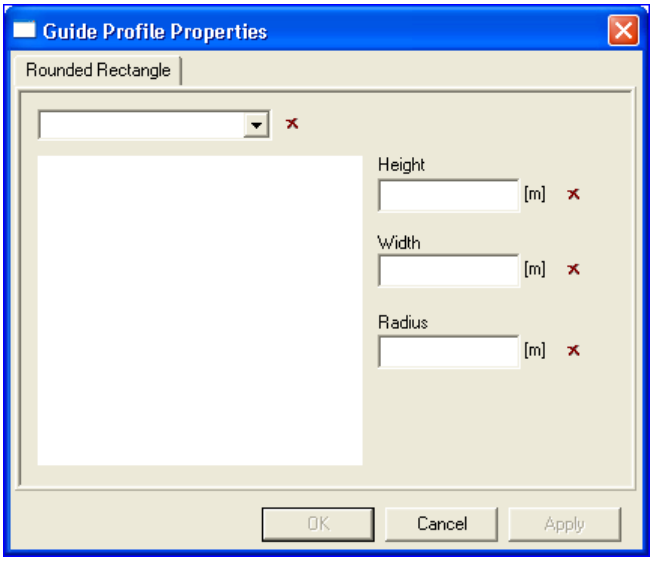


This command is scripted. Typically when using a radius of 1 m:

Curve29 = filletCurves(Curve10, Point(5.688202143 m,5.688202143 m,0 m), Curve4, Point(5 m,4.148269176 m,0 m), 1);

2.4.10 Profile

Purpose: Insert a profile to be used for punching or cut operation (from ***Tools|Structure|Punch***). See Section 3.6.7 of User Manual Volume 1 for details on how to use this feature.

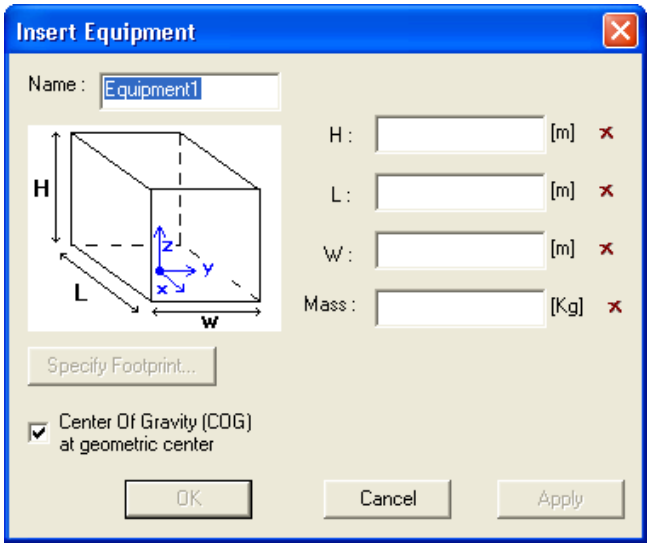


This command is scripted. Typically:

Manhole = ProfileRR(2m,1m,0.25m);

2.4.11 Equipment

Purpose: Define prismatic equipment for use in load cases to generate masses or loads. For further details see Section 3.9 of User Manual Volume 1.

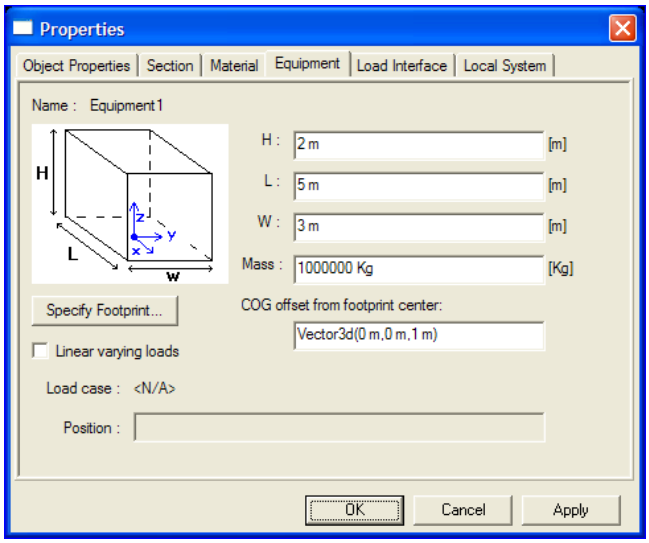


This command is scripted. Typically with height 2m, length 5m, width 3m and mass 1000 tonne:

Equipment1 = PrismEquipment(5m,3m,2m,1000 tonne);

2.4.11.1 COG offset

Purpose: To define the COG manually relative to the footprint centre. The COG may also be outside the equipment box. Define the equipment, select the equipment from the browser, RMB and choose Properties to edit the equipment.



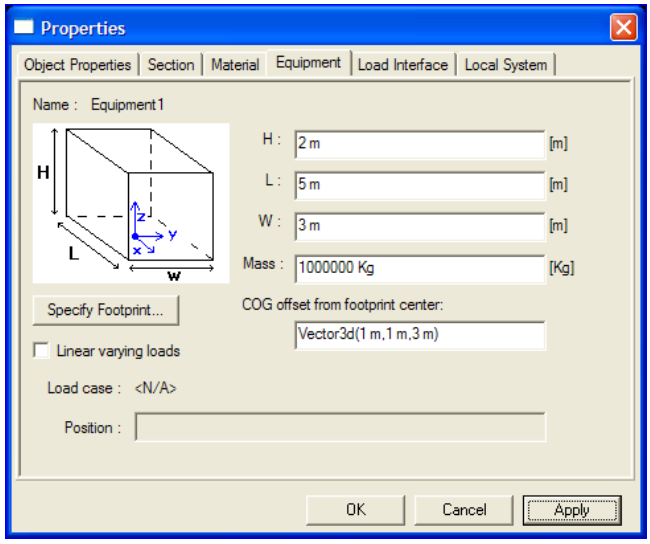
This command is scripted. Typically when defining the COG position at (1m, 1m, 3m):

Equipment1.centreOfGravity (Vector3d(1m,1m,3m));

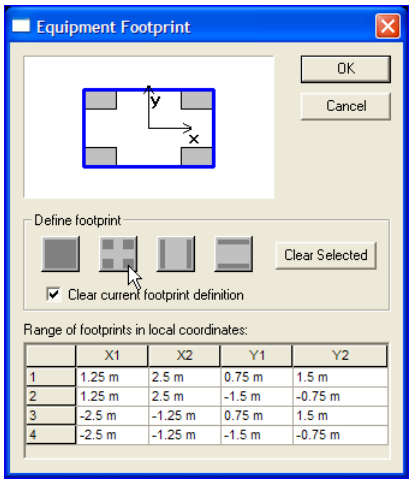
2.4.11.2 Specify footprint

Purpose: To define the footprint of the equipment. There are 4 templates that can be edited. The footprint may be outside the equipment border. Define the equipment, select the equipment from the browser, RMB and choose Properties to edit the equipment. When you click on the tab Specify Footprint there are 4 templates available.

There may be more than 4 footprints.



This command is scripted and dependent on which template is chosen and the size of each footprint. Typically:



Equipment1.clearFootprint();

Equipment1.addToFootprint(1.25 m, 2.5 m, 0.75 m, 1.5 m);

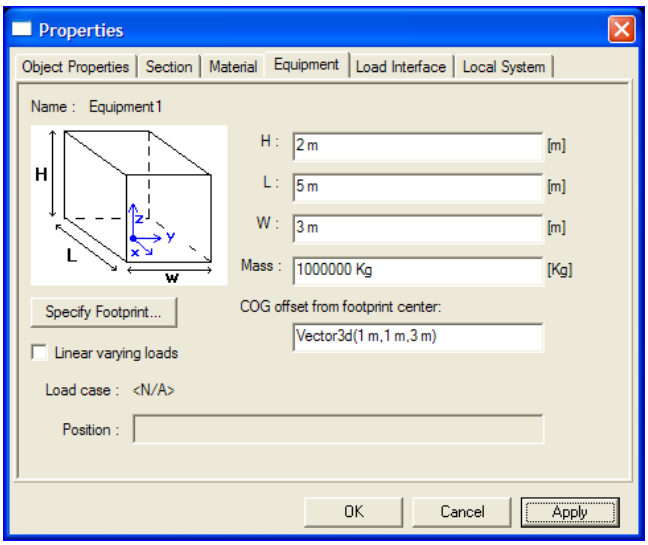
Equipment1.addToFootprint(1.25 m, 2.5 m, -1.5 m, -0.75 m);

Equipment1.addToFootprint(-2.5 m, -1.25 m, 0.75 m, 1.5 m);

Equipment1.addToFootprint(-2.5 m, -1.25 m, -1.5 m, -0.75 m);

2.4.11.3 Linear varying loads

Purpose: To specify whether generated loads shall be constant or linear (linear is the default option). It is necessary to place the equipment in a load case before this can be modified from linear to constant.



This command is scripted. Typically when altering to constant loads or linear for the equipment Helideck in load case LC_heli:

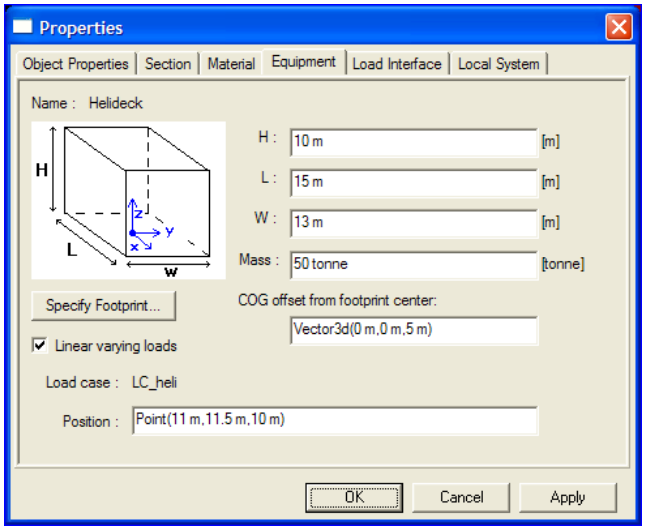
LC_heli.setCurrent();

LC_heli.constantLoad(Helideck);

LC_heli.varyingLoad(Helideck);

2.4.11.4 Position

Purpose: To specify the exact location of the equipment. The position denotes the location of the centre of the equipment bottom plane. It is necessary to place the equipment in a load case before the position can be modified.



This command is scripted. Typically when modifying the location to (15m, 20m, 12m):

LC_heli.placeAtPoint(Helideck, Point(15m,20m,12m), LocalSystem(Vector3d(1 m,0 m,0 m), Vector3d(0 m,0 m,1 m)));

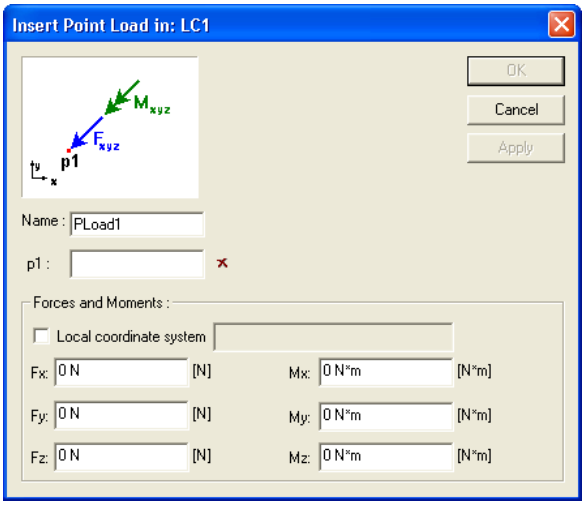
2.4.12 Explicit Load

Purpose: To Insert and define an explicit load like point load, line load, surface load, prescribed displacement and line temperature. A load case must be defined and set to current before any loads can be applied. For more details on how to apply loads, reference is made to Section 3.11 of User Manual Volume 1 and Chapter 4 of User Manual Volume 3.

2.4.12.1 Point Load

Purpose: To define a point load. This load can only be applied along a beam to give effect. The load is independent of the structure meaning that it does not follow when the relevant beam is moved or copied.

The point load may be defined according to the global coordinate system (default) or a local coordinate system.

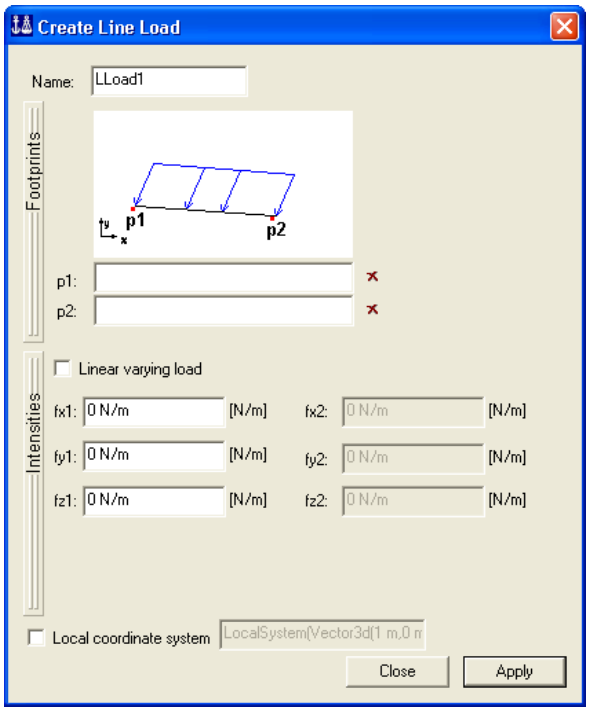


This command is scripted. Typically:

*PLoad1 = PointLoad(LC1, Point(4 m,5 m,10 m), 100 kN, 200 kN, -300 kN, -1000 kN*m, -2000 kN*m, 3000 kN*m);*

2.4.12.2 Line Load

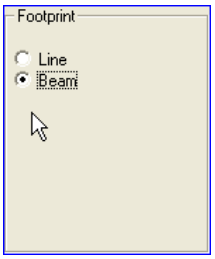
Purpose: Define a constant or linearly varying line load between two points along a beam. This load can only be applied along a beam to give effect. The load may be independent or dependent of the structure. To make the line load independent of the beam the load intensities relate to two end points. In this case the line load does not follow when moving or copying the beam.



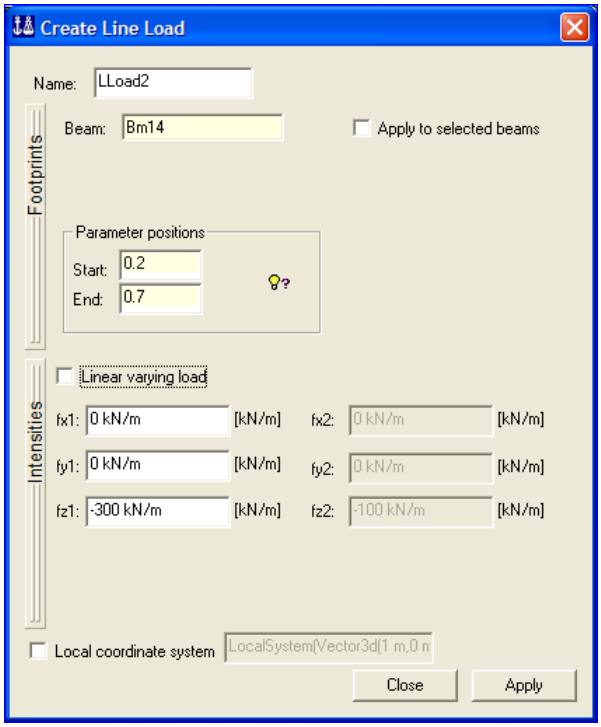
This command is scripted. Typically for a linear variation in z-direction:

LLoad1 = LineLoad(LC1, FootprintLine(Point(4 m,0 m,10 m), Point(12 m,0 m,10 m)), Component1dLinear(Vector3d(0 kN/m, 0 kN/m, -300 kN/m), Vector3d(0 kN/m, 0 kN/m, -100 kN/m)));

To make the line load dependent of the beam the load intensities relate to a beam(s) end positions. In this case the line load follows when moving, but not copying the beam. To switch from independent to dependent line load choose from footprint “line” or “beam”.



You can select multiple beams graphically and apply the same line load to all by using the radio button “Apply to selected beams”.



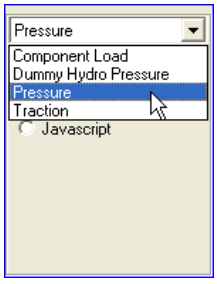
This command is scripted. Typically for a constant line load in z-direction between relative positions 0.2 and 0.7 on beam Bm14:

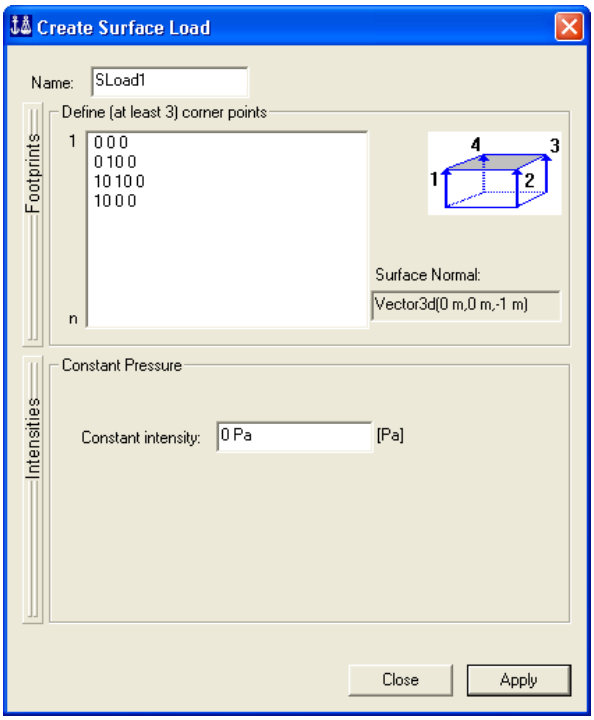
LLoad2 = LineLoad(LC1, FootprintBeam(Bm14, 0.2, 0.7), Component1dLinear(Vector3d(0 kN/m, 0 kN/m, -300 kN/m), Vector3d(0 kN/m, 0 kN/m, -100 kN/m)));

2.4.12.3 Surface Load

Purpose: To define a surface load on a plate or a shell. This load can only be applied on a plate or shell to give effect. The load may be independent or dependent of the structure. To make the line load independent of the plate the load intensities relate to three or more points. In this case the line load does not follow when moving or copying the beam.

There are a number of different intensities that can be used. Slide the “Intensities” bar to see the options.

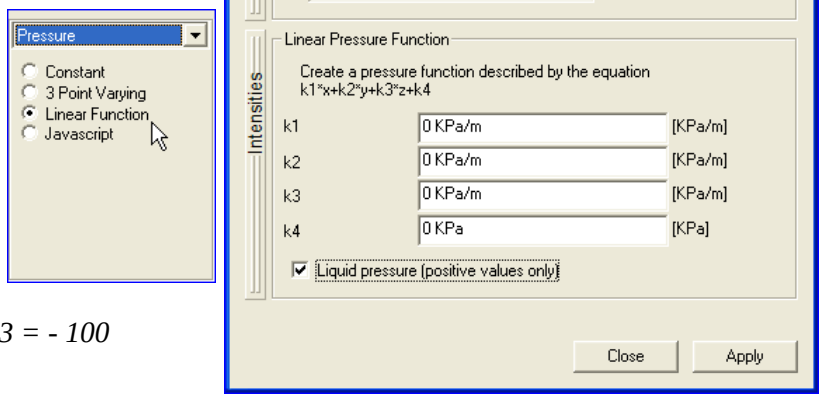




For plates the surface load may be independent or dependent. For shells the surface load must always be dependent by using a wet surface. In the following, all examples are based on a wet surface WS1.

2.4.12.3.1 Pressure

Purpose: To define a pressure on a plate or a shell. There are four options: Constant, 3 Point Varying, Linear Function (as shown below) and Javascript. The option Liquid pressure (positive values only) is among others used to compute hydrostatic pressure as only the positive pressure component is computed. Notice that this is the default option.

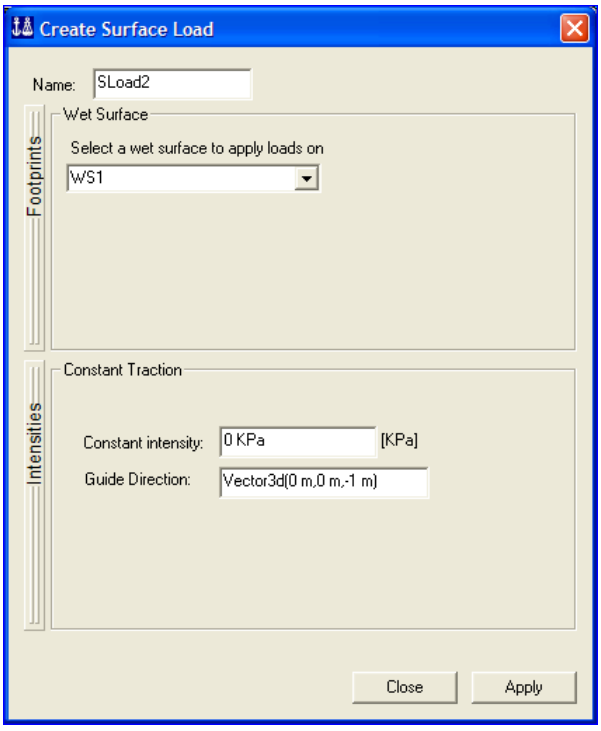


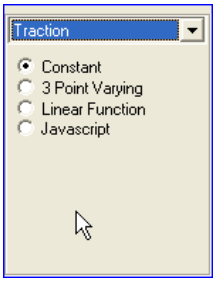
This command is scripted. Typically for a pressure with k4= 1000Kpa and k3 = - 100 KPa/m:

SLoad1 = SurfaceLoad(LC1, FootprintWetSurface(WS1),

2.4.12.3.2 Traction

Purpose: To define a traction on a plate or a shell. There are four options: Constant (as shown below), 3 Point Varying, Linear Function and Javascript.



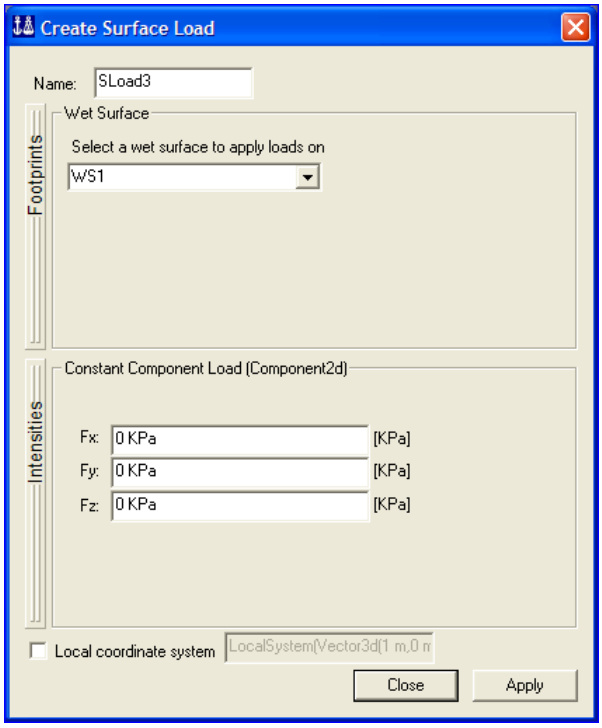


This command is scripted. Typically for a constant traction 100 Kpa:

SLoad2 = SurfaceLoad(LC1, FootprintWetSurface(WS1), Traction2dConstant(100 KPa, Vector3d(0 m,0 m,-1 m)));

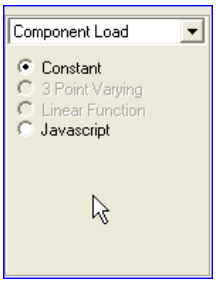
2.4.12.3.3 Component Load

Purpose: To define a component load on a plate or a shell. There are two options: Constant (as shown below) and Javascript.



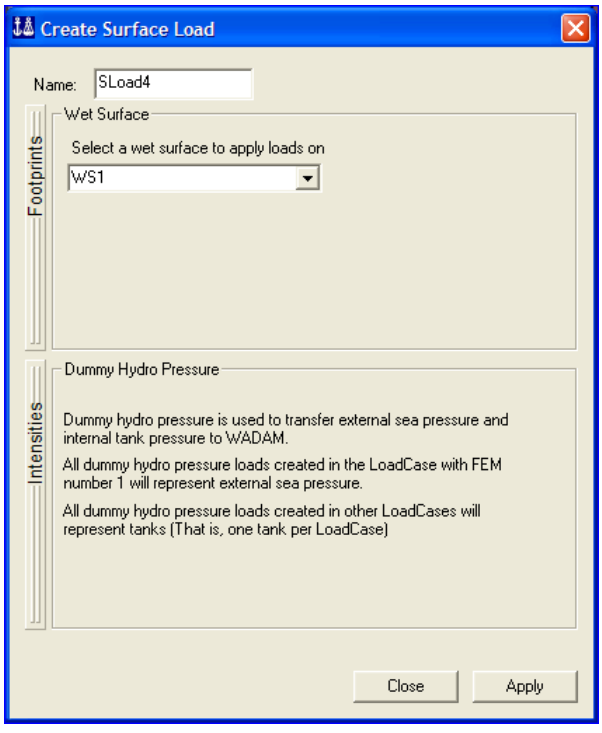
This command is scripted. Typically for Fx = 100 KPa and Fz = 200 KPa:

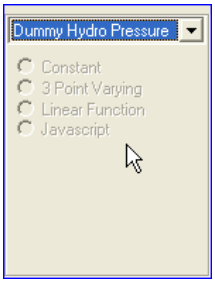
SLoad3 = SurfaceLoad(LC1, FootprintWetSurface(WS1), Component2dConstant(100 KPa, 0 KPa, 200 KPa));



2.4.12.3.4 Dummy Hydro Pressure

Purpose: To define a wet surface for use in HydroD. The wetted surface is used to decide the surfaces subjected to the sea (always finite element load case number 1) or surfaces belonging to a compartment (one load case per compartment). There are no load intensities to add since all the load application will be done in HydroD (Wadam).



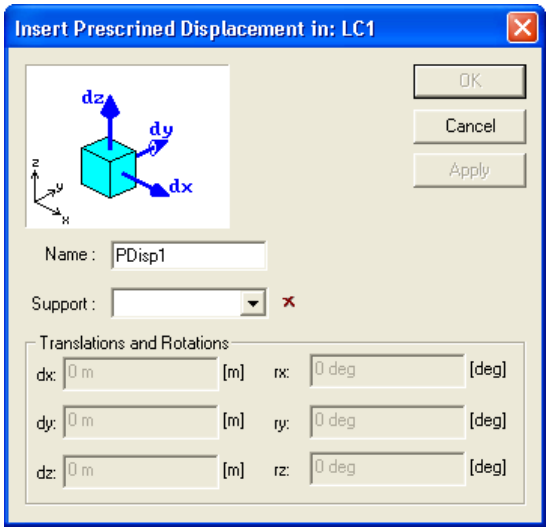


This command is scripted. Typically:

SLoad4 = SurfaceLoad(LC1, FootprintWetSurface(WS1), DummyHydroPressure());

2.4.12.4 Prescribed Displacements

Purpose: To specify a prescribed displacement at a support point. It must be used in connection with a prescribed boundary condition. See Section 3.11.5 of User Manual Volume 1 and Section 4.2.4 of User Manual Volume 3 for more details.



This command is scripted. Typically for a support point that has prescribed conditions in x and y direction with prescribed displacements of 2mm and 5mm respectively:

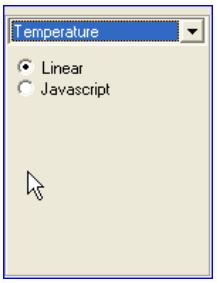
Sp3.boundary = BoundaryCondition(Prescribed, Prescribed, Fixed, Fixed, Fixed, Fixed);

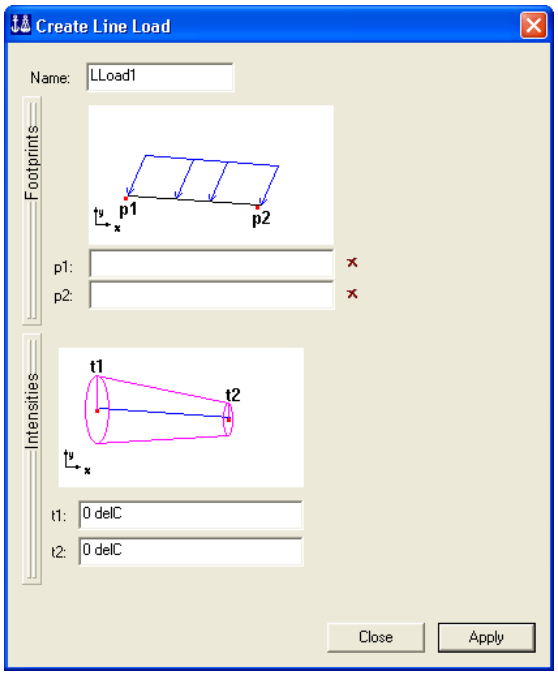
PDisp1 = PrescribedDisplacement(LC1, Sp3, 2mm, 5mm, 0 m, 0 deg, 0 deg, 0 deg);

2.4.12.5 Line Temperature

Purpose: To define constant or linearly varying temperature intensity between two points along a beam. Notice that the temperature is constant over the beams cross section. It is possible to specify the temperature by referring to two explicit points; in this case the load becomes independent of the beam and will not follow when the beam is moved or copied.

There are two options: Linear (as shown below or Javascript). Slide the "intensities" bar to change between the two options.





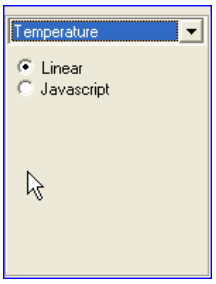
For more details, see Section 3.11.4 of User Manual Volume 1 or Section 4.2.3 of User Manual Volume 3.

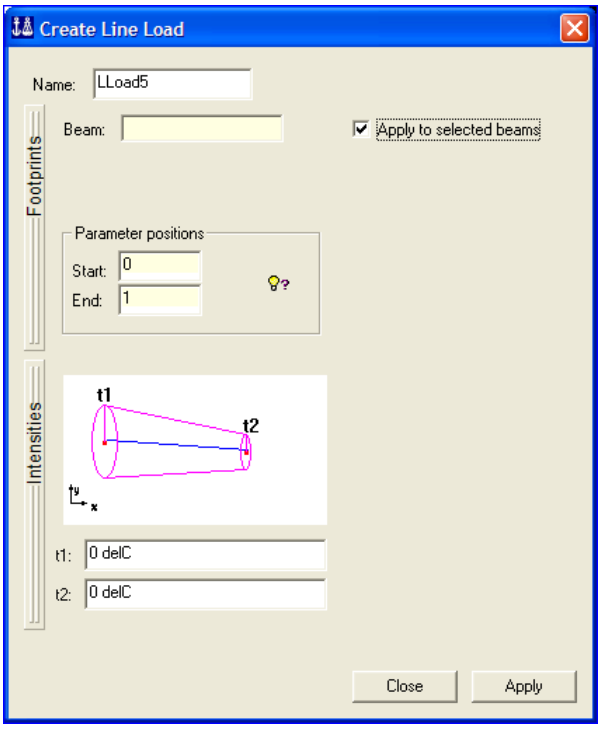
This command is scripted. Typically for a constant temperature of 100 delC between two points:

LLoad5 = LineLoad(LC1, FootprintLine(Point(2 m,0 m,1 m), Point(2 m,5 m,1 m)), Temperature1dLinear(100 delC, 100 delC));

It is also possible to specify the temperature by referring to relative positions of a beam(s); in this case the load becomes dependent of the beam and will follow when the beam is moved but not when it is copied.

There are two options: Linear (as shown below or Javascript). Slide the "intensities" bar to change between the two options.

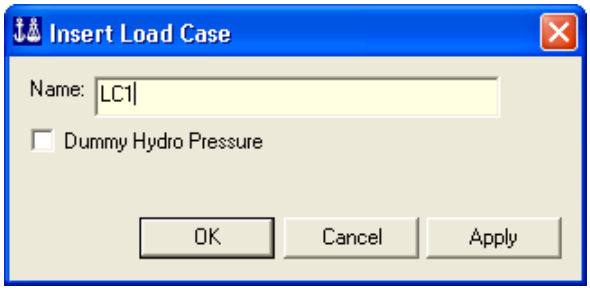




This command is scripted. Typically for a varying temperature load between the two end positions on beam Bm20:

LLoad6 = LineLoad(LC1, FootprintBeam(Bm20), Temperature1dLinear(50 delC, 100 delC));

2.4.13 Load Case

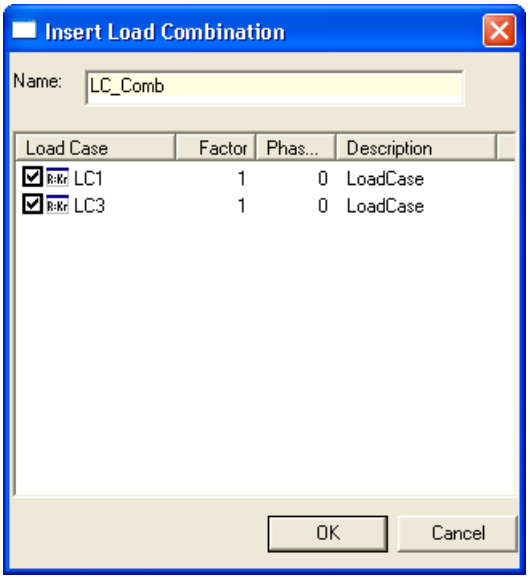


Purpose: To define a load case where equipments, weight lists, and explicit loads are applied. For more details see Section 3.8 of User Manual Volume 1 and Section 4.1 of User Manual Volume 3.

This command is scripted. Typically:

LC1 = LoadCase();

2.4.14 Load Combination



Purpose: To define load combinations built up of other load cases. It is possible to use a load combination in another load combination. For more details see Section 3.8 of User Manual Volume 1 and Section 4.1 of User Manual Volume 3.

This command is scripted. Typically:

LC_Comb = LoadCombination();

LC_Comb.addCase(LC1, 1);

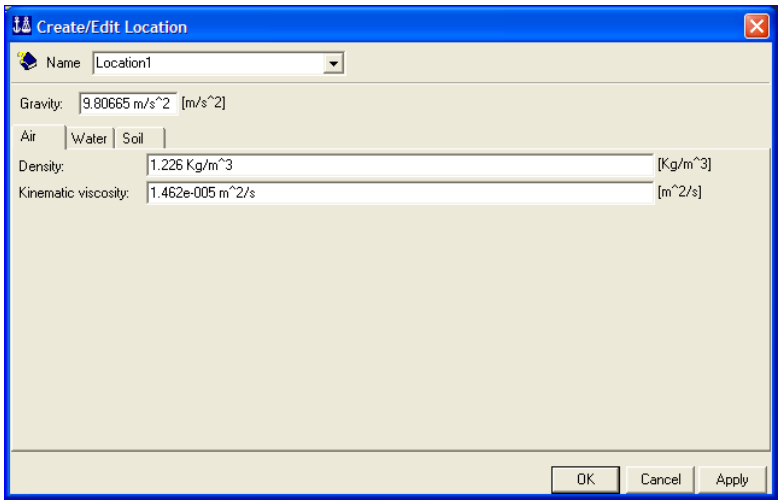
LC_Comb.addCase(LC3, 1);

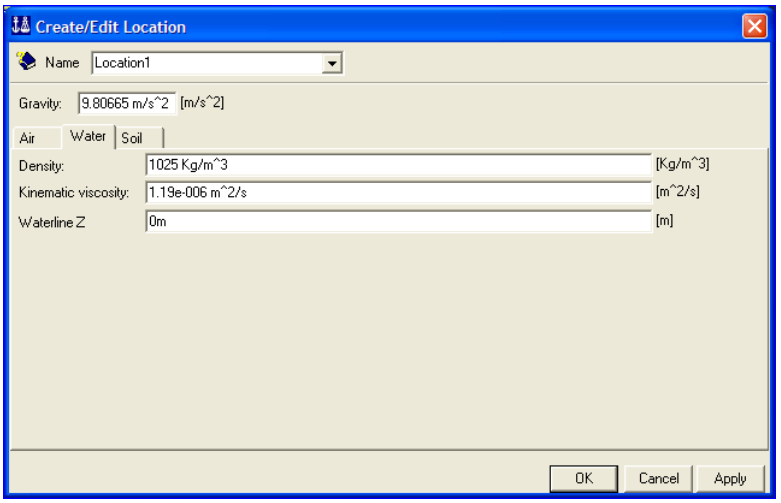
2.4.15 Insert – Environment

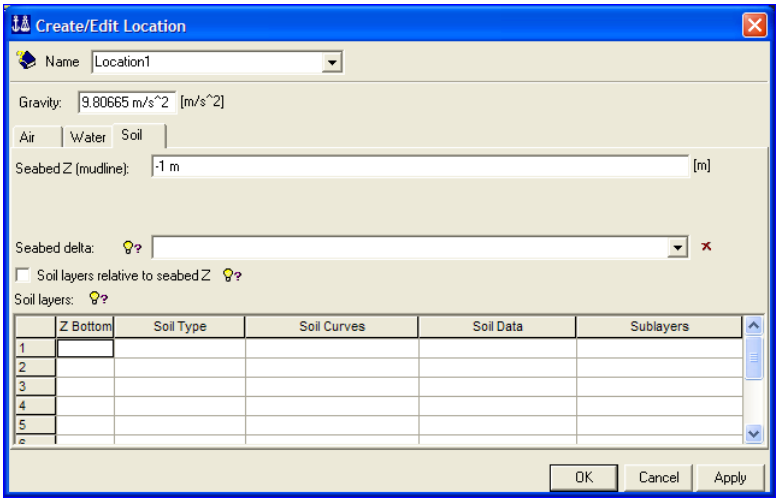
Purpose: To define global properties for air, water and the soil. Furthermore, the data for running a deterministic time condition (air and/or wave) is set up herein. Please see Volume 2 of the User Manual for more details.

2.4.15.1 Insert Location

Purpose: To define properties for air, water and the soil. All data for the air, water and the soil will be scripted upon clicking the "OK" or "Apply" buttons. The program default values are shown on the pictures. For the soil you may define soil type, soil curves and soil data from this input dialogue or select from the browser *Environment -> Soil*. See next Chapter for details.







This command is scripted. Typically:

Location1 = Location(0m, -1 m);

Location1.gravity = 9.80665 m/s^2;

Location1.air.density = 1.226 Kg/m^3;

Location1.air.kinematicViscosity = 1.462e-005 m^2/s;

Location1.water.density = 1025 Kg/m^3;

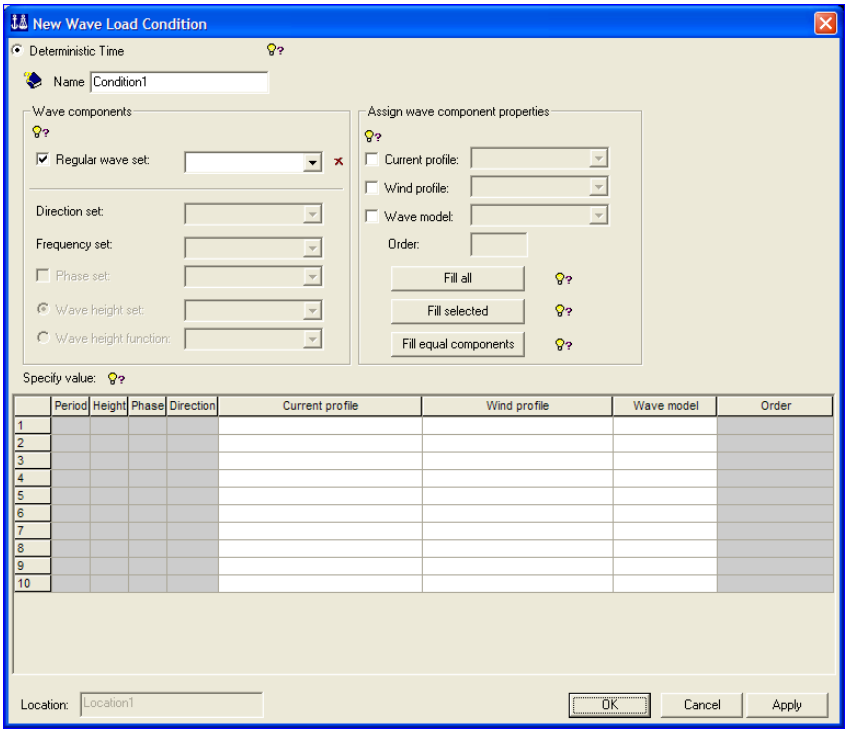
Location1.water.kinematicViscosity = 1.19e-006 m^2/s;

Location1.seabed.normaldirection = Vector3d(0 m,0 m,1 m);

Location1.relativeSoilLayers = false;

2.4.15.2 Insert Deterministic Time Condition

Purpose: To define how the environmental data will be used in a deterministic time domain analysis. The picture below shows that there is no default values set up for the wave load condition. As a minimum, the values for a regular wave set and the wave model must be specified to create a wave load condition. The relevant data for Current, Wind, Direction Set, Phase set, Wave height set, Wave height function may be defined from this dialogue or from the browser *Environment -> Directions* or *Environment ->Water*. See next Chapter for details. The Wave model may be of type *Airy, CalmSea, Cnoidal, Stoke5 or StreamFunction.* For StreamFunction it is also necessary to fill in the order as an integer between 1 and 24.



This command is scripted. The example below shows the script commands where the wave set "WaveSet1" (period 15 sec, wave height 5m, direction 0 deg and phase angle 0 deg) and the current profile "CurrentProfile1" (with varying current from 1 m/s^2 to 0.1 m/s^2 between elevation 10m above sea level down to sea bottom at -100m) are used. In addition the wave model is set to Stokes 5th. Typically:

WaveSet1 = RegularWaveSet();

WaveSet1.heightType(rwsHeight);

WaveSet1.add(RegularWave(0,5,WavePeriod(15),0));

CurrentProfile1_Elevations = Array(10,0,-50,-100);

CurrentProfile1_Directions = Array(0,0,0,0);

// 1 m/s^2 at 10m and 0m, 0.5 m/s^2 at -50m, 0.0 m/s^2 at -100m

CurrentProfile1_Velocities = Array(1,1,0.5,0);

CurrentProfile1 = CurrentProfileRelDir(CurrentProfile1_Elevations,CurrentProfile1_Directions, CurrentProfile1_Velocities,dtRelativeHeading);

Condition1 = DeterministicTime(Location1);

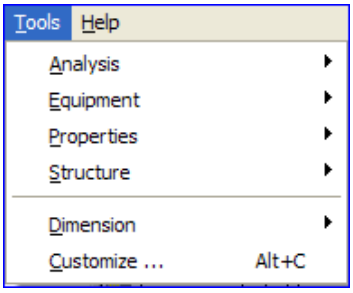
Condition1.waterSurface.regularWaveSet = WaveSet1;

Condition1.populate();

Condition1.component(1).water.current(CurrentProfile1);

Condition1.component(1).waterSurface.waveModel(Stokes5());

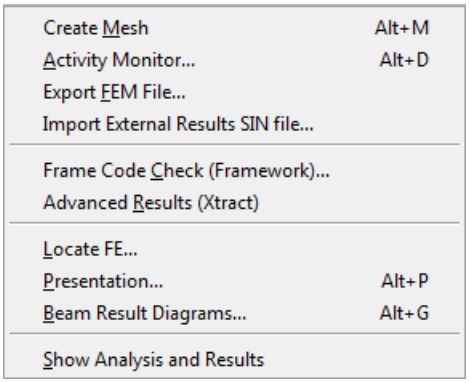
2.5 The Tools pulldown menu



The Tools pulldown menu contains commands to create a mesh, execute analysis, look at results, import equipment lists, create new material based on mass scaling, clean-up and divide of geometry, find length and angles and customize names and picture generation.

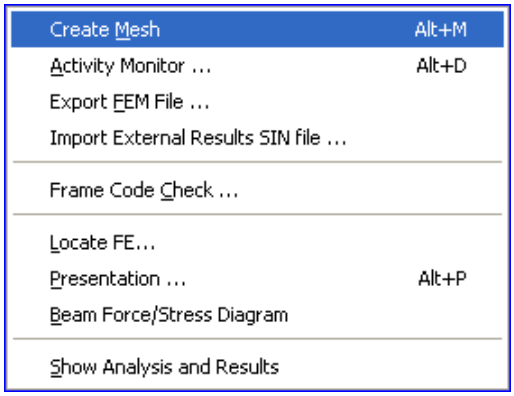
Each of these commands is listed in the following with a typical script command.

2.5.1 Tools – Analysis



Run your analysis, view results, export or import Sesam neutral files, perform code checking using previous code checking standards as part of Framework, locate finite elements and results presentations.

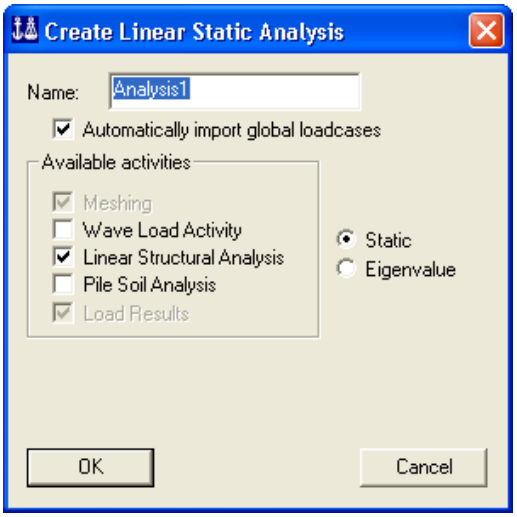
2.5.1.1 Create Mesh



Purpose: Create a finite element mesh and produce a FEM file.

This command is not scripted. You may edit the journal file and include CreateMesh();

2.5.1.2 Activity Monitor



Purpose: To select one of the predefined processes for executing a linear structural analysis with or without wave loads, pile and soil. The analysis programs Sestra, Wajac and Splice are used by GeniE to solve the analyses; these programs require a separate license key to run.

Each of the processes is pre-defined, but it is possible to edit and keep the settings for later use within the same workspace.

This command is scripted. Typically:

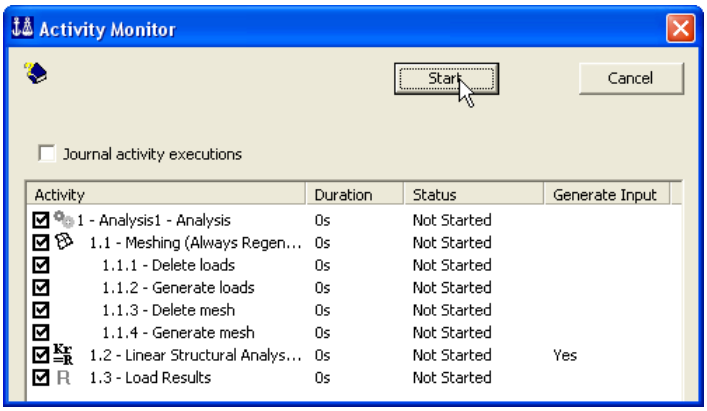
Analysis1 = Analysis(true);

Analysis1.add(MeshActivity());

Analysis1.add(LinearAnalysis());

Analysis1.add(LoadResultsActivity());

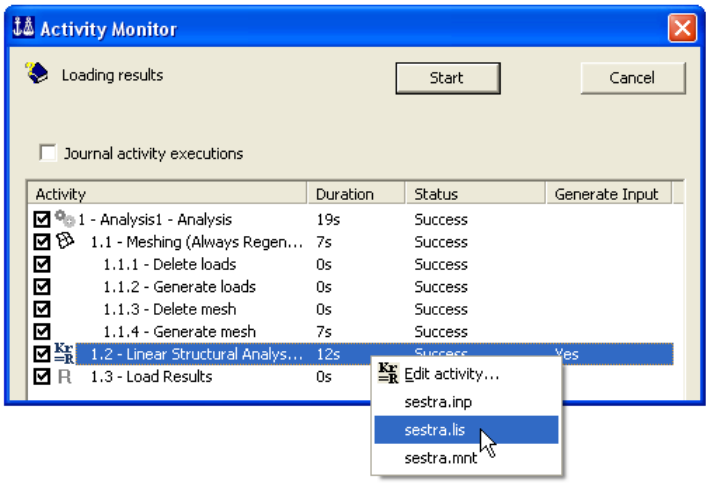
2.5.1.2.1 Linear structural analysis - static



Purpose: To execute and control input data for a static linear structural analysis. For more details see Section 3.13 of User Manual Volume 1 and Chapter 7 of User Manual Volume 3.

Provided you tick off for "Journal activity executions" this command is scripted. Typically:

Analysis1.execute();

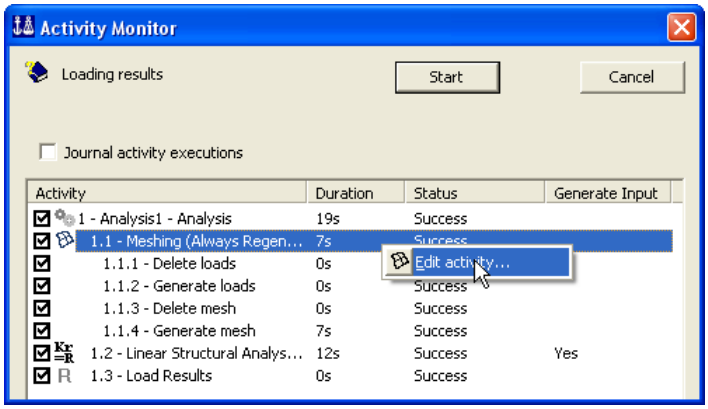


When the analysis has been performed each activity will contain a status flag *Success*, *Warning* or *Error*.

To investigate you right click on one of the activities to find more information. In the example to the right the listing file from Sestra is chosen.

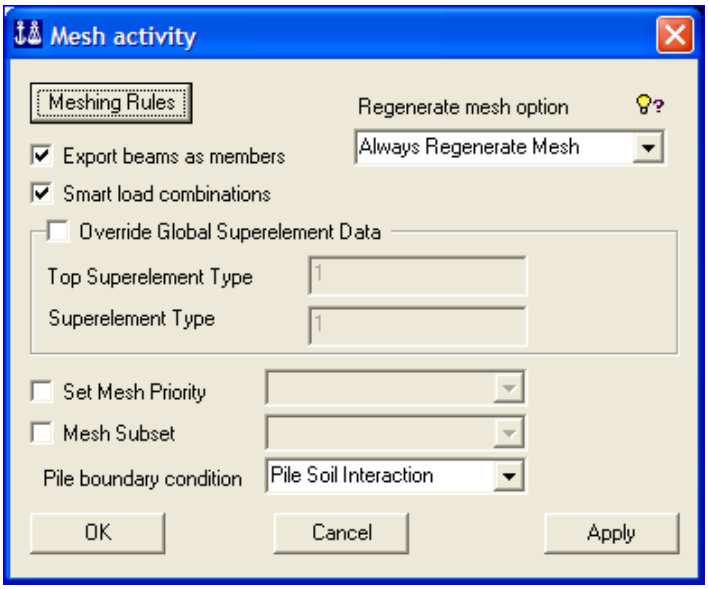
This command is not scripted.

It is also possible to set up criteria for the meshing. Right click the activity *Meshing* to access the relevant commands.



The *Meshing Rules* will open the command ***Edit|Rules|Meshing***.

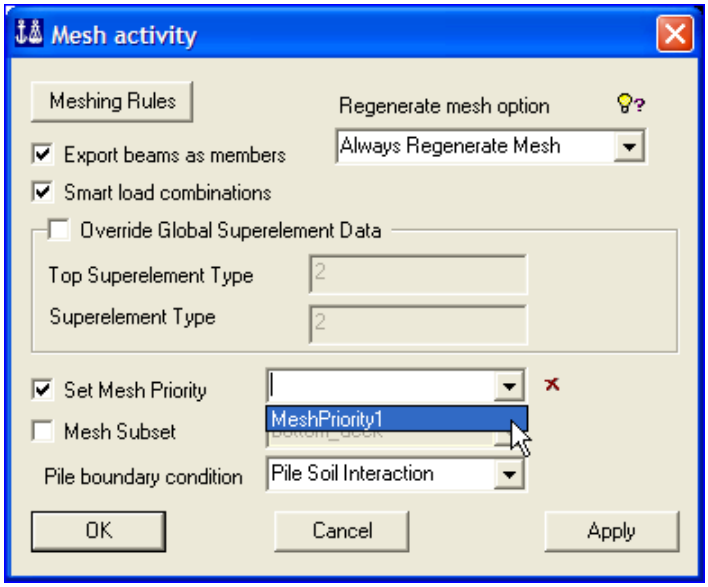
The *Export beams as members* will ensure that the finite element model contains member data in addition to pure finite element data – a member may be built up from several finite elements. You should always use this option unless you know you will not need it.



The *Smart load combinations* will reduce the analysis time since the load combinations are not part of a structural analysis. The results in GeniE include load combinations since all results are combined after the analysis. You are advised to use this option unless there are particular reasons for not doing so. Typical examples may be if you want all load combinations to be part of Sestra or you want to use load combinations for other analysis
purposes like in Usfos.

From the Override Global Superelement Data you can set the superelement type. The default is 1 and you can modify it from here or from
the Meshing Rules (***Edit|Rules|Meshing***).

When ticking off for the *Set Mesh Priority* the meshing will be performed accordingly, i.e. prioritized plates or beams will be meshed first. Please see the Chapter "The Browser Menu" to learn how to create a mesh priority.



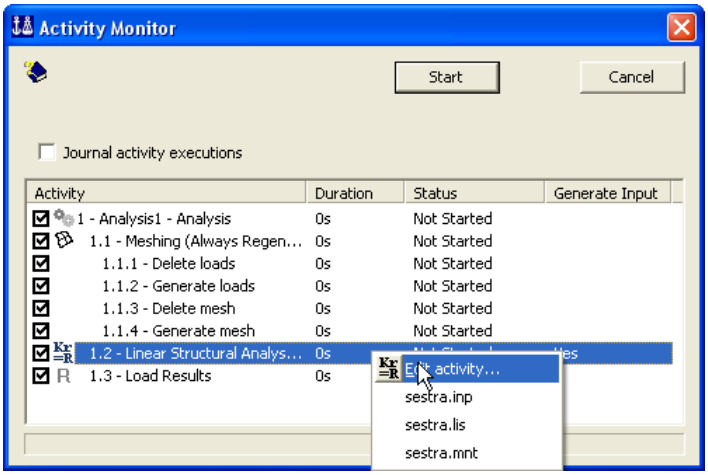
The Mesh Subset option is used to mesh parts of the concept model only. A boundary condition may also be part of a set – this means that a complete finite element model may be generated for sets (for analysis or for creation of super elements) as intersecting loads are automatically included.

This command is scripted, typically:

Analysis1.step(1).subset = bottom_deck;

Analysis1.step(1).meshPriority = MeshPriority1;

To modify the analysis activity you right click the activity *Linear Structural Analysis* and select *Edit activity*.

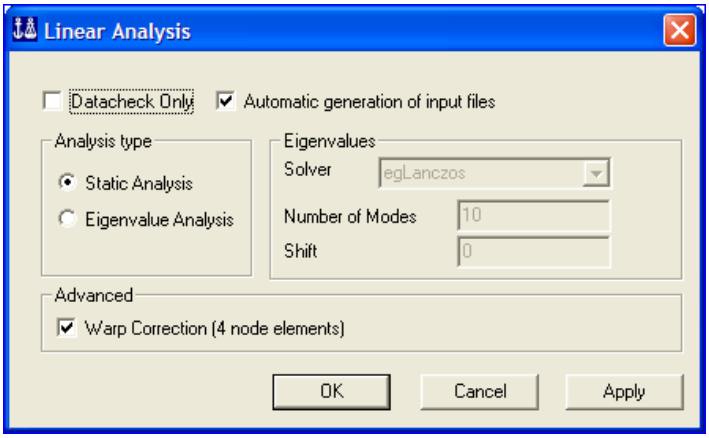


The *Datacheck Only* will set up the input file to Sestra so a data check is performed but no execution of finite element analysis.

This command is scripted, typically:

Analysis1.step(2).dataCheckOnly = true;

If you want to edit and keep the input file to Sestra you should deselect the *Automatic generation of input files* and click on the file sestra.inp to edit. If you do not deselect this option the modified input file to Sestra is not kept for subsequent analysis.



This command is scripted, typically:

Analysis1.step(2).generateInput = false;

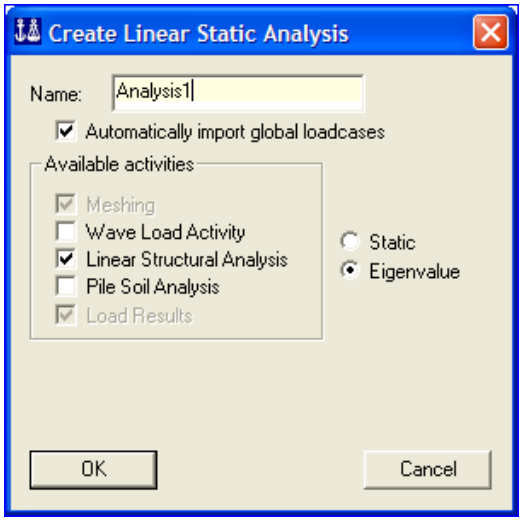
The option for Warp Correction will use Sestra's option for improving the stiffness for warped 4 noded shell elements (i.e. elements with "nodes out of plane").

This command is scripted, typically:

Analysis1.step(2).warpCorrection = false;

2.5.1.2.2 Linear structural analysis – eigenvalue

Purpose: To execute and control input data for an eigenvalue analysis.



Provided you tick off for "Journal activity executions" this command is scripted. Typically:

Analysis1 = Analysis(true);

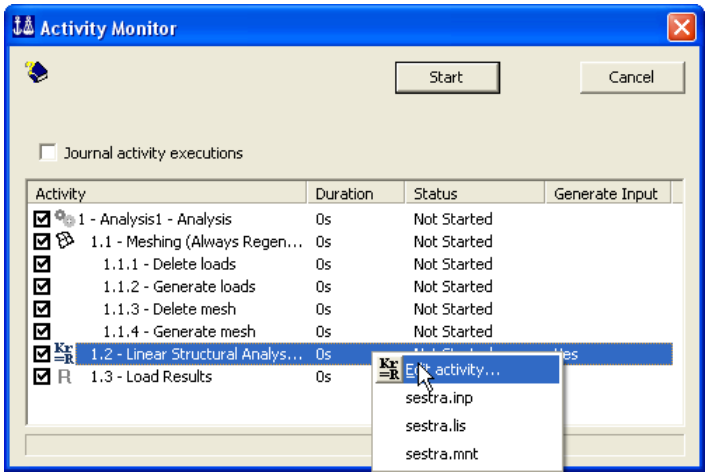
Analysis1.add(MeshActivity());

Analysis1.add(LinearAnalysis());

Analysis1.step(2).setEigenvalueAnalysis(egLanczos, 10);

Analysis1.add(LoadResultsActivity());

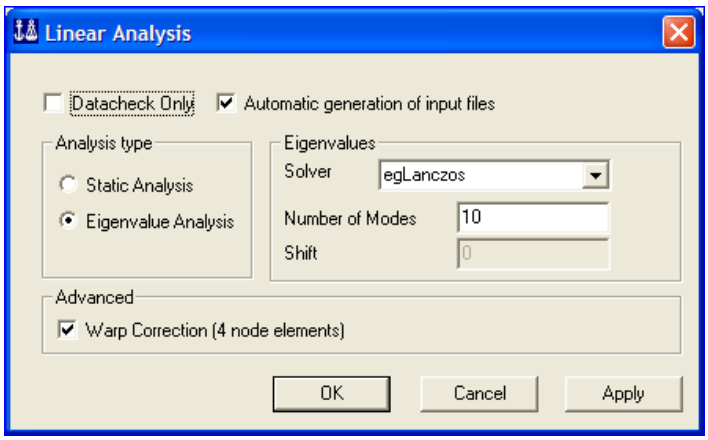
Analysis1.execute();



The default option is to use Lanczos solver with 10 eigen-modes. To modify you right click the activity *Linear Structural Analysis* and select *Edit activity*.

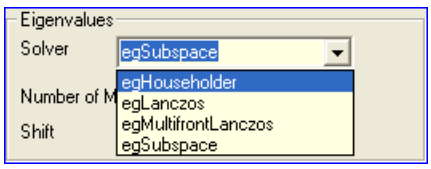
The following eigenvalue algorithms are available for free vibration analysis:

- An implicitly restarted Lanczos' method combined with a multifrontal solver
- Householder's method
- Subspace Iteration
- Lanczos' method



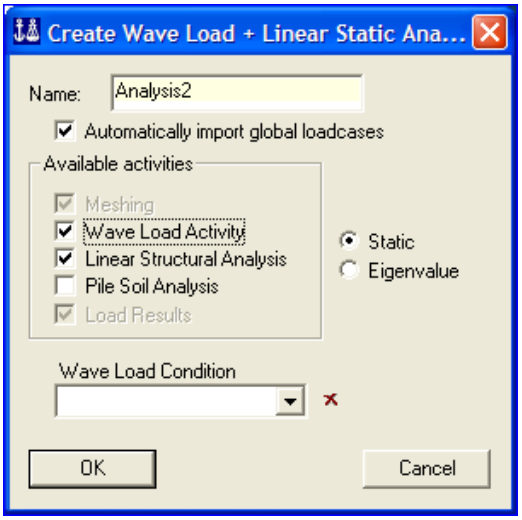
This command is scripted, typically:

Analysis1.step(2).setEigenvalueAnalysis (egMultifrontLanczos, 15, 2);



2.5.1.2.3 Wave Load Activity

Purpose: To execute and control input data for a wave load activity with or without a subsequent linear structural analysis automatically accounting for the wave, wind or current loads. For more details, see e.g. Section 3.2.3 of User Manual Volume 2.



It is required to define a wave load condition that can be referred to when defining the wave load activity (***Insert|Environment|Deterministic Time Condition***).

Provided you tick off for "Journal activity executions" this command is scripted. Typically:

Analysis1 = Analysis(true);

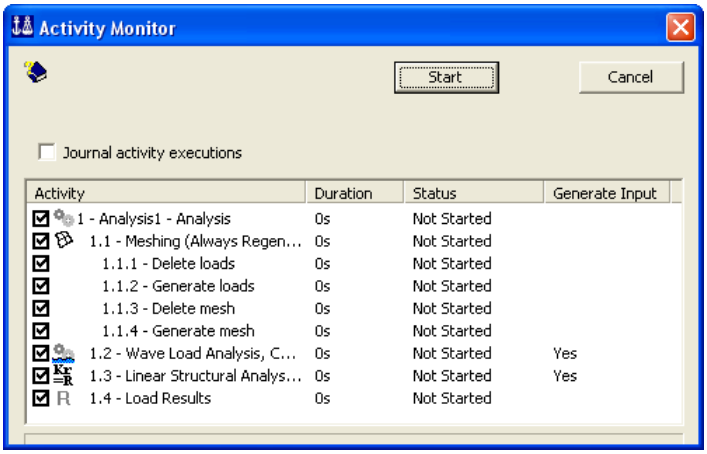
Analysis1.add(MeshActivity());

Analysis1.add(WaveLoadActivity (Condition1));

Analysis1.add(LinearAnalysis());

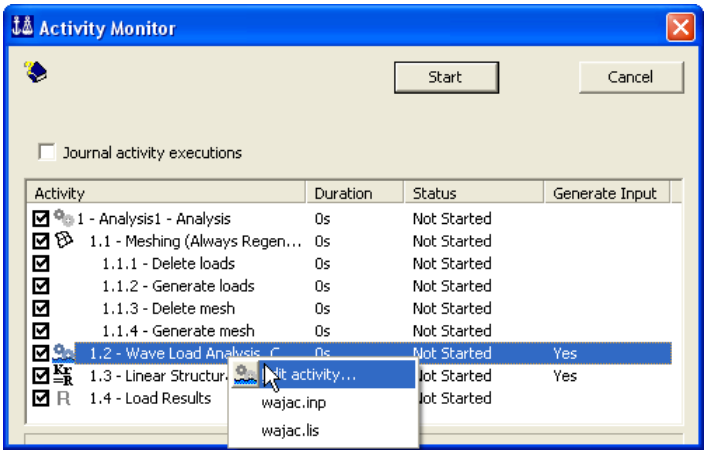
Analysis1.add(LoadResultsActivity());

Analysis1.execute();



In the example above the wave load condition Condition1 has been chosen.

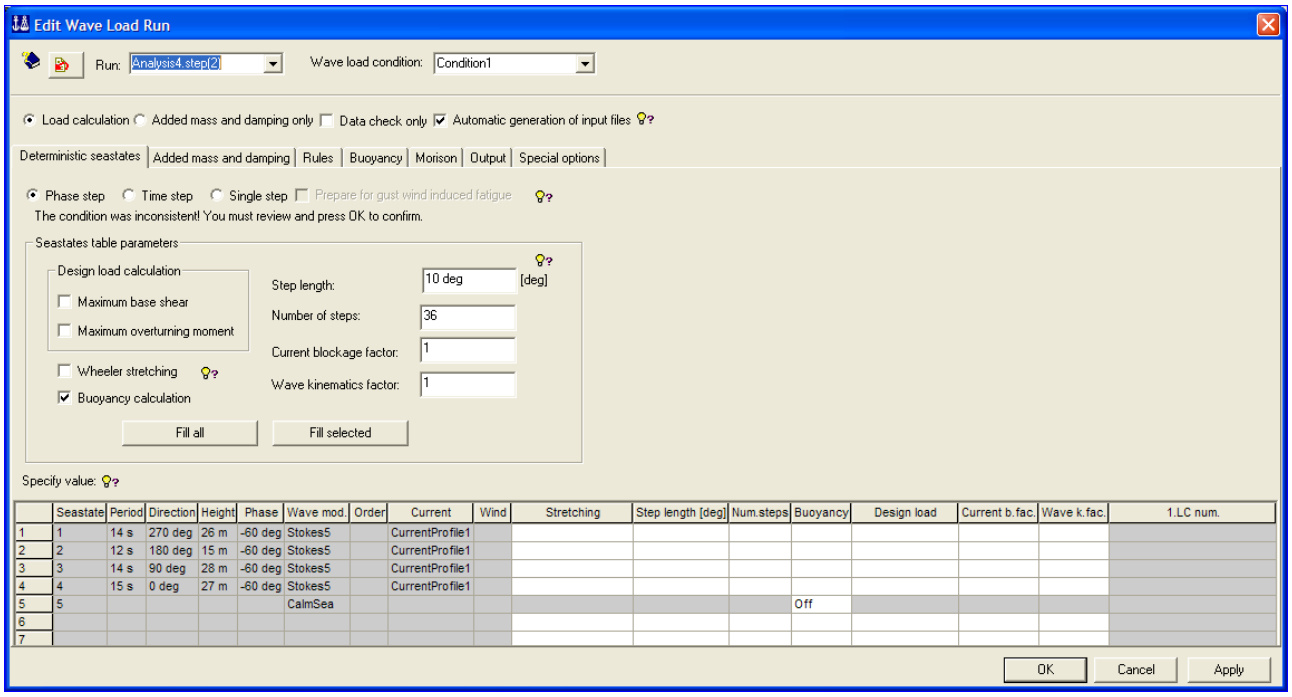
To run the analysis it is required to define execution directives for Wajac. This is done by editing the activity *Wave Load Analysis*.



2.5.1.2.3.1 Load calculation

A wave load calculation performed in GeniE by use of Wajac must be a deterministic time domain analysis. The type of load calculation is determined by the hydro condition dataset selected.

The following input parameters should be considered depending on your task (the dialogue below is filled out with data from a typical wave load condition *Condition1*):



Deterministic seastates:

Phase step, *Time step* or *Single step* can be selected. Please notice that if you want to create analysis results for use in a gust wind analysis in Framework only the option *Single step* can be used.

In case you check for *Maximum base shear* and Maximum overturning moment there are only two load cases part of the load transfer file to the structural analysis.

Appropriate *Current blockage factor* may be specified by the user. The total current profile will be multiplied by this factor after calculation of apparent wave period and before combination with wave kinematics.

Wave kinematics factor: Appropriate spreading factor may be specified by the user. Horizontal velocities and accelerations from the selected wave theory and apparent wave period will be multiplied by this factor.

When ticking off for the *Wheeler stretching*, both waves and current is stretched for linear wave theory (Airy waves). For non-linear wave theory only the current is stretched.

You may also decide not to include *Buoyancy calculation* in the hydrodynamic load calculation

When you use the *Fill All* button the relevant data is filled in to the complete table (the white cells as shown above). If you want to fill only parts of the table, you should select the relevant rows (one row at a time or continuous rows) at the bottom and click *Fill Selected*.

This command is scripted. By using the default options as shown above, typically:

Analysis1.step(2).deterministicSeastates().populate();

Analysis1.step(2).deterministicSeastates().seastate(1).dataPhase(NoStretching,10 deg,36,true,NoDesignLoads,1,1);

Analysis1.step(2).deterministicSeastates().seastate(2).dataPhase(NoStretching,10 deg,36,true,NoDesignLoads,1,1);

Analysis1.step(2).deterministicSeastates().seastate(3).dataPhase(NoStretching,10 deg,36,true,NoDesignLoads,1,1);

Analysis1.step(2).deterministicSeastates().seastate(4).dataPhase(NoStretching,10 deg,36,true,NoDesignLoads,1,1);

Analysis1.step(2).deterministicSeastates().seastate(5).dataPhase(NoStretching,0,1,true, NoDesignLoads,1,1);

Added mass and damping only:

Added mass and damping are calculated and written to the Loads Interface File independently of the choice of load calculation type. The default value is not to calculate added mass and damping for transfer to subsequent dynamic structural analysis.

Include internal water option involves that added mass is written to the Loads Interface File including internal water in flooded members.

The Use C_m=2.0 option involves that C_m = 2.0 (added mass coefficient = 1.0) is used for all elements irrespective of the Cm values used in the force calculation.

Calculate damping option means that damping is written to the Loads Interface File.

This command is scripted. By selecting the above options above, typically:

Analysis1.step(2).addedMassAndDamping().calculateAddedMass(true);

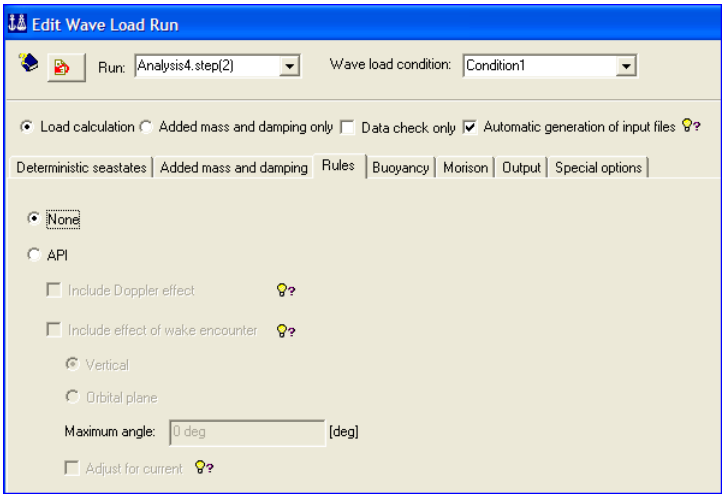
Analysis1.step(2).addedMassAndDamping().calculateDamping(true);

Analysis1.step(2).addedMassAndDamping().includeInternalWater(true);

Analysis1.step(2).addedMassAndDamping().useCm(true);

Rules:

It is possible to compute the wave loads in accordance with the API rules. The default is user definition of the wave load computations (i.e. no API rules).



The *Include Doppler effect* involves that the Doppler effect may be accounted for by using apparent wave (API 2.3.1b1).

Include effect of wake encounter involves that the effect of wake encounter may be included. There are two options:

Vertical: The dependency of wake encounter will be applied to all members within 15 degrees of the vertical (API 2.3.1b7) as specified by API.

Orbital plane: The dependency of wake encounter will be applied to all members within 15 degrees of the orbital plane of the wave kinematics.

- If a *maximum angle* is given this value will substitute 15 degrees.

- The effect of wake encounter may be *adjusted for current* (API C2.3.1b7).

This command is scripted. By selecting the above options above (and maximum angle 25 degrees), typically:

Analysis1.step(2).rules().setRuleType(wrAPI);

Analysis1.step(2).rules().includeDoppler(true);

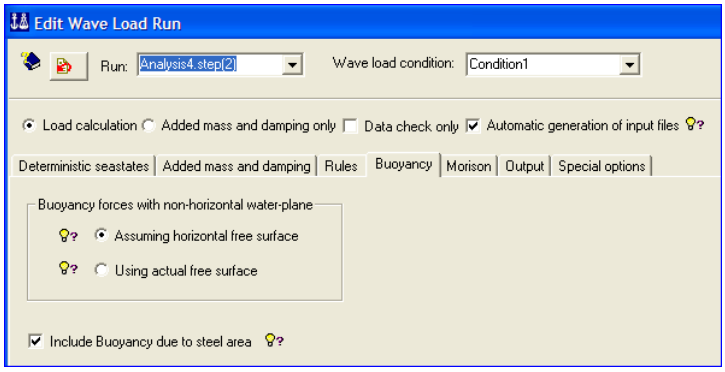
Analysis1.step(2).rules().applyWakeType(wrToVertical);

Analysis1.step(2).rules().adjustForCurrent(true);

Analysis1.step(2).rules().maxAngle = 25 deg;

Buoyancy:

It is possible to override the default options for how to calculate the buoyancy – the default options are shown below.



Assuming horizontal free surface:

- Buoyancy is calculated assuming a horizontal surface above each member.

- The largest of the surface elevations above the member ends is used at both ends.

- This option is default and is only available in time domain.

Using actual free surface:

- Buoyancy forces are calculated using the actual, i.e. a non-horizontal water-plane that gives small buoyancy components in the horizontal plane.

By default the *Include Buoyancy due to steel area* option is on meaning that buoyancy of the steel is included.

This command is scripted. By selecting the above options above, typically:

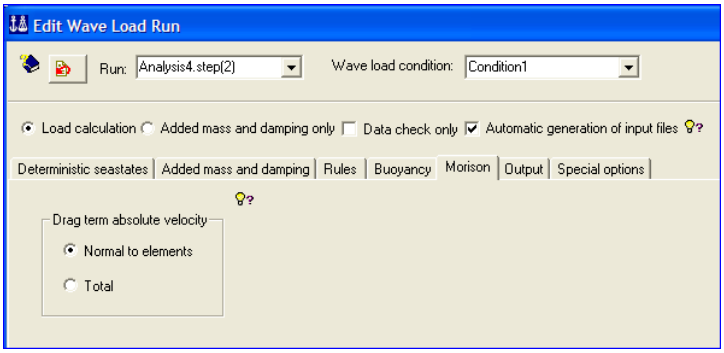
Analysis1.step(2).buoyancy().horisontalFreeSurface(true);

Analysis1.step(2).buoyancy().actualFreeSurface(false);

Analysis1.step(2).buoyancy().steelAreaBuoyancy(true);

Morison:

You may define if velocity decomposition shall be used when calculating the normal components of the drag forces, i.e. how to handle the v^2 term in the Morison equation.



Normal to elements involves that the absolute value of the normal velocity is multiplied with the velocity component normal to the member in the Morison equation (i.e. abs(v_n) is used). This is the default option.

Total involves that the absolute value of the normal velocity is replaced by the absolute value of the total velocity in the Morison equation (i.e. abs(v) is used.)

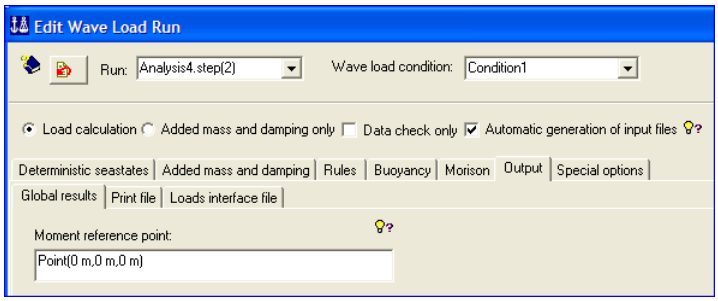
This command is scripted. By selecting the above options above, typically:

Analysis1.step(2).morison().normalToElement(true);

Output:

Under the Output tab you can specify the overturning moment reference point, how much to add to the printed file from Wajac and details for the loads interface file.

The *Moment reference point* about which the overturning moments are calculated are by default at the mud line directly below or above the origin of the global coordinate system. An alternative point may be specified by you.

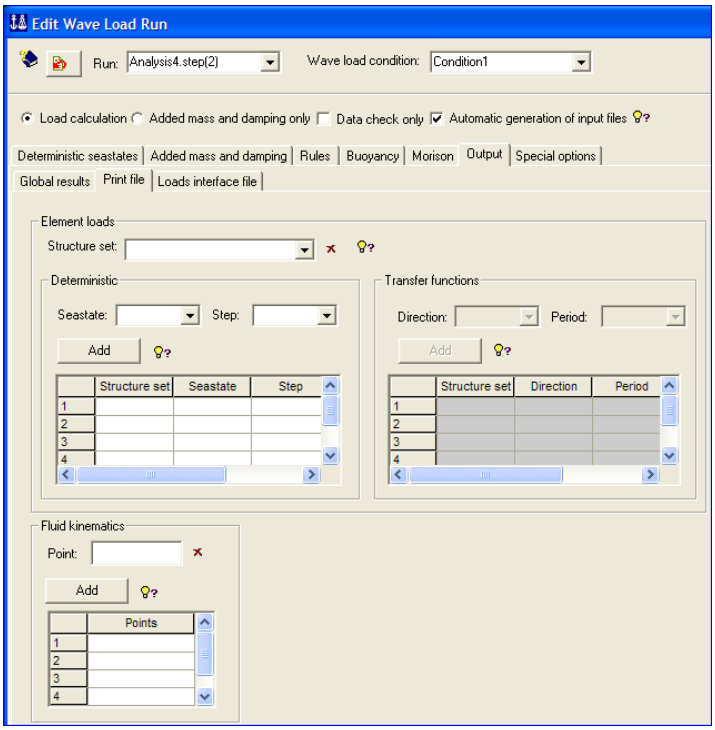


This command is scripted. By specifying the z-value to 50 meters, typically:

Analysis1.step(2).output().globalResults().momentRefPoint = Point(0 m,0 m,50 m);

The *Print file* option allows you to add additional details to the Wajac listing file from the hydrodynamic analysis.

Under *Structure Set* you can select parts of structure that should have force intensities printed. The selected sets will be included in the *Add operations* for *Deterministic* and *Fluid kinematics*.

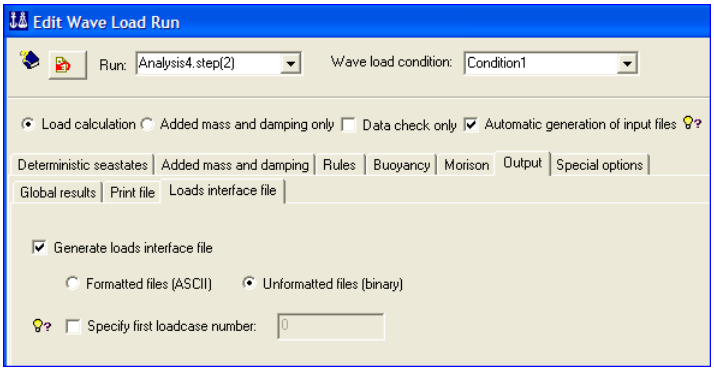


This command is scripted. By selecting the named set "Jacket", sea state number "2" and step "10", typically:

Analysis1.step(2).output().printFile().addDeterministicData(Jacket,2,10);

The *Loads interface file* option controls type of file format and a manual control of the first load case number.

The default option is to use an unformatted file and that the first wave load case number is a continuation of the basic load case numbers.



This command is scripted. By selecting unformatted files and start of first wave load case number to "45", typically:

Analysis1.step(2).output().loadsInterfaceFile().formatted(false);

Analysis1.step(2).output().loadsInterfaceFile().loadCaseNumbering(true);

Analysis1.step(2).output().loadsInterfaceFile().firstLoadCaseNumber = 45;

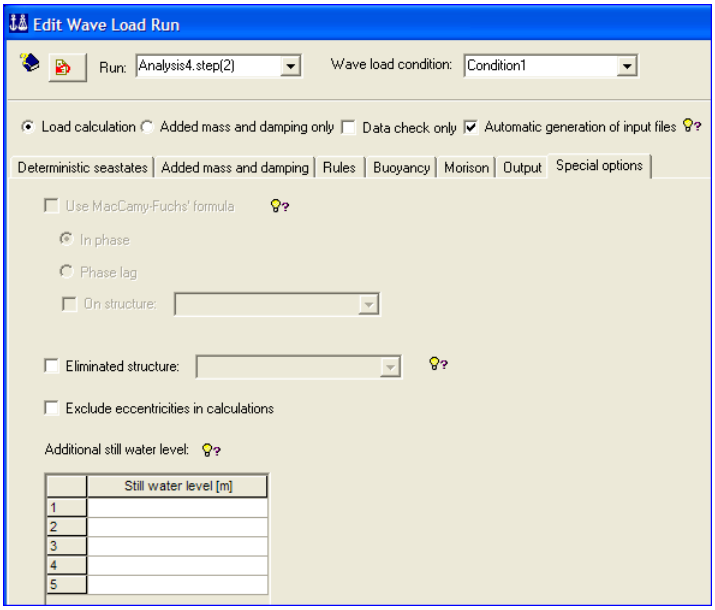
Special options:

Under the Special options tab you can eliminate structural parts from the wave load analysis. You can also add additional still water levels.

An *Eliminated structure* set may be selected from named sets and this part of the structure is eliminated from hydrodynamic load calculations.

Exclude eccentricities in calculations will disregard any eccentricities of the beams. If you choose this option, the beam length will in most cases be increased (node to node). As such this is not the default option.

The Additional still water level you can add additional still water levels. The water levels are relative to the still water level specified under the environment data.



This command is scripted. By excluding eccentricities, eliminating the named set "Topside" and adding two new still water levels at +2m and -3 m relative to the base still water level, typically:

Analysis1.step(2).specialOptions().excludeEccentricities(true);

Analysis1.step(2).specialOptions().useEliminatedStructure(true);

Analysis1.step(2).specialOptions().eliminatedStructure = Topside;

Analysis1.step(2).specialOptions().removeAllLevels();

Analysis1.step(2).specialOptions().addLevel(2m);

Analysis1.step(2).specialOptions().addLevel(-3m);

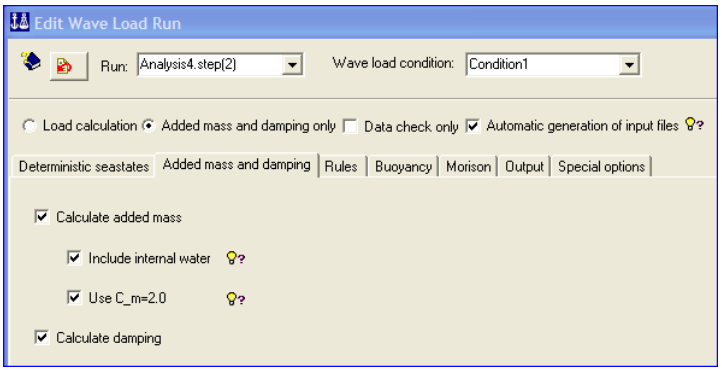
2.5.1.2.3.2 Added mass and damping only

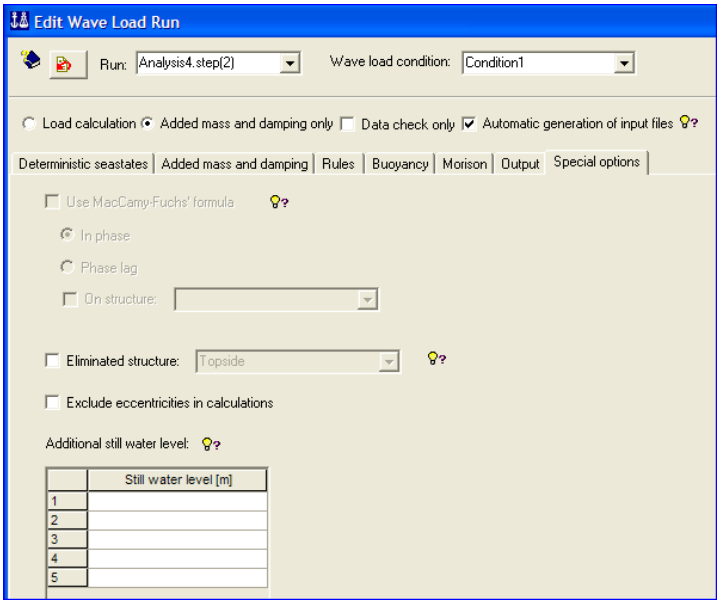
This option implies that no load cases will be calculated and written to the loads interface file – only added mass and damping are considered.

There are two tabs that can be accessed from this option; the *Added mass and damping* and the *Special options*.

Notice that at least one of the *Calculate added mass* and *Calculate damping* must be selected.

For a definition of input parameters, please see previous section on *Load calculations*.





This command is scripted. By selecting all the options under "Added mass and damping", typically:

Analysis1.step(2).addedMassAndDampingOnly(true);

Analysis1.step(2).addedMassAndDamping().calculateDamping(true);

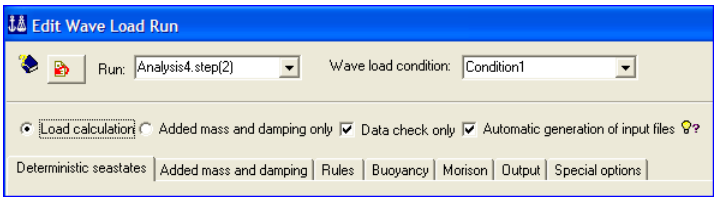
Analysis1.step(2).addedMassAndDamping().useCm(true);

Analysis1.step(2).addedMassAndDamping().includeInternalWater(true);

2.5.1.2.3.3 Data check only

The data check option prevents Wajac from a full execution of the analysis; a data check will be performed of the input data to determine whether they are consistent.

The result from a data check can be inspected on the Wajac listing file.



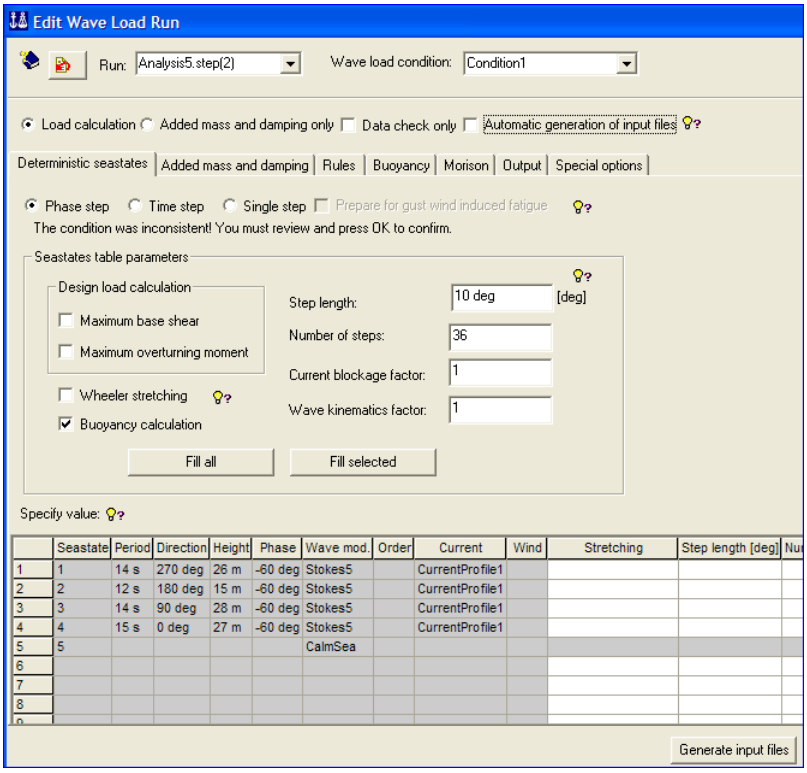
This command is scripted. By activating the option "Data check only", typically:

Analysis1.step(2).dataCheckOnly(true);

2.5.1.2.3.4 Automatic generation of input files

The default option ("True") implies that the input file for Wajac is automatically generated. In other words, if you modify one of the parameters this file is automatically updated prior to the next analysis run. If you want to have a manual control of the input file you should deselect this option.

In this case a *Generate input files* button appears. When you click on this button you create an input file based on the parameters defined. This file may be edited and changed. See also the previous section for linear static analysis on how to control and edit the input file.



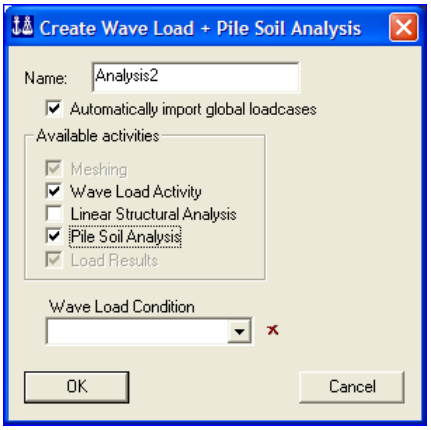
When deactivating the automatic input generation, it should be noted that changes in the environment, structure or properties should not be performed, as then the correspondence between the GeniE concept model and the Wajac input file may become out of sync and invalid.

This command is scripted. By activating the option "Automatic generation of input files", typically:

Analysis4.step(2).generateInput = false;

2.5.1.2.4 Pile Soil Analysis

Purpose: To execute and control input data for a pile soil analysis. For more details, see e.g. Section 3.6 of User Manual Volume 2.



It is required to define a wave load condition that can be referred to when defining the wave load activity (***Insert|Environment|Deterministic Time Condition***).

Provided you tick off for "Journal activity executions" this command is scripted. Typically:

Analysis1 = Analysis(true);

Analysis1.add(MeshActivity());

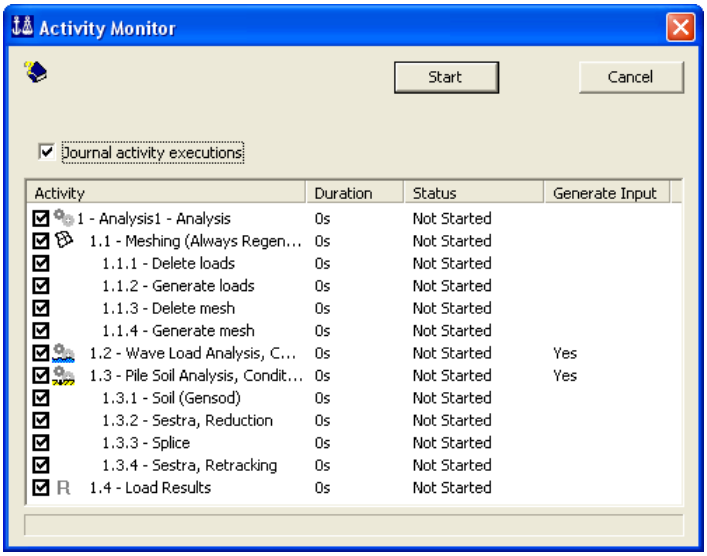
Analysis1.add(WaveLoadActivity (Condition1));

Analysis2.add(PileSoilAnalysis (Condition1));

Analysis1.add(LinearAnalysis());

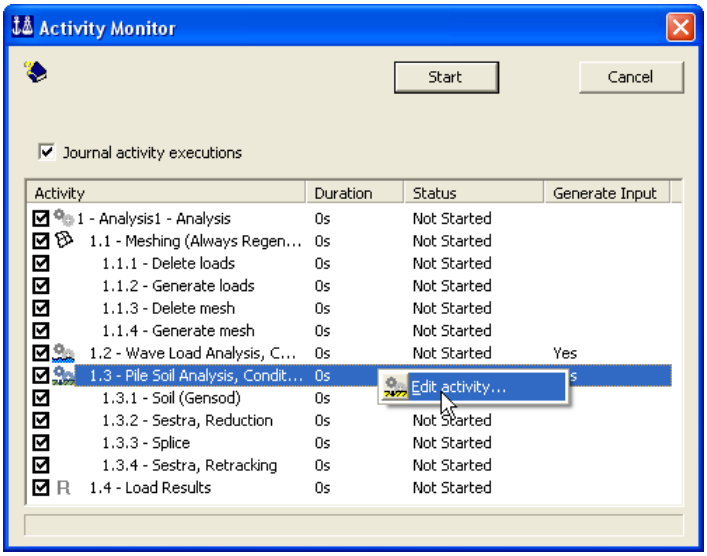
Analysis1.add(LoadResultsActivity());

Analysis1.execute();



In the example above the wave load condition Condition1 has been chosen.

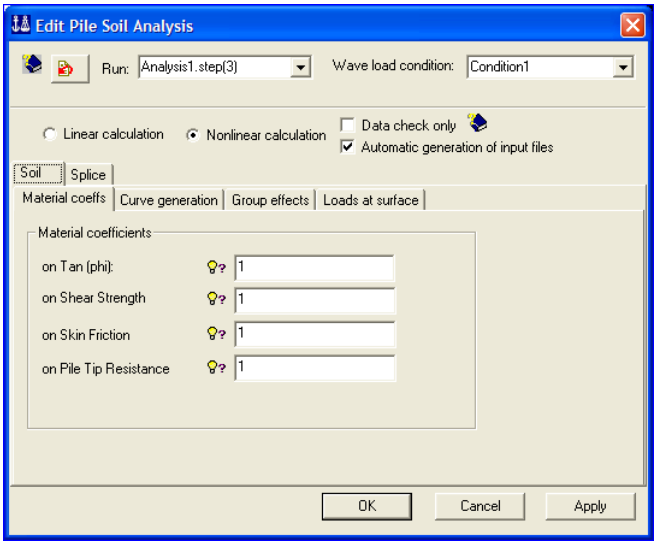
To run the analysis it is required to define execution directives for Wajac (see previous Section) and Splice (including Gensod). This is done by editing the activity *Pile Soil Analysis*.



2.5.1.2.4.1 Nonlinear calculation

To run a pile soil analysis in GeniE by use of Splice (including Gensod) requires that special geotechnical data is given for the soil material, soil curves, group effects and how to handle loads at the soil surface. In addition number of iterations and convergence criteria must be specified for the non-linear pile soil analysis performed in Splice.

In addition there are other properties for the piles and the soil that need to be defined, see pile characteristics as defined in e.g. Sections 2.2.6.1 and 2.4.1.4 as well as soil definitions from the browser menu *Environment*.



Material coefficients

Material coefficients can be seen as safety factors used to modify the soil properties without changing the actual values. The actual values are divided by the coefficients meaning that a coefficient greater then one (1) reduces the capacity.

There are four coefficients related to the friction angle (Tan (phi)), un-drained shear strength, skin friction and pile tip resistance. The default values for the soil material coefficients are 1; i.e. the actual value is the same as the capacity.

This command is scripted. Typically:

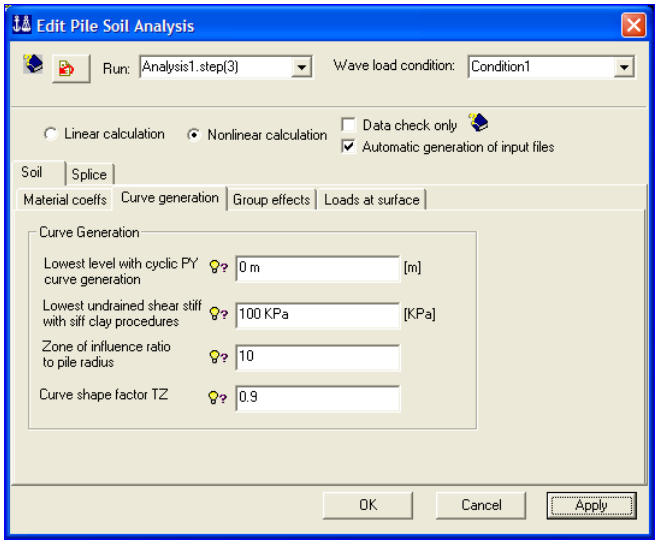
Analysis1.step(3).soil.materialCoeffs.tanPhiCoeff = 1.1;

Analysis1.step(3).soil.materialCoeffs.shearStrengthCoeff = 1.2;

Analysis1.step(3).soil.materialCoeffs.skinFrictionCoeff = 1.3;

Analysis1.step(3).soil.materialCoeffs.pileTipResistanceCoeff = 1.4;

Curve generation



Lowest level of cyclic PY generation:
- Cyclic PY data will be generated for all layers below the specified limit.

Lowest undrained shear stiffness with stiff clay procedures:
- The API code distinguishes between "soft clay" and "stiff clay". Procedures for calculating PY data are, however, only given for "soft" clay. Clay layers will be considered "stiff" when the undrained strength value given is higher than the limit specified. In such case the Reese et al (1975) are used instead of the API "soft" clay procedures.

Zone of influence ratio to pile radius:
- Outside the zone of influence the shear displacements due to pile axial loading are neglected. Typical values may be 10 to 30.

Curve shape factor TZ:
- Dimensionless factor that governs the shape of the TZ curve up to peak resistance. Zero gives a linear curve. Typical values may be 0.8 to 0.99.

The two latter values are needed for the Kraft et al (1981) procedures to generate tz-data based on tz-code = 200.

This command is scripted. Typically:

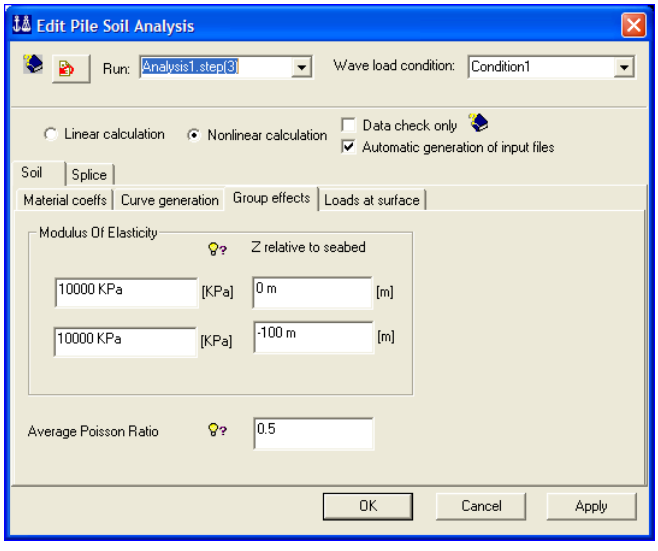
Analysis1.step(3).soil.curveGeneration.lowestLevelWithCyclicPY = 1 m;

Analysis1.step(3).soil.curveGeneration.lowestShearStiff = 120 KPa;

Analysis1.step(3).soil.curveGeneration.zoneOfInfluenceTZ = 20;

Analysis1.step(3).soil.curveGeneration.curveShapeFactorTZ = 0.85;

Group effects



Group interaction effects (pile-soil-pile interaction) are calculated from the elastic half-space theory assuming an E-modulus that increases linearly with depth.

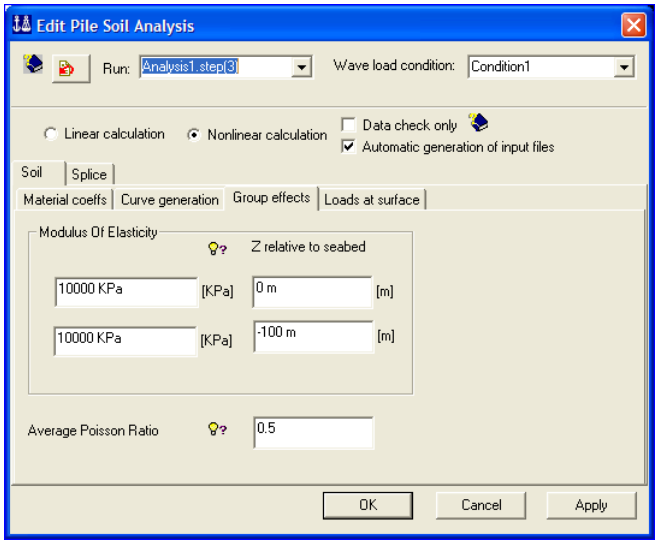
This command is scripted. Typically:

Analysis1.step(3).soil.groupEffects. averagePoissonRatio = 0.6;

Analysis1.step(3).soil.groupEffects. modulusOfElasticity(10000 KPa,0 m, 20000 KPa,-150 m);

Loads at surface

Different types of vertical loading may exist at the soil surface in the vicinity of the pile. Such loading will influence the vertical stress down along the pile and hence the lateral resistance.



Splice (Gensod) approximately accounts for:

- A uniform Vertical stress at soil surface
- An embankment type loading
- A circular loaded area with the pile at its centre

The positions of the different types of surface loads are therefore given relative to a hypothetical average pile. Vertical loads are assumed to act at a Z level corresponding to the general scoured soil surface. The required inputs for each of the 3 options are:

- Vertical stress at soil surface:
    - Uniform vertical stress acting over a large area
- Embankment loads are defined by:
    - Vertical stress under the flat part of the embankment loading
    - Width of the sloping part of the embankment loading
    - Horizontal distance from the pile axis to the toe of the embankment. Positive if the pile is located outside the embankment, negative if the pile is inside.
- Circular loaded area is defined by:
    - Vertical stress under the circular loaded area
    - Radius of the circular loaded area

This command is scripted. Typically:

Analysis1.step(3).soil.loadsAtSurface.verticalStressAtSurface = 200 KPa;

Analysis1.step(3).soil.loadsAtSurface.verticalStressUnderEmbankment = 100 KPa;

Analysis1.step(3).soil.loadsAtSurface.widthOfEmbankmentSlopingPart = 1 m;

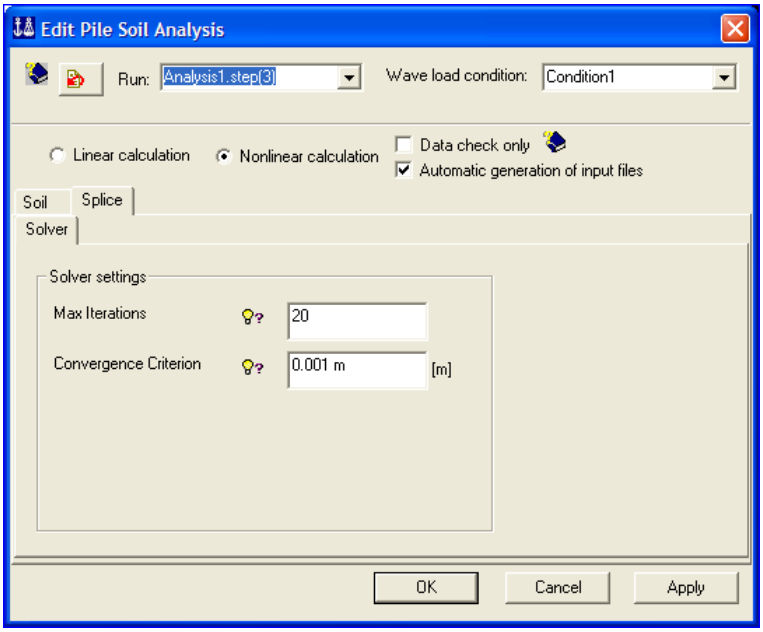
Analysis1.step(3).soil.loadsAtSurface.distancePileToEmbankmentToe = 0.5 m;

Analysis1.step(3).soil.loadsAtSurface.verticalStressUnderCircularLoadedArea = 50 KPa;

Splice

The solver settings that are used to control the Splice analysis are:

- Max iterations. The iterations are stopped when the convergence criterion is met or when the maximum number of iterations is reached. Maximum value is 50.
- Convergence criterion. The convergence criterion is met when the last iteration change in displacements along the pile is less than the specified length value.



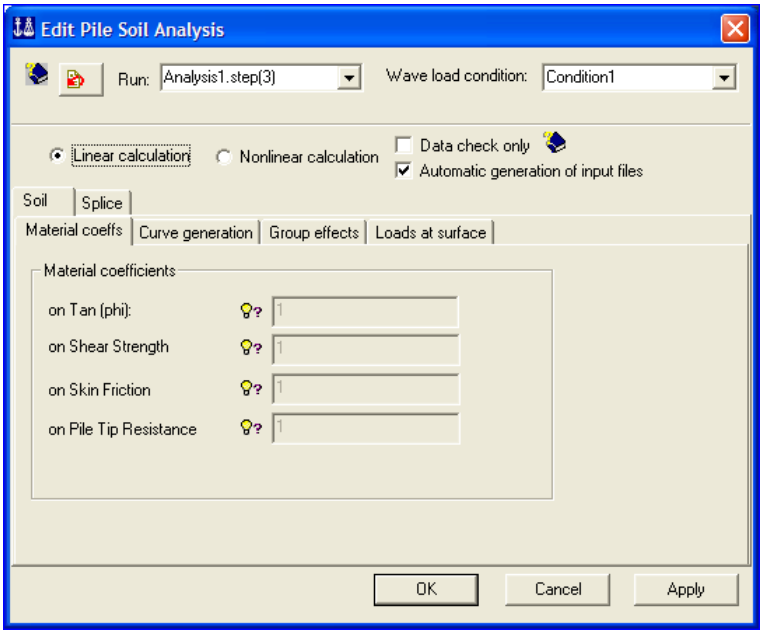
This command is scripted. Typically:

Analysis1.step(3).splice.solver.maxIterations = 25;

Analysis1.step(3).splice.solver.convergenceCriterion = 0.0015 m;

2.5.1.2.4.2 Linear calculation

It is possible to neglect the pile and soil analysis by inserting fixed boundary conditions along the piles and disregard the pile-soil analysis. This is the same as executing a wave and structural analysis where all piles are fixed at the finite element positions.



This command is scripted. Typically:

Analysis1 = Analysis(true);

Analysis1.add(MeshActivity());

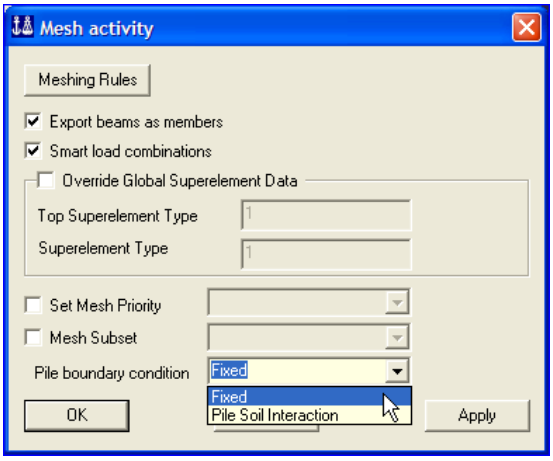
Analysis1.add(WaveLoadActivity (Condition1));

Analysis1.add(PileSoilAnalysis (Condition1));

Analysis1.add(LoadResultsActivity());

Analysis1.step(3).nonlinearAnalysis(false);

It is required to specify the pile boundary conditions to "Fixed" from the mesh activity.

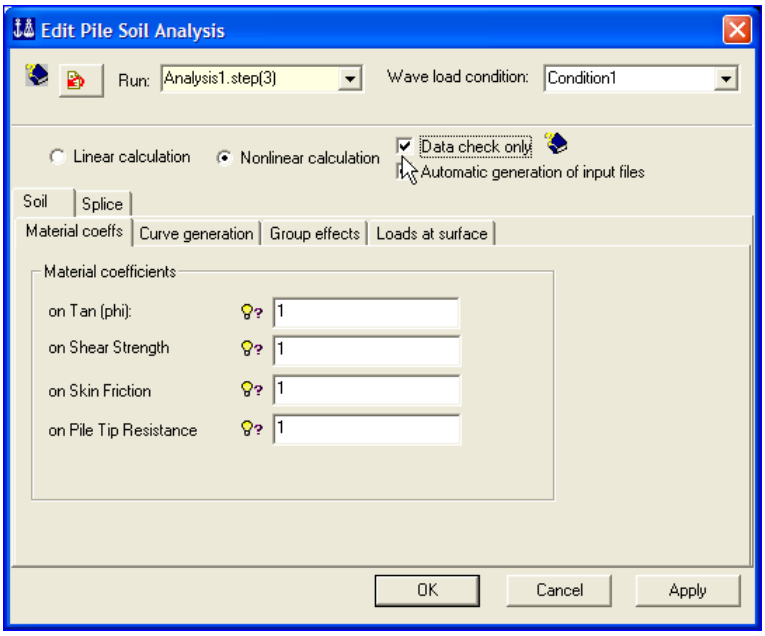


This command is scripted. Typically:

Analysis1.step(1).pileBoundaryCondition = pmFixed;

2.5.1.2.4.3 Data check only

The data check only option will run the Gensod activity, but only check the input data for the Splice analysis.



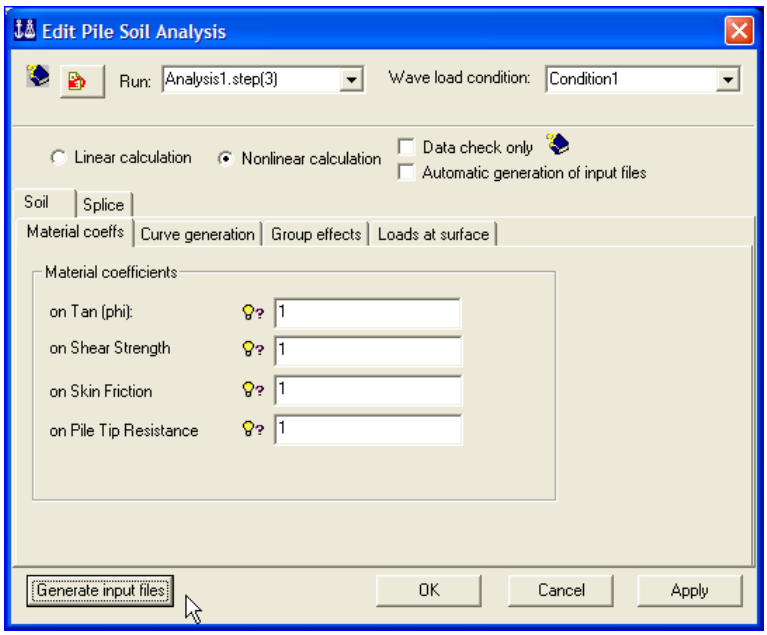
This command is scripted. Typically:

Analysis1.step(3).dataCheckOnly(true);

2.5.1.2.4.4 Automatic generation of input files

The default option ("True") implies that the input files for Gensod and Splice are automatically generated. In other words, if you modify one of the parameters this file is automatically updated prior to the next analysis run. If you want to have a manual control of the input file you should deselect this option.

In this case a *Generate input files* button appears. When you click on this button you create an input file based on the parameters defined. This file may be edited and changed. See also the previous section for linear static analysis on how to control and edit the input file.



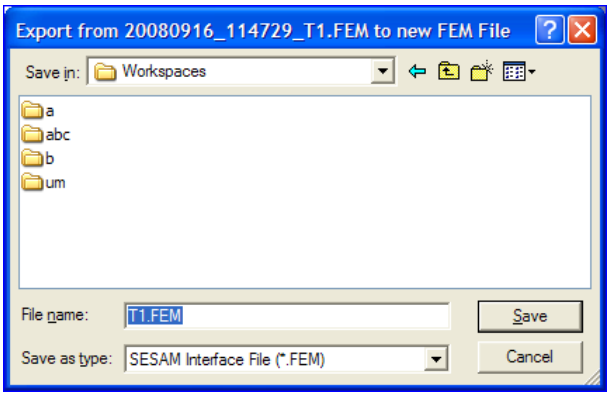
When deactivating the automatic input generation, it should be noted that changes in the environment, structure or properties should not be performed, as then the correspondence between the GeniE concept model and the Gensod and Splice input files may become out of sync and invalid.

This command is scripted. By activating the option "Automatic generation of input files", typically:

Analysis1.step(3).generateInput = false;

2.5.1.3 Export FEM

Purpose: To export an existing finite element model (FEM) to a designated area with a user given name.



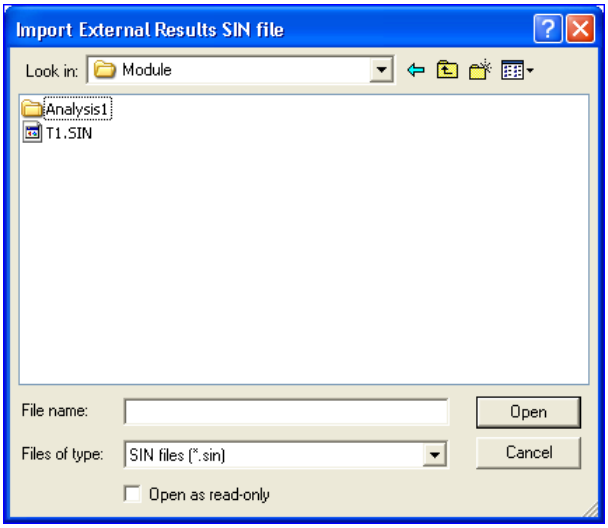
This command is not scripted. You can include the export capability by editing the journal file to typically include

FemExporter = ExportMeshFem();

FemExporter.DoExport("C:/Location/T1.FEM");

2.5.1.4 Import External Results SIN file

Purpose: To import a results file on SIN format. There are strict limitations on this functionality. See User Manual Volume 4 for documentation.



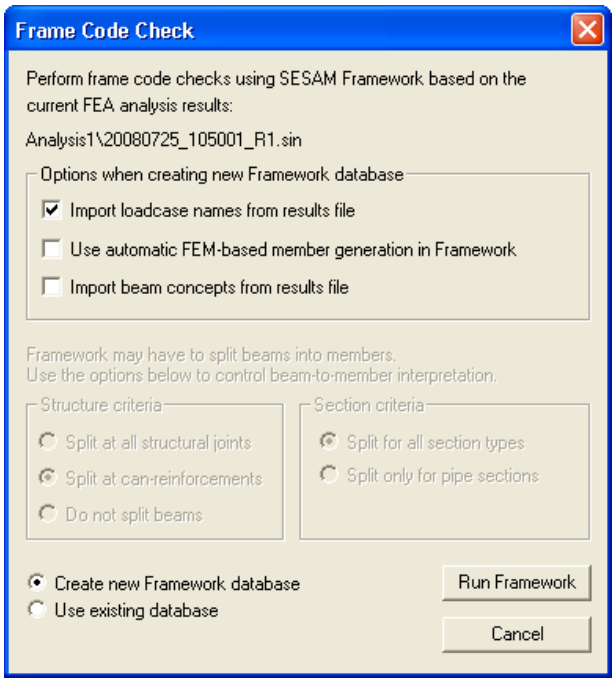
This command is scripted and a typical example is given below.

SinImporter = ImportResultsSin();

SinImporter.DoImport("C:/Location/T1.SIN");

2.5.1.5 Frame Code Check

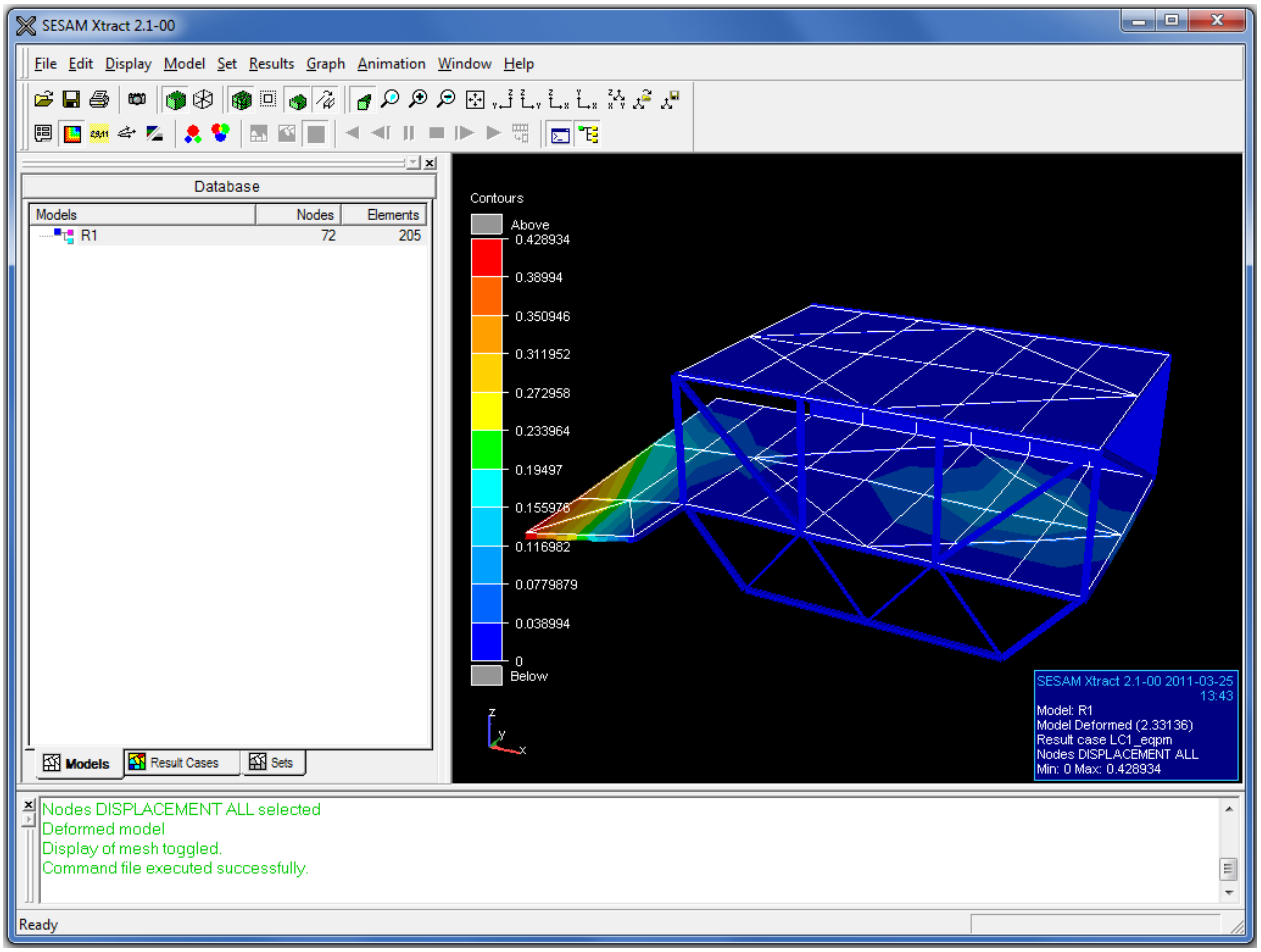
Purpose: Starting the code check program Framework and data transfer of FEM and concept data. See Appendix C of User Manual Volume 4 for more details.



This command is not scripted.

2.5.1.6 Advanced Results (Xtract)

Purpose: Starting the general post processor Xtract.

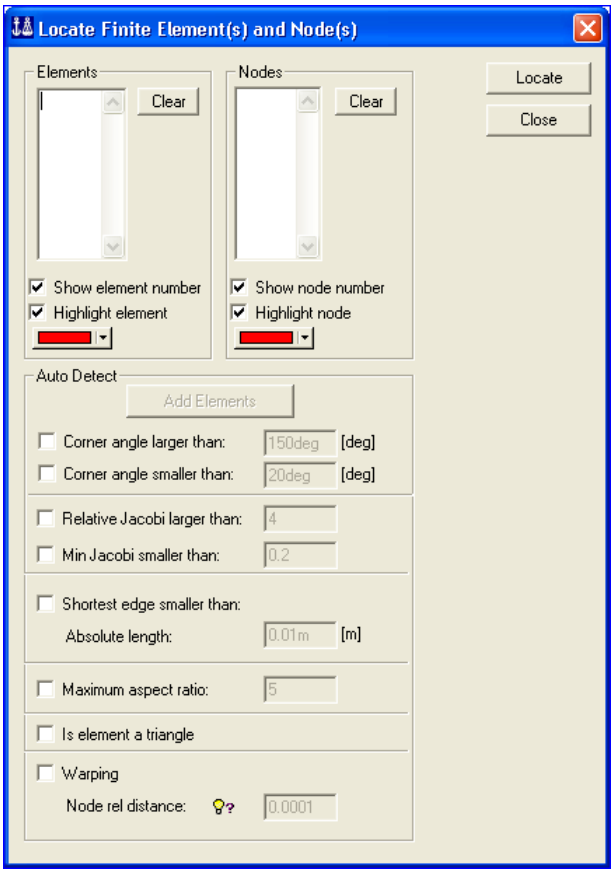


This command is not scripted.

2.5.1.7 Locate FE

Purpose: Feature allowing you to locate finite element and node numbers based on manual input of numbers or based on criterions for corner angle, relative Jacobi determinant, edges, aspect ratio, triangle element type or warping. See Section 6.1.5 of User Manual Volume 3 for more details.

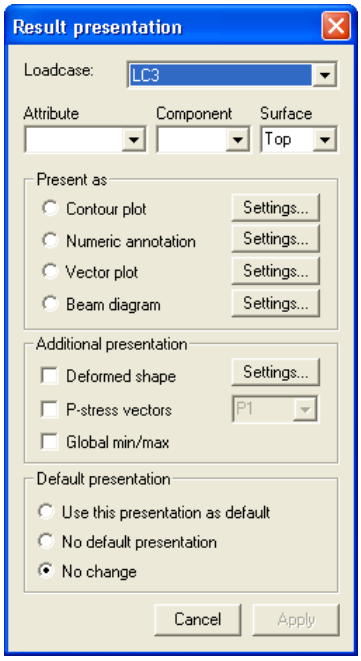
This command is not scripted.



2.5.1.8 Presentation

Purpose: A graphic driven menu on presenting various types of results, setting attributes, and changing load cases. For more details, see Section 3.14 of User Manual Volume 1 or Chapter 8 of User Manual Volume 3.

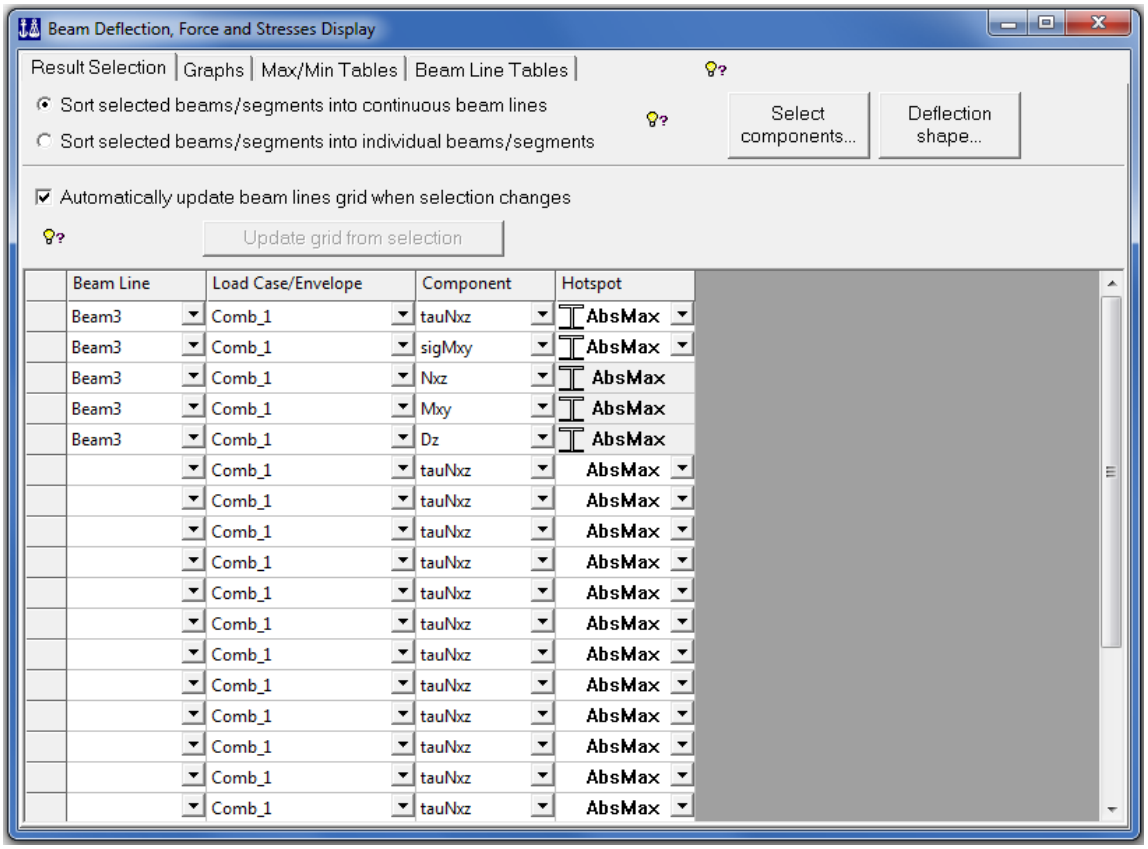
This command is not scripted.



2.5.1.9 Beam Force/Stress Diagram

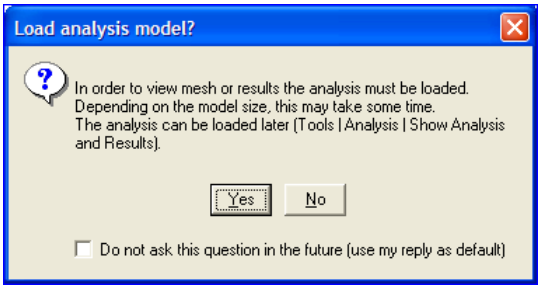
Purpose: To present and export beam force & stress data to Excel or other tools supporting the xml standard. See Chapter 4 of User Manual Volume 4 for more details.

This command is not scripted.



2.5.1.10 Show Analysis and Results

Purpose: Feature to import existing analysis results in case you choose not to load the results when opening a workspace. The picture to the right shows the pop-up menu when you open an existing workspace. In case you answer "no" you can load the finite element mesh and the analysis results by the use of the command ***Tools|Analysis|Show Analysis and Results***.

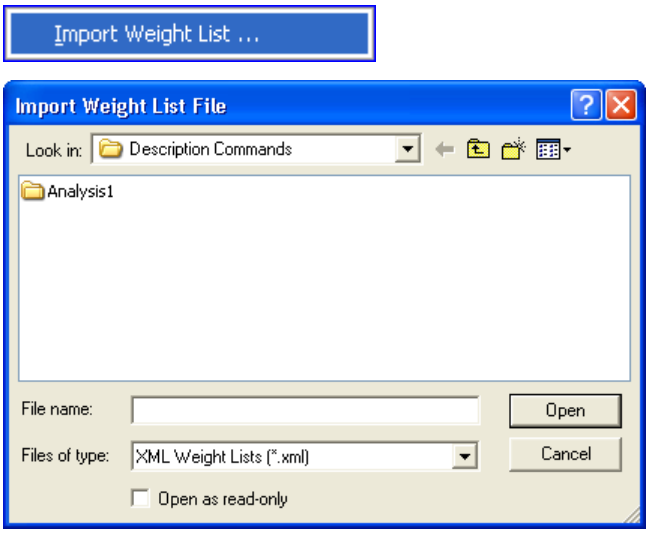


This command is not scripted.

2.5.2 Equipment

From the ***Tools|Equipment*** menu you can import weight lists.

2.5.2.1 Import Weight List



Purpose: Read in an existing weight list from an external source in XML (E**X**tendable **M**arkup **L**anguage) or CSV (**C**omma **S**eparated **V**alues) format. See Section 3.10 of User Manual Volume 1 for more information.

The data must at least contain the following information; the order on the data file does not make any difference as long as the headers are the same.

- weight_item.name : a unique name of the item (mandatory)
- weight_item description : additional description (optional)
- weight_item.weight.dry : the mass of the item (mandatory)
- weight_item.position.x/y/z : position of the item in global axis system (mandatory)
- weight_item.dimension.dx/dy/dz : dimension of the item (optional, but recommended)

Note that the weight list is dimension less and the data will be imported according to the current *input unit settings*.

This command is scripted, typically

WeightList("C:/Location/Your_weight_list.xml");

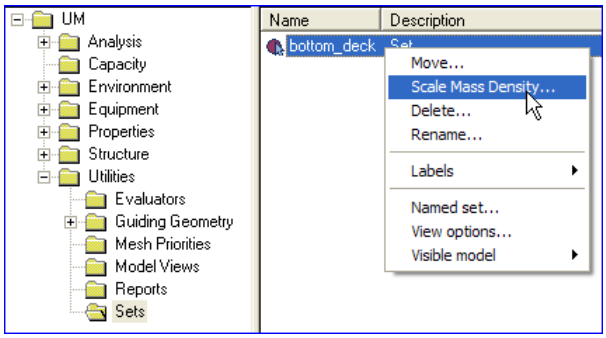
Or

WeightList("C:Location/Your_weight_list.csv");

Note that the above will import data, but not place the equipment items in individual load cases, see Section 3.10 of User Manual Volume 1 for instructions on how to do this.

2.5.3 Properties

It is possible to scale the masses to match a target mass from the context sensitive menu in the Browser Utilities Sets. This option will use mass density factors to scale the masses. I.e. no new material properties will be made.

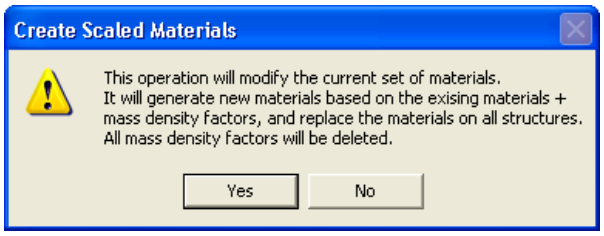


This command is scripted, typically when scaling the mass of the set "bottom_deck" with a factor of 2.6):

```
MassFactor1 = MassDensityFactor(2.6);
bottom_deck.massDensityFactor = MassFactor1;
```

2.5.3.1 Create Scaled Materials

Purpose: Make a new material and connect to beam or plate following a mass scaling operation. See Section 3.16.5 of User Manual Volume 1 for more details.

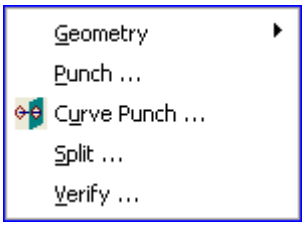


This command is scripted, typically when making new materials affected of the mass scaling factor 2.6 (in this case the material St48 is affected – a new material St48_2_6 is made and applied to the structural parts of the set):

```
St48_2_6 = St48.copy();
St48_2_6.density = St48.density * 2.6;
Bm1.material = St48_2_6;
Bm2.material = St48_2_6;
Bm3.material = St48_2_6;
```

2.5.4 Structure

The ***Tools|Structure*** pulldown menu contains features for cleaning up the geometry, punch operations, split structural parts and a verification of the structure.

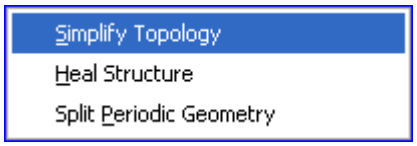


2.5.4.1 Geometry

This option gives access to features for simplify topology, heal structure and split periodic geometry.

2.5.4.1.1 Simplify Topology

Purpose: Remove un-necessary topology lines or points introduced by previous plate/plate or plate/beam or plate/feature edge intersections. See also Section 3.3.6 of User Manual Volume 3.

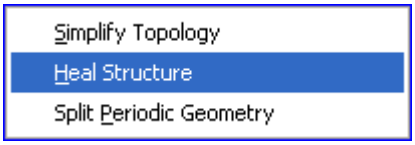


This command is scripted, typically:

SimplifyTopology();

2.5.4.1.2 Heal Structure

Purpose: Feature for reconnection of bad beam models. Model healing analyses each beam in the model and searches for candidates that should intersect the beam in question. To do this it is necessary to categorize the beams. The healing operation assumes that a beam which beam ends touch the interior of another beam is categorized as less important than the beam it touches. For more details, see Section 3.6.9 of User Manual Volume 1.



This command is scripted, typically based on a selection of 10 beams:

HealStructureSet = Set();

HealStructureSet.add(Bm1);

HealStructureSet.add(Bm2);

HealStructureSet.add(Bm3);

HealStructureSet.add(Bm4);

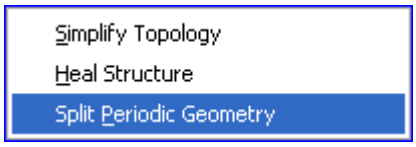
.....

HealStructureSet.add(Beam10);

HealStructure(HealStructureSet);

Delete(HealStructureSet);

2.5.4.1.3 Split Periodic Geometry

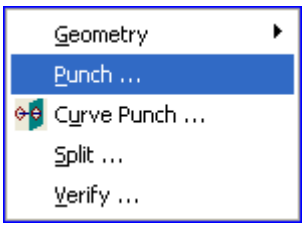


Purpose: Insert an edge along the surface in the longitudinal direction of a periodic geometry like for example a tube. Tubular surfaces may be made from a sweeping operation where a 360 degree guideline is used as reference. In some cases GeniE is not able to make a finite element mesh on such surfaces. In these rare events, you can use this feature to insert an edge along the surface in the longitudinal direction. For more details, see Section 6.2.1.2 of User Manual Volume 3.

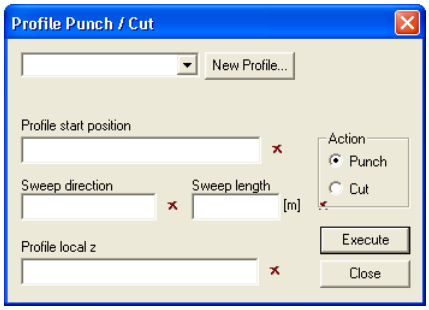
This command is scripted, typically:

SplitPeriodicGeometry();

2.5.4.2 Punch



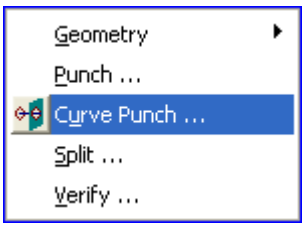
Purpose: Create holes or cut lines in the structure using a user-defined profile. The profile can be defined from the dialogue box or from ***Insert|Profile***. By using the "Punch" option plate and stiffener material will be removed from the model. The "Cut" option will create cut lines where the profile intersects the model. Please see Section 3.3.1.11 of User Manuals for details on how to do both operations.



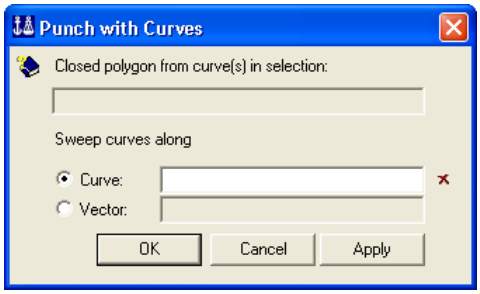
This command is scripted, typically when using profile "Man-hole" at start position (1m, 1m, 1m) in negative x-direction. Furthermore the profile coordinate system is aligned with the global y-direction and the length of the cut profile is 5 meters:

Man_hole.punch(Point(1 m,1 m,1 m),Vector3d(-1,0,0), Vector3d(0,1,0),5);

2.5.4.3 Curve Punch



Purpose: Remove all structure within a volume as defined by a selection of curves that form a closed loop and a given vector (or a curve). For more details, see Section 3.3.1.10 of User Manual Volume 3.

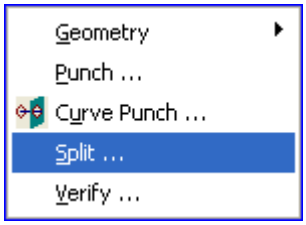


This command is scripted, typically when using curves Curve1 -> Curve 4 (making a closed loop) and a vector of 10 meters in negative z-direction:

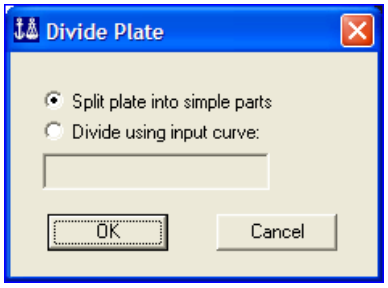
PunchWithCurves(Array(Curve1,Curve2,Curve3,Curve4), Vector3d(0,0,-10));

2.5.4.4 Split

Purpose: Split selected beams and/or plates to minor parts based on a) intersection with other beams, plates or feature edges or b) intersection with a guide curve.



This command is scripted, typically when dividing plate Pl1 into 3 minor plates:



Validate(Pl1.primitivePartCount == 8);

Pl1.explode(CloneNameMask());

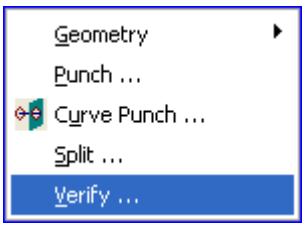
Validate(Pl1_1);

Validate(Pl1_2);

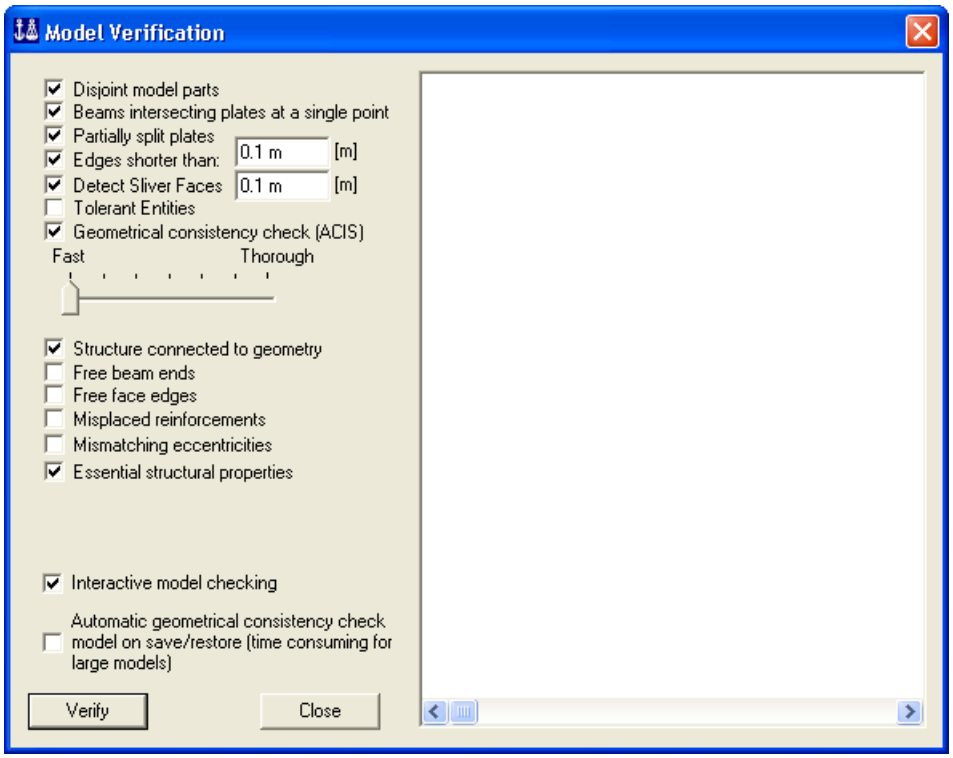
Validate(Pl1_3);

2.5.4.5 Verify

Purpose: To verify the concept model. This feature is described in Section 3.3.7 of User Manual Volume 3.

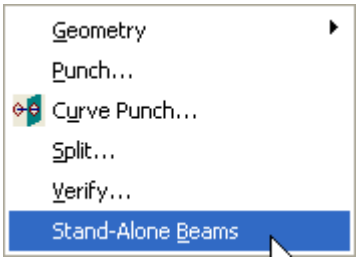


This command is not scripted.



2.5.4.6 Stand-Alone Beams

Purpose: To identify beams that are not fully attached to any shell. This feature is described in Section 3.3.8 of User Manual Volume 3.



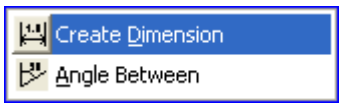
This command is not scripted.

2.5.5 Dimension

Find length between points or angle between beams

2.5.5.1 Create Dimension

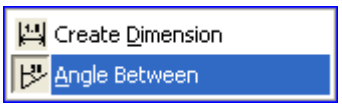
Purpose: Find distance between two points and present the value graphically.



This command is not scripted.

2.5.5.2 Angle Between

Purpose: Find angle between two beams and present value graphically.



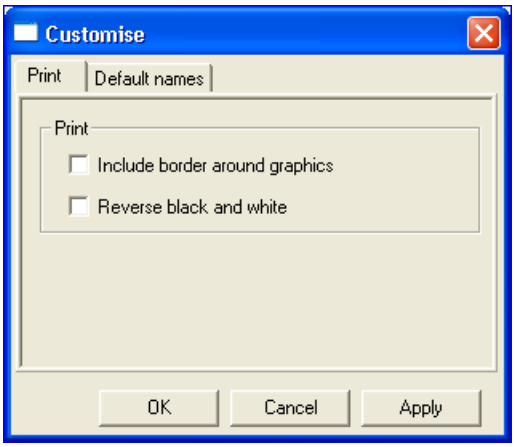
This command is not scripted.

2.5.6 Customise

The customize pulldown menu allows you to specify options for how to print pictures and define settings for default names.

2.5.6.1 Print

Purpose: To decide whether to include a border around graphics and/or reverse black and white background when printing a picture. Notice that these settings do not apply for picture generations.

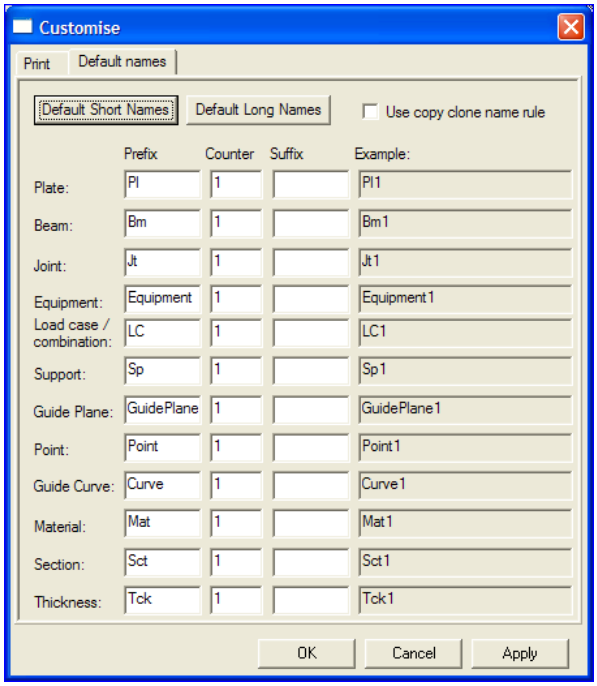


This command is not scripted.

2.5.6.2 Default names

Purpose: To specify a default naming schema. See Section 3.1.9 of User Manual Volume 1 for more details.

The program default names are shown to the right.



This command is scripted. By modifying the prefix, counter value and suffix as shown to the right, the script commands are:

DefaultName(typeFlatPlate,"MyPL",10,"UM");

DefaultName(typeStraightBeam,"MyBm",10,"UM");

DefaultName(typeFrameJoint,"MyJt",10,"UM");

DefaultName(typeEquipment,"MyEquipment",10,"UM");

DefaultName(typeBasicState,"MyLC",10,"UM");

DefaultName(typeSupportPoint,"MySp",10,"UM");

DefaultName(typePlane,"MyGuidePlane",10,"UM");

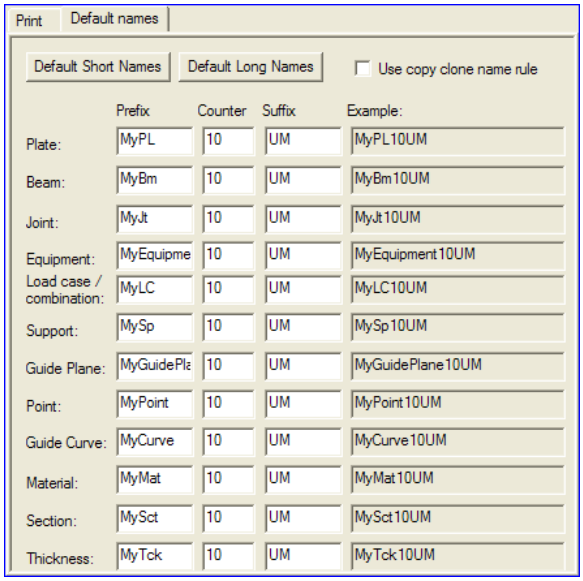
DefaultName(typePoint,"MyPoint",10,"UM");

DefaultName(typeGuideCurve,"MyCurve",10,"UM");

DefaultName(typeMaterial,"MyMat",10,"UM");

DefaultName(typeSection,"MySct",10,"UM");

DefaultName(typeThickness,"MyTck",10,"UM");



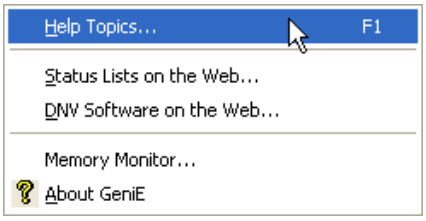
2.6 The Help pulldown menu

Get help online from the GeniE help pages, open DNV on web or list copyrights and 3rd party software used

2.6.1 Help Topics

Purpose: Get help online from GeniE's help pages in user manuals, videos or tutorials.

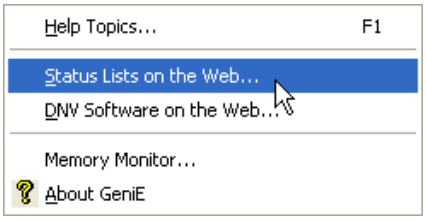
This command is not scripted.



2.6.2 Status Lists on the Web

Purpose: Starts the Status List program and looks up GeniE items.

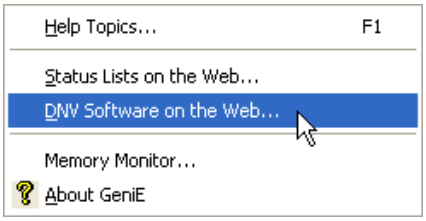
This command is not scripted.



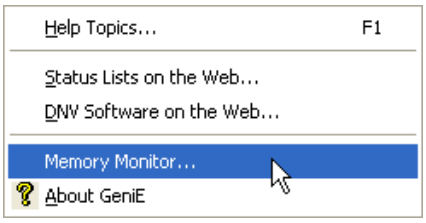
2.6.3 DNV Software on the Web

Purpose: Start-up of DNV Software address on web.

This command is not scripted.

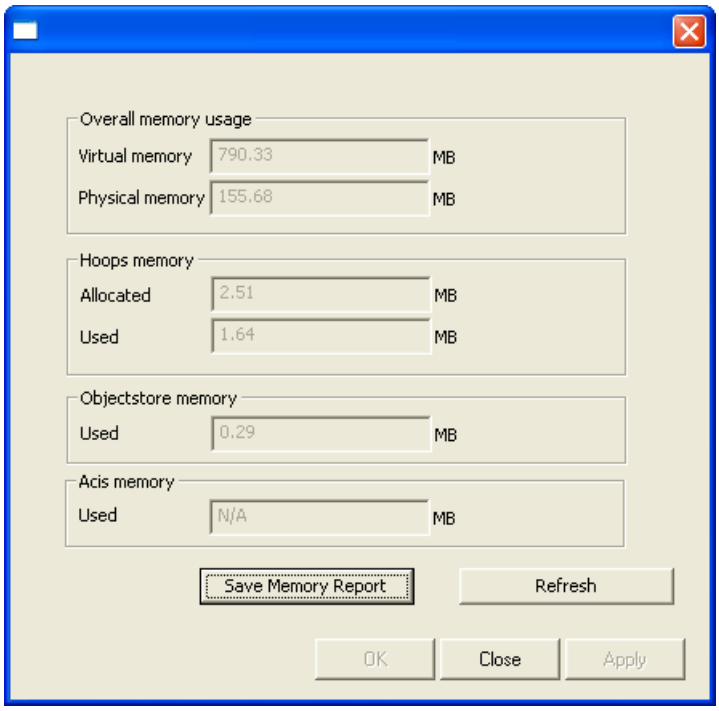


2.6.4 Memory Monitor



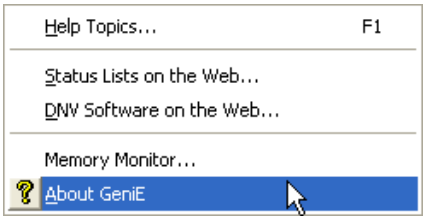
Purpose: Shows the memory consumption of the current workspace, divided into the different categories that GeniE is using.

If you are working on the limits of your system's available memory, or if you are close to the maximum memory GeniE can utilize, it is a good idea to keep an eye on the memory usage listed in the memory monitor.



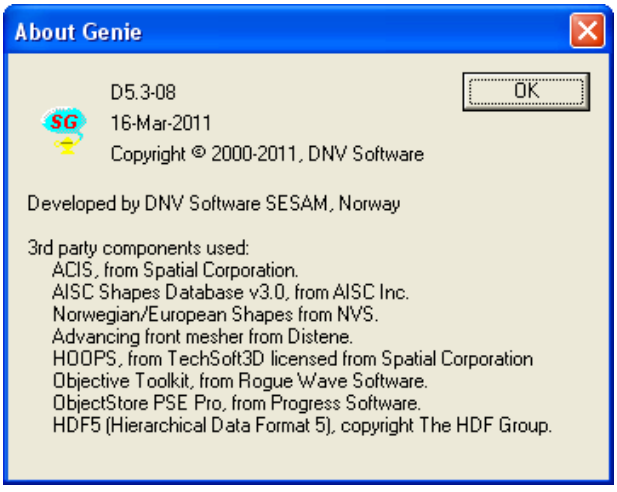
This command is not scripted.

2.6.5 About GeniE



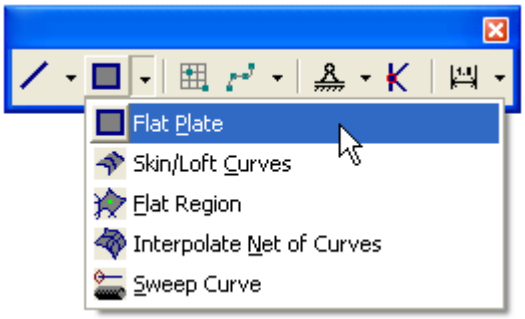
Purpose: Lists 3rd party software used in GeniE.

This command is not scripted.



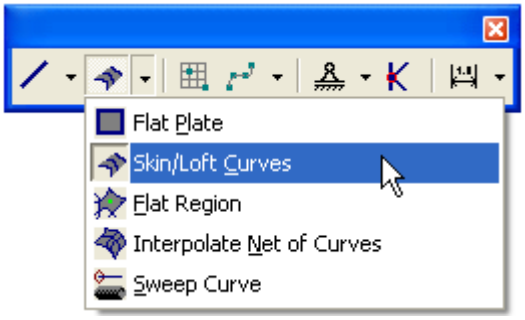
Tool buttons

There are a number of tool buttons that give access to the most commonly used commands. They may be dynamic, meaning that the active tool button may give access to different commands. A dynamic tool button is recognised by having a pulldown arrow embedded on the GUI. One example is the tool button for Insert Plate. It can give access to several commands, like inserting a flat plate or inserting a curved plate using a skin/loft curve operation.



Flat Plate option selected

Flat Plate is the active operation of the tool button



Skin/Loft Curves option selected

Skin Curves is the active operation of the tool button

The other available choices behaves in the same way when it comes to selecting and setting them active.

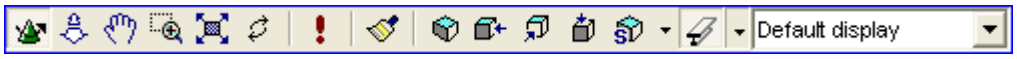
The tool buttons are grouped in eight Toolbars described below. All tool buttons have tool tips, these are listed in **bold**.

2.7 The Main toolbar

Tool button	*Tool tip*	*Description*
	New Workspace	Create a new workspace

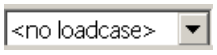
	Open Workspace	Open existing workspace
	Save Workspace	Save current workspace
	Copy with transform	Copy selection using transformation
	Delete	Delete selection
	About	Explains GeniE program version and subcontractors

2.8 The View Manipulation toolbar



Tool button	*Tool tip*	*Description*
	Rotate	Rotation in all 3 degrees of freedom (use right mouse button RMB)
	Zoom	Zoom in or zoom out (RMB). Place pointer to decide where to zoom. Moving mouse upwards or to the right mean zoom-in, while down or to the left means zoom-out
	Pan	Move model to desired position on display (RMB)
	Zoom rubberband	Create a rubber band to zoom in (RMB)
	Fit	Automatic scale view so that whole model is shown on display
	Spin	Remembers the last rotation and speed of it and makes this a continuous spin
	Refresh graphics	Cleans up graphics and remove all dimensions
	Color code	Toggle on/off your selection of labelling
	Iso view	View from isometric point
	View from X	View in negative X-direction
	View from Y	View in positive Y-direction
	View from Z	View in negative Z-direction
	Save Model View	Save and retrieve a model view
	Outline view	Display beams in outline view (3D view, but no thickness)
	Wireframe view	Display beams in wireframe view
	Solid view	Display beams in solid view (3D view including thickness)
Default display	**Display configuration**	Select the current display configuration. There are 10 system configurations. Additional display configurations can be made from ***View|Options***.

2.9 The Loadcase toolbar

Tool button	*Tool tip*	*Description*
	**Default loadcase selection**	Set loadcase to current and display current loadcase

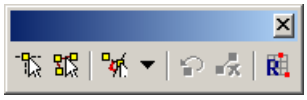
2.10 The Labels toolbar

Tool button	*Tool tip*	*Description*
	**Label**	Labels selected object (Coordinates, Diagram value, Material, Name, Section, Thickness)

2.11 The Object Types toolbar

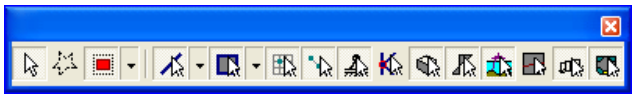
Tool button	*Tool tip*	*Description*
	Straight beam	Insert a straight beam between two snap points
	Straight overlapping beam	Insert overlapping beams between two snap points
	Pile	Insert a pile between an elevation and snap point along a beam axis
	Curved Beam	Insert a curved beam between three or more snap points
	Flat plate	Insert a plan plate between four or more snap points
	Skin/loft curves	Skinning: Insert a curved shell between two or more curved lines Lofting is a variant of skinning involving both shells and curves to control the shape of the new surface to be created.
	Curve-net Interpolation	Insert a curved shell interpolating two or more arrays of N and M curved lines, which form a set of (N-1)x(M-1) rectangular patches.
	Sweep Curves	Insert a straight or curved shell by sweeping a curve along another curve or vector
	Guide plane	Insert a guide plane between four snap points. The guideplane will be created having 5 equal relative spacings
	Guide point	Insert a guide point by clicking one snap point
	Guide line	Insert a guide line between two snap points
	Guide spline	Insert a guide curve (spline) between three or more snap points
	Poly Curve	Insert a poly curve with three or more snap points
	Guide elliptic arc	Insert a guide ellipse with three snap points (origin, start, end)
	Guide circle	Insert a guide circle with three snap points(origin radius, plane)
	Model curve	Insert a model curve along a topological edge with start and stop points
	Fillet curves	Insert a part of a circle with tangents to two straight lines
	Support point	Insert a support point
	Support curve	Insert a support curve along beam, model curve, or guide lines
	Joint	Insert a single joint
	Dimension	Find and display the length between two points (1st and 2nd click on points, 3rd click to position the length on the display)
	Angle between	Find and display angle between two beams (1st click on 1st beam, 2nd click on 2nd beam)

2.12 The Create Methods Toolbar



Tool button	*Tool tip*	*Description*
	Snap point	Positioning beam, plate, guideplane, support points etc. one by one
	Snap point loop	Positioning beam, plate, guideplane, support points etc. sequentially, e.g. end of beam is start of next beam
	Snap perpendicular	Insert a beam perpendicular to another, 1st click is from start point, 2nd click on perpendicular beam
	Snap tangential	Insert a guideline tangential to another curve, 1st click is from start point, 2nd click on curve
	Snap plane	Temporary snap points are defined at the intersection between beams and a snap plane
	Snap eccentric	Connect a beam to another beam end using its eccentric position.
	Undo snap point	Undo selection of previous snap point
	Clear snap points	Undo selection of all snap points
	Reference point modelling	Specifies journalling of reference point modelling on the journal file

2.13 The Selection Toolbar

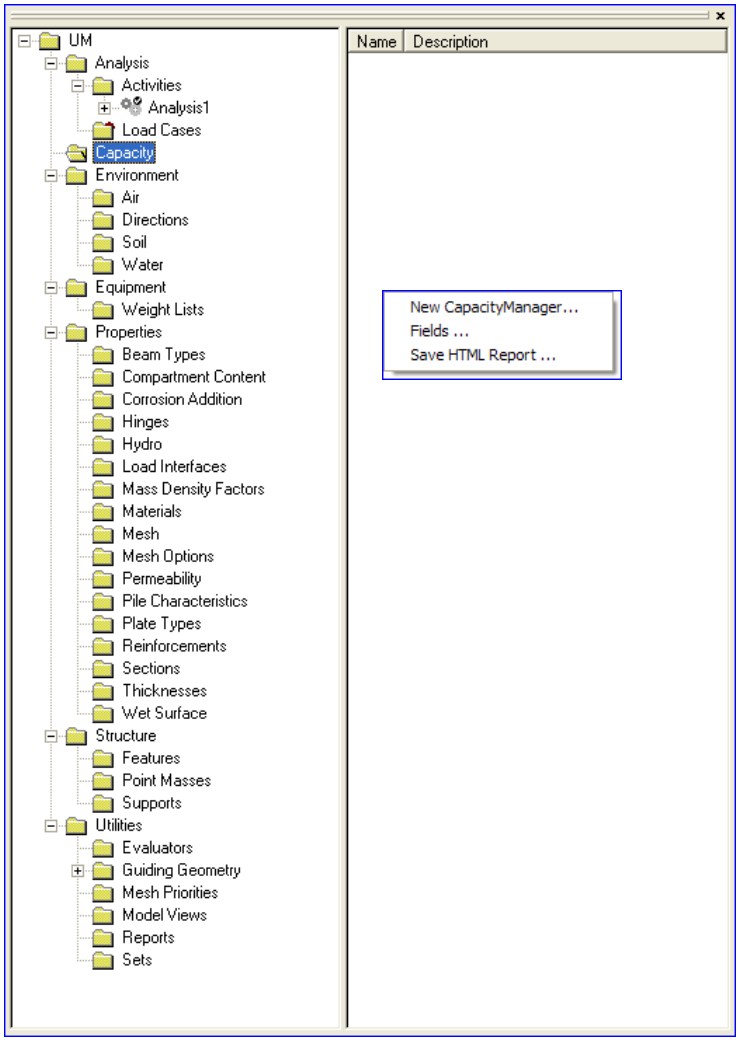


Tool button	*Tool tip*	*Description*
	Selection	Method for selecting one by one or rubberband (LMB). Together with Shift more advanced selections can be made
	Polygon select	Method for advanced selection where you can make an arbitrary select area (LMB). Make sure that you make a closed envelope.
	Enclosed by rubberband	Objects need to be fully enclosed by rubberband to be part of selection
	Touched by rubberband	Objects need to be touched by rubberband to be part of selection
	Select visible	Visible objects only selected
	Filter beam	Toggle on/off for selection of beams
	Filter segment	Toggle on/off for selection of segmented members
	Plate selection on/off	Toggle on/off for selection of plates
	Filter side	Toggle on/off for selection of one side of a plate. Used when applying wet surface for panel modelling
	Guide selection on/off	Toggle on/off for selection of guide planes
	Guide curve selection on/off	Toggle on/off for selection of guide curves
	Support selection on/off	Toggle on/off for selection of support (points and curves)
	Joint selection on/off	Toggle on/off for selection of joints
	Equipment selection on/off	Toggle on/off for selection of equipment
	Diagram selection on/off	Toggle on/off for selection of load diagrams
	Environment selection on/off	Toggle on/off for selection of environment
	Feature edge selection on/off	Toggle on/off for selection of feature edge
	Load selection on/off	Toggle on/off for selection of loads
	Compartment selection on/off	Toggle on/off for selection of compartments

3. THE BROWSER MENU

This Chapter describes how the browser menu works with particular focus on commands only available from the browser; the capacity manager and the commands necessary to define the environment. For all of these commands there are examples on the scripting as well. The other commands are available from the *Tools Pulldown Menu* and the scripting is documented in the previous Chapter.

The browser has a left pane and a right pane window. To access the menu's available from the menu right click **(RMB)** the browser folder or click on the browser folder and right click in the right pane window (see the example to the right).



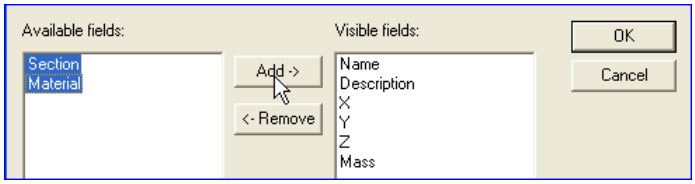
Common for all commands is that it is possible to modify the "fields" and that the content of a browser folder can be opened in an html view.

These commands are not listed in the following.

The example below shows the program defaults for the browser folder "Equipments"

Name	Description	X [m]	Y [m]	Z [m]	Mass [tonne]
Flare	Prism Equipment				
Helideck	Prism Equipment				
Weight Lists	Folder				

By modifying the "Fields" to also include "Section" and "Material" the browser content is changed:



Name	Description	X [m]	Y [m]	Z [m]	Mass [tonne]	Section	Material
Flare	Prism Equipment					<None>	<None>
Helideck	Prism Equipment					<None>	<None>
Weight Lists	Folder						

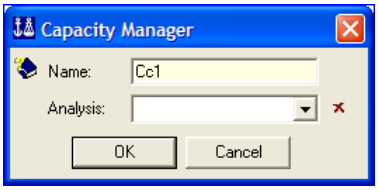
3.1 Analysis

1st level browser	*2nd level browser*	*3rd level browser*	*Available commands*
Activities			***New Analysis***
	<Activity Name>		***Activity Monitor***
			Edit Analysis
			New Loadcase
			New Load Combination
			Set Active
			Paste
			Delete
			Rename
		<Activity Name>.step(n)	***Edit Activity(n)***
Load Cases			***New Loadcase***
			New Load Combination
			Set Active
			Paste
	<Loadcase name>		***Set Current***
			Generate Applied Loads
			Recompute Load Sums
			New Load Combination
			Properties
			Copy
			Paste
			Delete
			Rename

3.2 Capacity

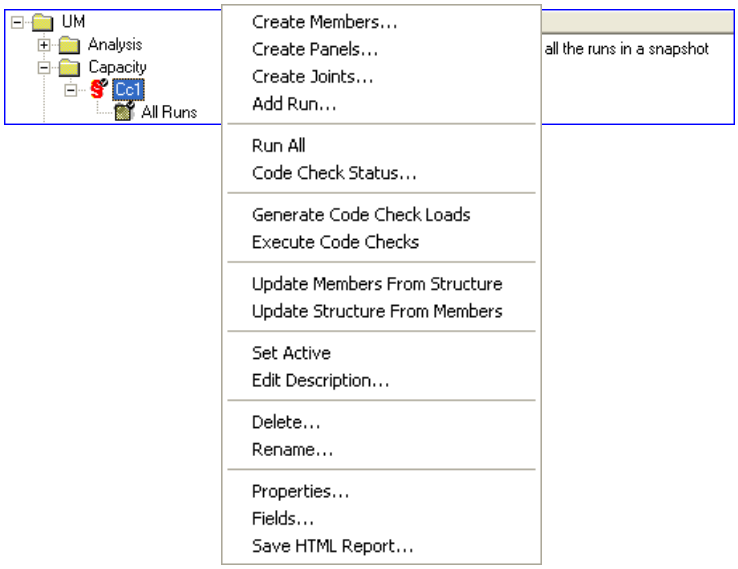
These commands are only available from the *Browser Menu*. Examples of the script commands are thus also listed in this Section. For more details on member and joint capacity checking, see Chapter 3 of User Manual Volume 4. For plate checking according to CSR Bulk rules, see Chapter 2 of User Manual Volume 5.

The content of the Capacity Folder is created when you right click the folder and define a new capacity manager.



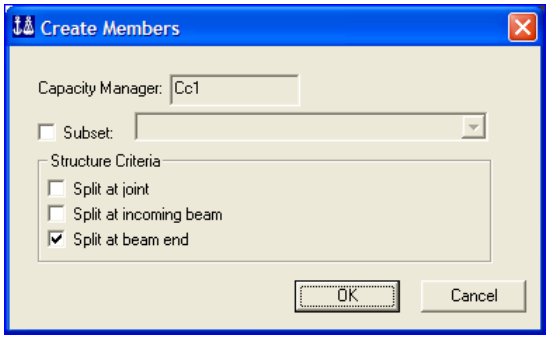
This command is scripted. Typically when using "Analysis1" as basis for the capacity manager:

Cc1 = CapacityManager(Analysis1);



3.2.1 Create Members

Purpose: To define members for use in a member check. It is possible to make members of parts of the structure by referring to "Subsets" (or "Named Sets"). A continuous structural beam may be split at joints, at incoming beams or at beam ends to form the capacity members.



This command is scripted. Typically when using the options "Split at incoming beam" and referring to the subset "Row_1":

MemberCreationOpts = MemberCreationOption();

MemberCreationOpts.splitAtJoint = false;

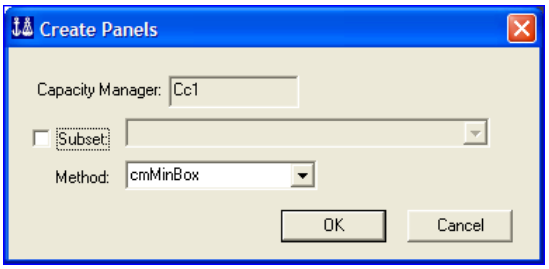
MemberCreationOpts.splitAtIncomingBeam = true;

MemberCreationOpts.splitAtBeamEnd = false;

Cc1.createMembers(Row_1, MemberCreationOpts);

3.2.2 Create Panels

Purpose: To define panels for use in a plate check. It is possible to make panels of parts of the structure by referring to "Subsets" (or "Named Sets"). A panel may be defined using the option "cmMinBox" or "cmMaxAreaMoment".

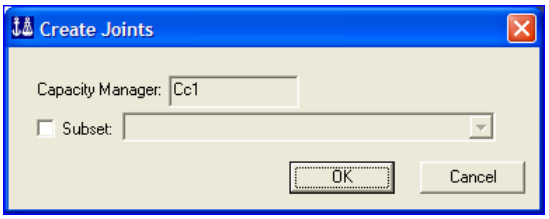


This command is scripted. Typically when using the options "cmMinBox":

Cc1.createPanels(cmMinBox);

3.2.3 Create Joints

Purpose: To define structural joints that is part of a tubular joint check. It is required that the joints are defined as joints (***Insert|Joint***) prior to the analysis.

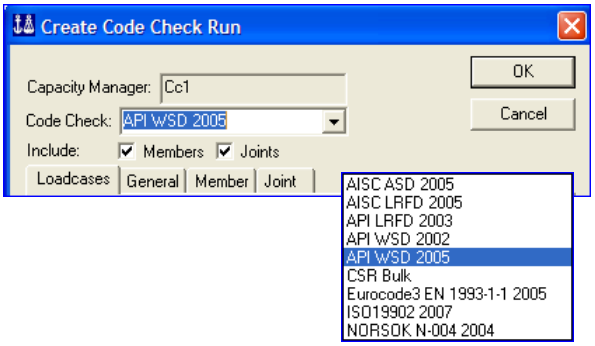


This command is scripted. Typically:

Cc1.createJoints();

3.2.4 Add Run

Purpose: To define which code check standard to use and which elements to include in a code check. The available code check standards are shown to the right.



This command is scripted. Typically when using API WSD 2005:

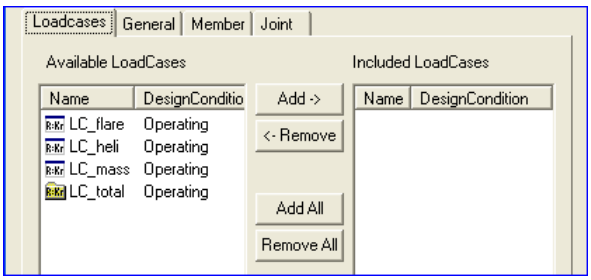
Cc1.AddRun(ApiWsdRun2005());

Cc1.run(1).includeMembers = true;

Cc1.run(1).includeJoints = true;

3.2.4.1 Loadcases

Purpose: To define the load cases to be part of a code check run (the load cases as shown to the right is an example).



This command is scripted. Typically when using the load case "LC_Total" only:

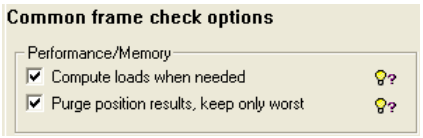
Cc1.run(1).addLoadCase(LC_total);

When removing a load case from the run the typical script command is:
Cc1.run(1).removeLoadCase(LC_total);

3.2.4.2 General

Purpose: To define global and general code checking factors.

Some general factors are common for most frame code checks.



This command is scripted, typically:

Cc1.run(1).generalOptions.computeLoadsAsNeeded = true;

Cc1.run(1).generalOptions.purgePositionResults = true;

Most factors are different for the various code checking standards. They are listed in the following paragraphs.

3.2.4.2.1 AISC ASD 2005

Purpose: To define global and general code checking factors according to AISC ASD 2005.

This command is scripted, typically:

Cc1.AddRun(AiscAsdRun());

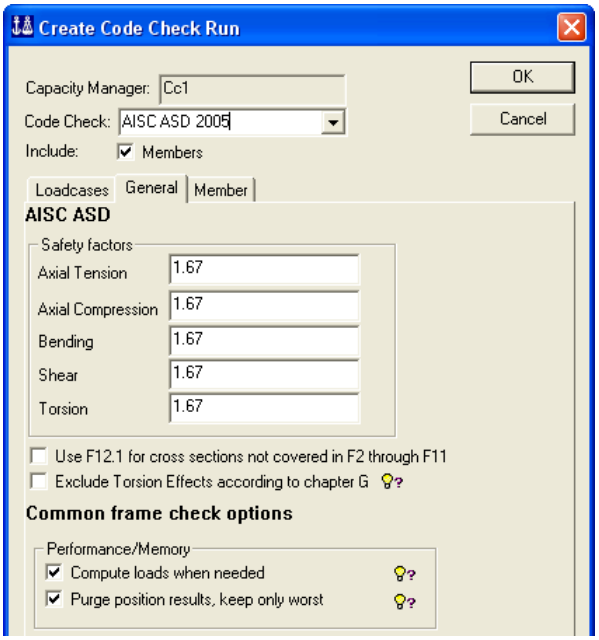
Cc1.run(1).generalOptions.tensionFactor = 1.67;

Cc1.run(1).generalOptions.torsionFactor = 1.67;

Cc1.run(1).generalOptions.shearFactor = 1.67;

Cc1.run(1).generalOptions.bendingFactor = 1.67;

Cc1.run(1).generalOptions.compressionFactor = 1.67;



3.2.4.2.2 AISC LRFD 2005

Purpose: To define global and general code checking factors according to AISC LRFD 2005.

This command is scripted, typically:

Cc1.AddRun(AiscLrfdRun());

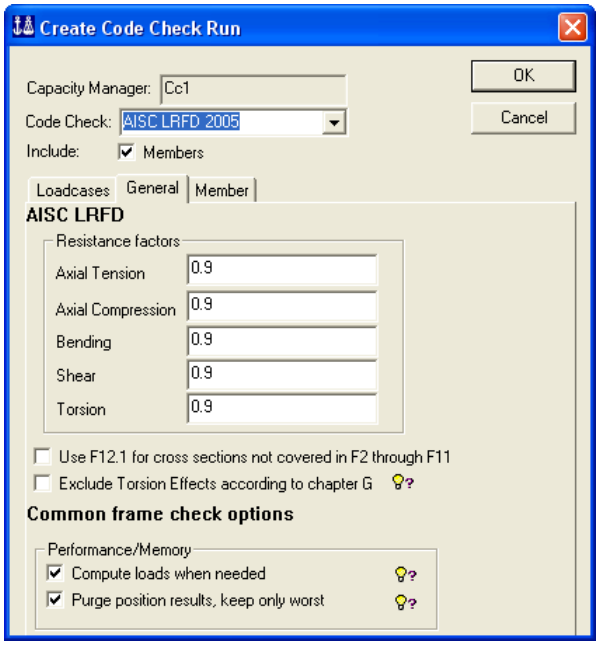
Cc1.run(1).generalOptions.torsionFactor = 0.9;

Cc1.run(1).generalOptions.shearFactor = 0.9;

Cc1.run(1).generalOptions.bendingFactor = 0.9;

Cc1.run(1).generalOptions.tensionFactor = 0.9;

Cc1.run(1).generalOptions.compressionFactor = 0.9;



3.2.4.2.3 API LRFD 2003

Purpose: To define global and general code checking factors according to API LRFD 2003. The factors prevail for members and joints.

This command is scripted, typically:

Cc1.AddRun(ApiLrfdRun());

Cc1.run(1).generalOptions.RFPipeXPress = 0.8;

Cc1.run(1).generalOptions.RFPipeShear = 0.95;

Cc1.run(1).generalOptions.RFPipeBend = 0.95;

Cc1.run(1).generalOptions.RFPipeTens = 0.95;

Cc1.run(1).generalOptions.RFPipeComp = 0.85;

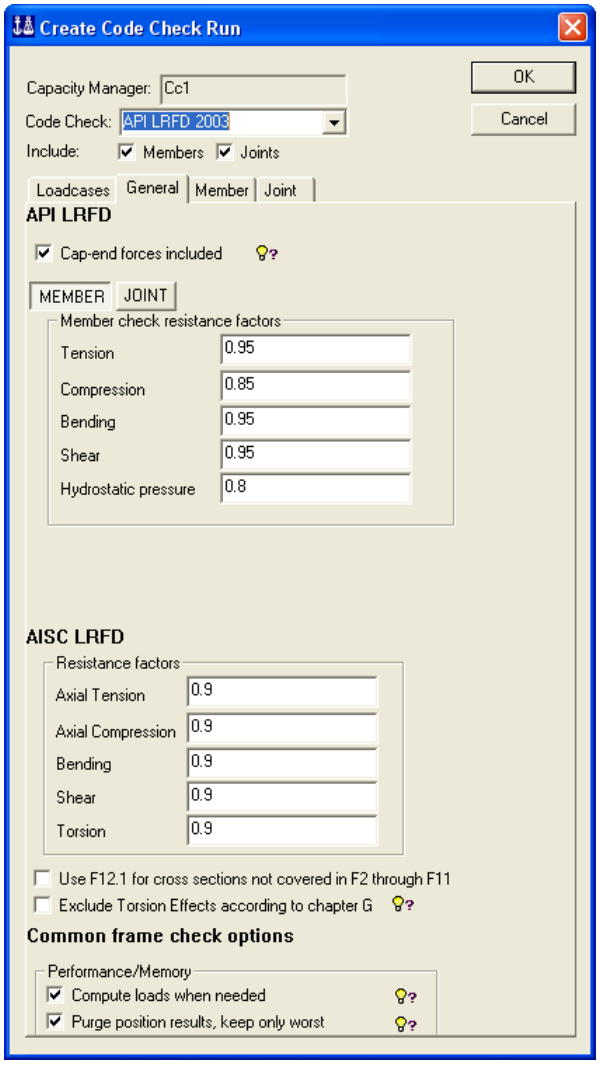
Cc1.run(1).generalOptions.aisc.torsionFactor = 0.9;

Cc1.run(1).generalOptions.aisc.shearFactor = 0.9;

Cc1.run(1).generalOptions.aisc.bendingFactor = 0.9;

Cc1.run(1).generalOptions.aisc.tensionFactor = 0.9;

Cc1.run(1).generalOptions.aisc.compressionFactor = 0.9;



Cc1.run(1).generalOptions.RFJointKTens = 0.95;

Cc1.run(1).generalOptions.RFJointKComp = 0.95;

Cc1.run(1).generalOptions.RFJointKipb = 0.95;

Cc1.run(1).generalOptions.RFJointKopb = 0.95;

Cc1.run(1).generalOptions.RFJointYTens = 0.9;

Cc1.run(1).generalOptions.RFJointYComp = 0.95;

Cc1.run(1).generalOptions.RFJointYipb = 0.95;

Cc1.run(1).generalOptions.RFJointYopb = 0.95;

Cc1.run(1).generalOptions.RFJointXTens = 0.9;

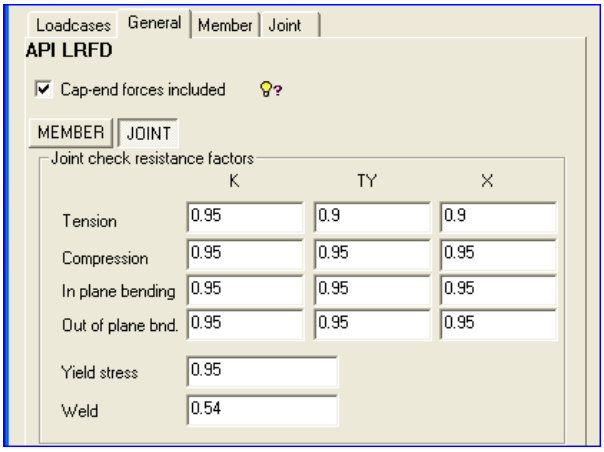
Cc1.run(1).generalOptions.RFJointXComp = 0.95;

Cc1.run(1).generalOptions.RFJointXipb = 0.95;

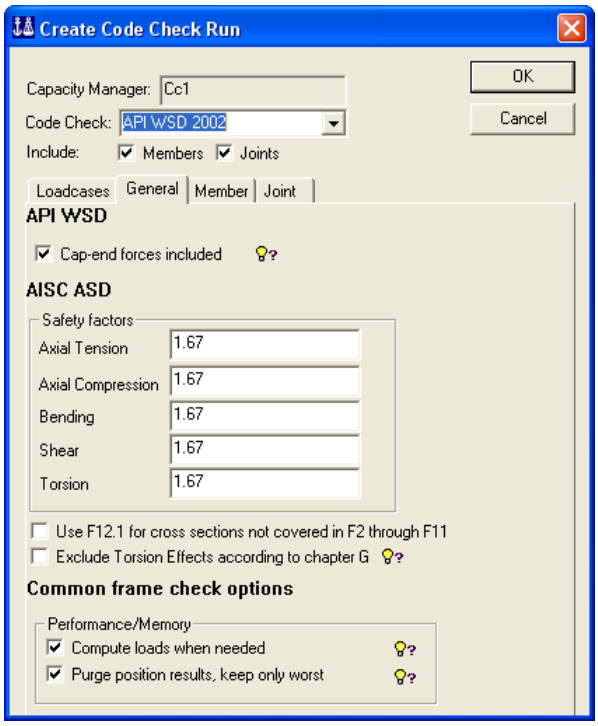
Cc1.run(1).generalOptions.RFJointXopb = 0.95;

Cc1.run(1).generalOptions.RFJointYield = 0.95;

Cc1.run(1).generalOptions.RFJointWeld = 0.54;



3.2.4.2.4 API WSD 2002



Purpose: To define global and general code checking factors according to API WSD 2002.

This command is scripted, typically:

Cc1.AddRun(ApiWsdRun());

Cc1.run(1).generalOptions.aisc.torsionFactor = 1.67;

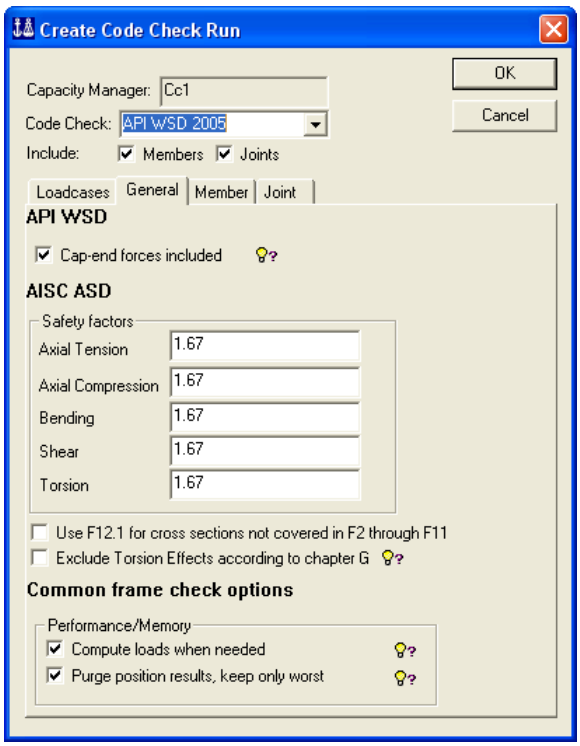
Cc1.run(1).generalOptions.aisc.shearFactor = 1.67;

Cc1.run(1).generalOptions.aisc.bendingFactor = 1.67;

Cc1.run(1).generalOptions.aisc.tensionFactor = 1.67;

Cc1.run(1).generalOptions.aisc.compressionFactor = 1.67;

3.2.4.2.5 API WSD 2005



Purpose: To define global and general code checking factors according to API WSD 2005.

This command is scripted, typically:

Cc1.AddRun(ApiWsdRun2005());

Cc1.run(1).generalOptions.aisc.torsionFactor = 1.67;

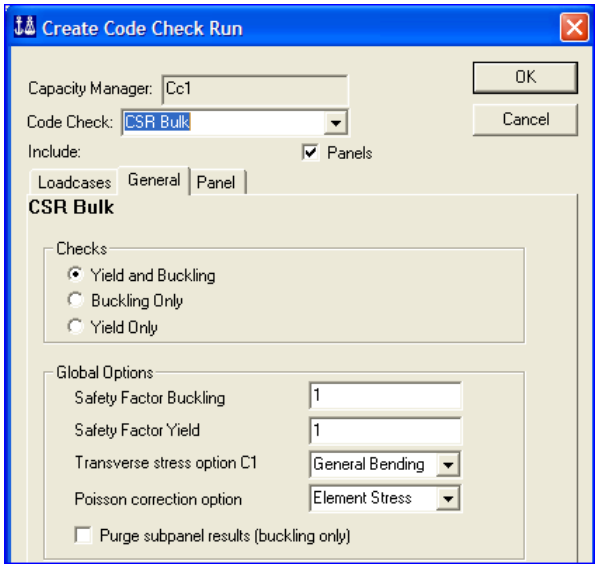
Cc1.run(1).generalOptions.aisc.shearFactor = 1.67;

Cc1.run(1).generalOptions.aisc.bendingFactor = 1.67;

Cc1.run(1).generalOptions.aisc.tensionFactor = 1.67;

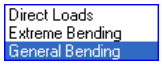
Cc1.run(1).generalOptions.aisc.compressionFactor = 1.67;

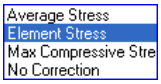
3.2.4.2.6 CSR Bulk



Purpose: To define global and general code checking factors according to CSR Bulk. The checks can be combined yield and buckling or separate buckling and yield.

The options for Transverse stress option C1 and Poisson Correction option are shown below.





This command is scripted, typically:

Cc1.AddRun(CSRBulkRun());

Cc1.run(1).generalOptions.checkBuckling = true;

Cc1.run(1).generalOptions.checkYield = true;

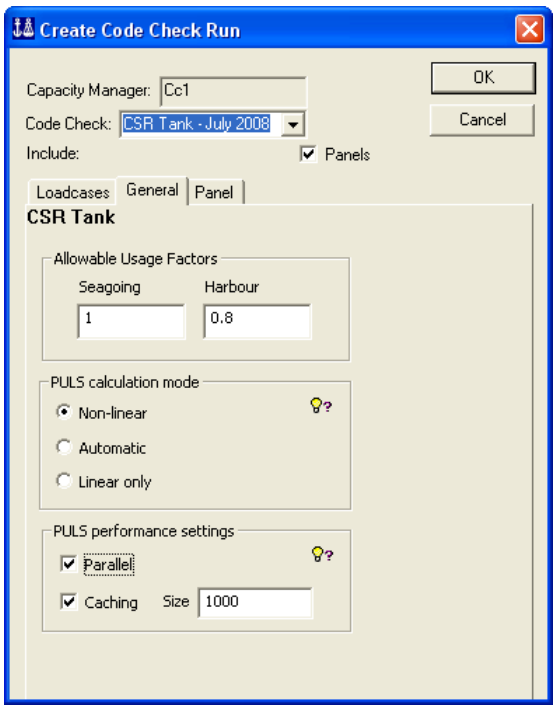
Cc1.run(1).generalOptions.safetyFactorBuckling = 1;

Cc1.run(1).generalOptions.safetyFactorYield = 1;

Cc1.run(1).generalOptions.transverseStressOption = tsGeneralBending;

Cc1.run(1).generalOptions.poissonCorrectionOption = psElementStress;

3.2.4.2.7 CSR Tank



Purpose: To define global and general code checking factors according to CSR Tank.

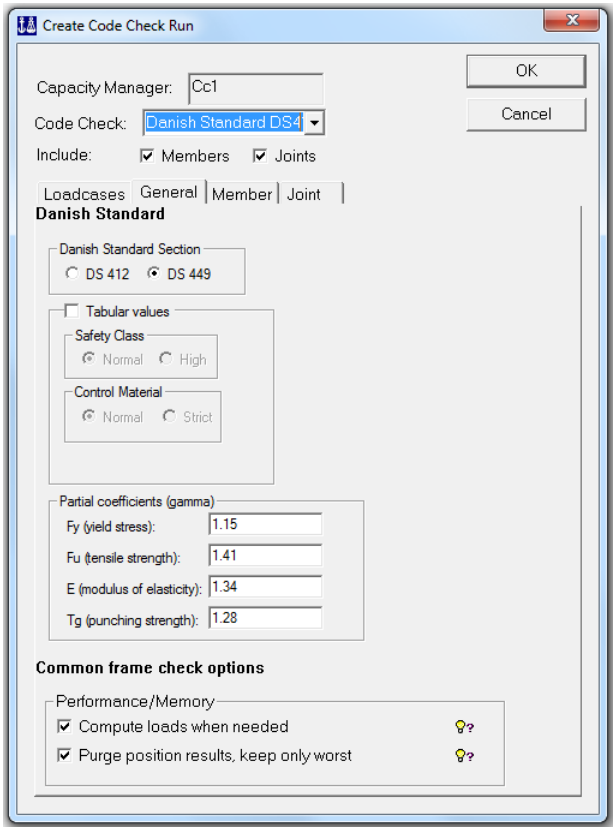
Check “Parallel” to utilize all available processors when running PULS. This makes the codecheck run faster, but as it takes all available processor capacity it will not be possible to do other work on your computer while PULS is running.

This command is scripted, typically:

Cc1.AddRun(CSRTankRun());

Cc1.run(1).generalOptions.parallelMode = true;

3.2.4.2.8 Danish Standard DS 412 / DS 449



Purpose: To define global and general code checking factors according to Danish Standard DS 412 and DS 449.

This command is scripted, typically:

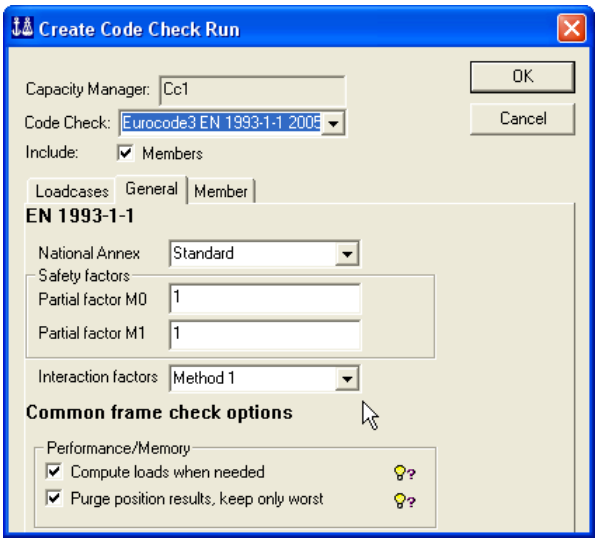
Cc1.AddRun(DSRun());

Cc1.run(1).generalOptions.isDS449CodeCheck = true;

Cc1.run(1).generalOptions.gammaFy = 1.15;

Cc1.run(1).generalOptions.gammaFu = 1.41;

3.2.4.2.9 Eurocode3 EN 1993-1-1 2005



Purpose: To define global and general code checking factors according to Eurocode3 EN 1993-1-1 2005. There are different national annexes; they have different safety factors.

This command is scripted, typically for annex "Standard":

Cc1.AddRun(EN199311Run());

Cc1.run(1).generalOptions.nationalAnnex = naStandard;

Cc1.run(1).generalOptions.partialFactorM0 = 1;

Cc1.run(1).generalOptions.partialFactorM1 = 1;

Cc1.run(1).generalOptions.method1 = true;

This command is scripted, typically for annex "Danish Normal":

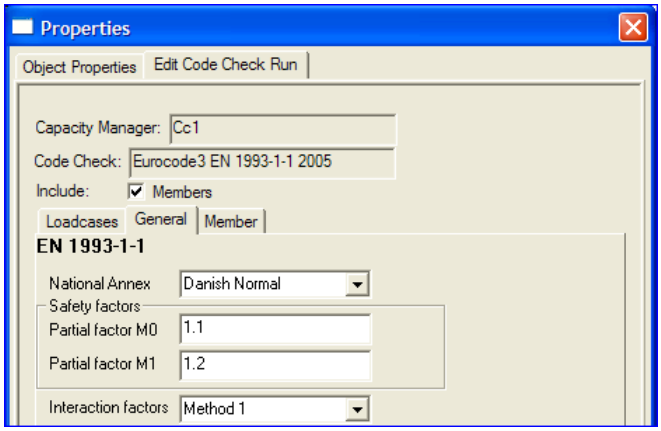
Cc1.AddRun(EN199311Run());

Cc1.run(1).generalOptions.nationalAnnex = DanishNormal;

Cc1.run(1).generalOptions.partialFactorM0 = 1.1;

Cc1.run(1).generalOptions.partialFactorM1 = 1.2;

Cc1.run(1).generalOptions.method1 = true;



This command is scripted, typically for annex "Danish Stricter":

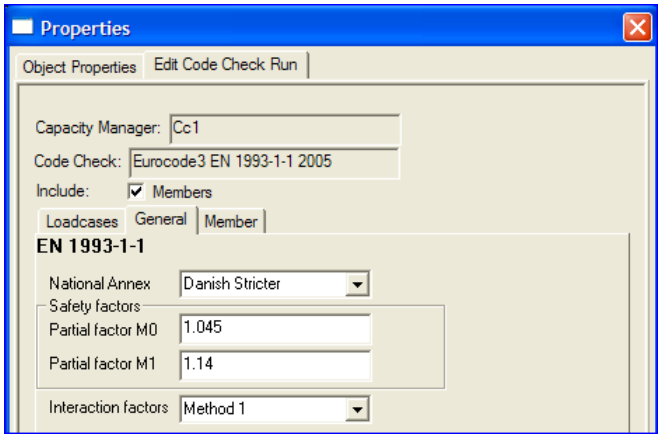
Cc1.AddRun(EN199311Run());

Cc1.run(1).generalOptions.nationalAnnex = naDanishStricter;

Cc1.run(1).generalOptions.partialFactorM0 = 1.045;

Cc1.run(1).generalOptions.partialFactorM1 = 1.14;

Cc1.run(1).generalOptions.method1 = true;



This command is scripted, typically for annex "Norwegian grouse":

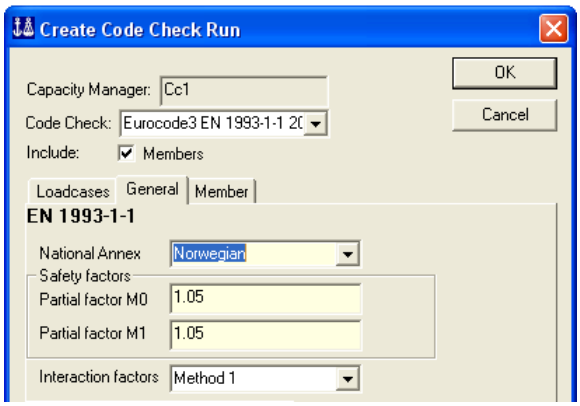
Cc1.AddRun(EN199311Run());

Cc1.run(1).generalOptions.nationalAnnex = naNorwegian;

Cc1.run(1).generalOptions.partialFactorM0 = 1.05;

Cc1.run(1).generalOptions.partialFactorM1 = 1.05;

Cc1.run(1).generalOptions.method1 = true;



3.2.4.2.10 ISO 19902 2007

Purpose: To define global and general code checking factors according to ISO 19902 2007. The factors prevail for members and joints.

This command is scripted, typically for members:

Cc1.AddRun(ISO19902Run());

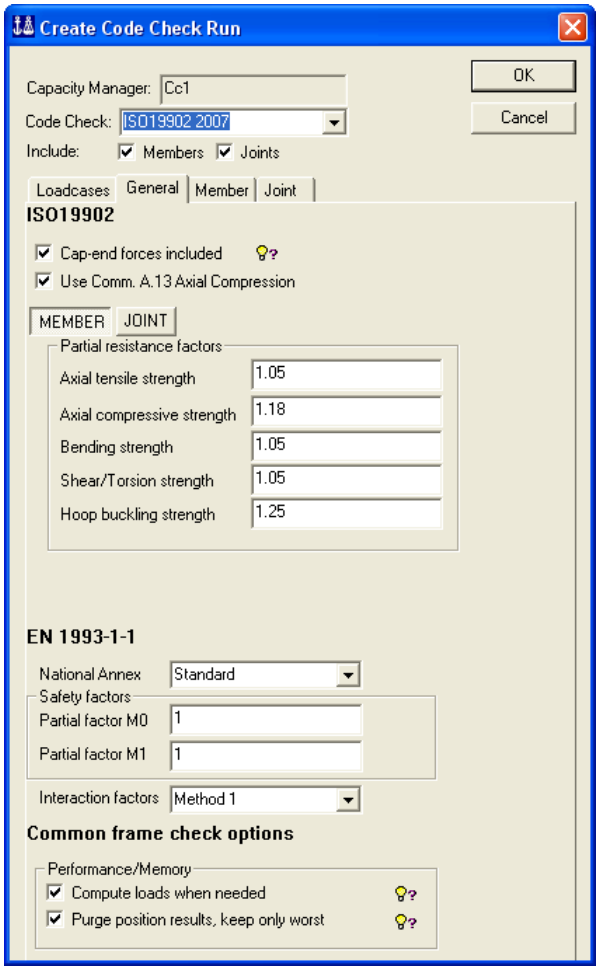
Cc1.run(1).generalOptions.RFPipeXPress = 1.25;

Cc1.run(1).generalOptions.RFPipeShear = 1.05;

Cc1.run(1).generalOptions.RFPipeBend = 1.05;

Cc1.run(1).generalOptions.RFPipeTens = 1.05;

Cc1.run(1).generalOptions.RFPipeComp = 1.15;



This command is scripted, typically for joints:

Cc1.run(1).generalOptions.C1_Y_ax = 25;

Cc1.run(1).generalOptions.C1_X_ax = 20;

Cc1.run(1).generalOptions.C1_K_ax = 14;

Cc1.run(1).generalOptions.C1_mom = 25;

Cc1.run(1).generalOptions.C2_Y_ax = 11;

Cc1.run(1).generalOptions.C2_X_ax = 22;

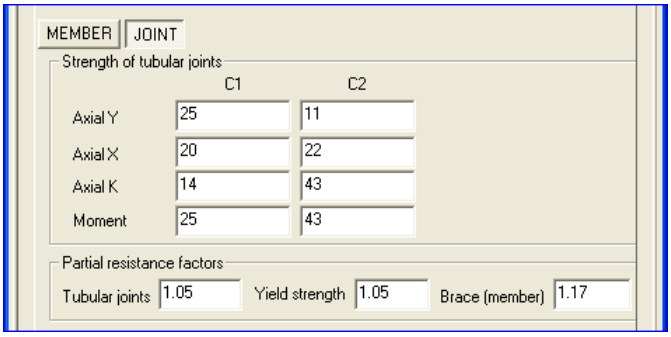
Cc1.run(1).generalOptions.C2_K_ax = 43;

Cc1.run(1).generalOptions.C2_mom = 43;

Cc1.run(1).generalOptions.RFJointYield = 1.05;

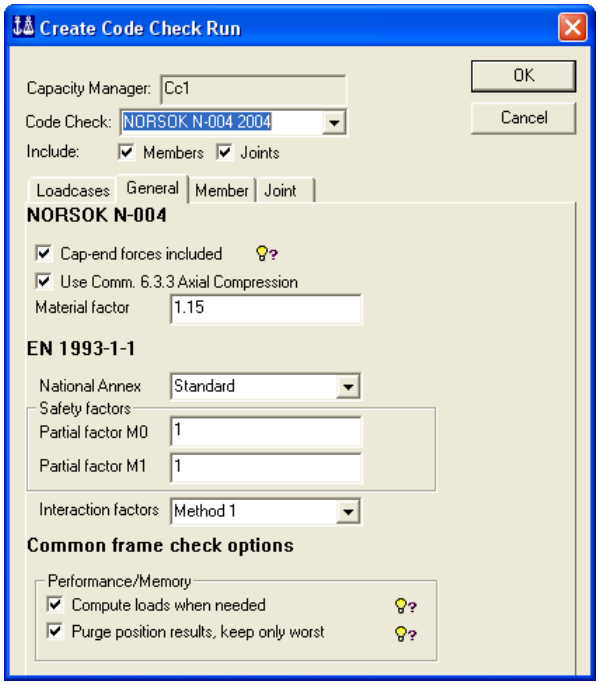
Cc1.run(1).generalOptions.RFJoint = 1.05;

Cc1.run(1).generalOptions.RFJointZj = 1.17;



3.2.4.2.11 NORSOK N-004 2004

Purpose: To define global and general code checking factors according to NORSOK N-004 2004.



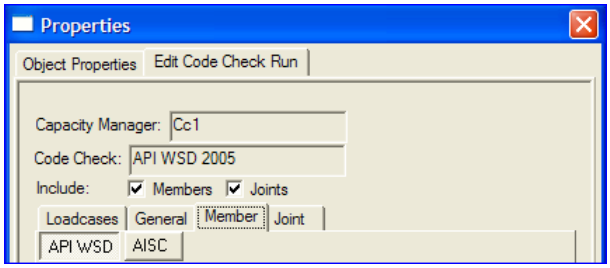
This command is scripted, typically:

Cc1.AddRun(NorsokRun());

Cc1.run(10).generalOptions.materialFactor = 1.15;

3.2.4.3 Member and panel

Purpose: To define global and member or panel specific code checking factors. The factors are different for the various code checking standards. They are listed in the following.

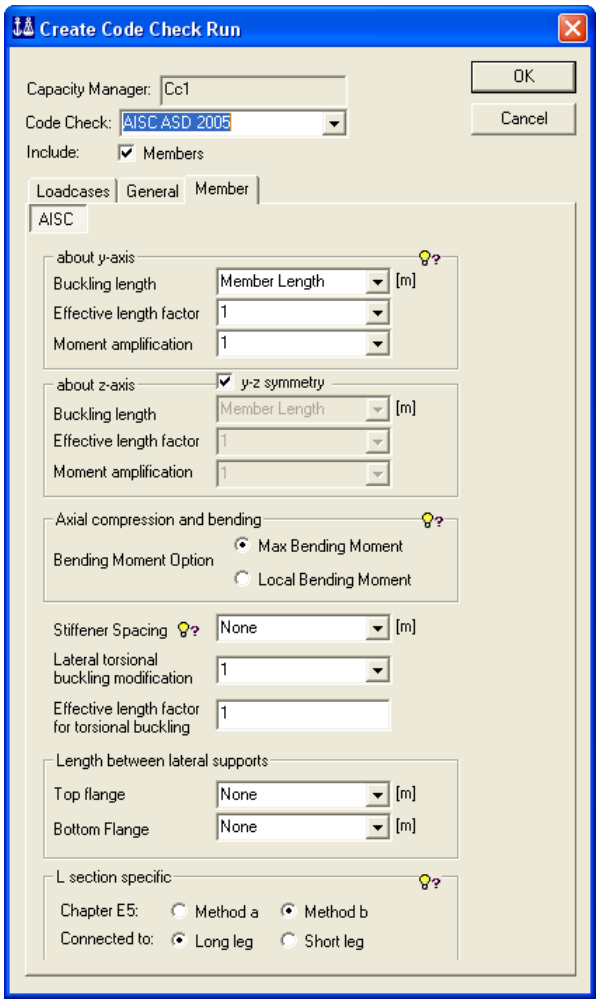


3.2.4.3.1 AISC ASD 2005

Purpose: To define global and member specific code checking factors according to AISC ASD 2005.

This command is scripted, typically when using lateral torsional buckling modification 2:

Cc1.run(1).memberOptions.
lTorsionalBucklingFactor = 2;



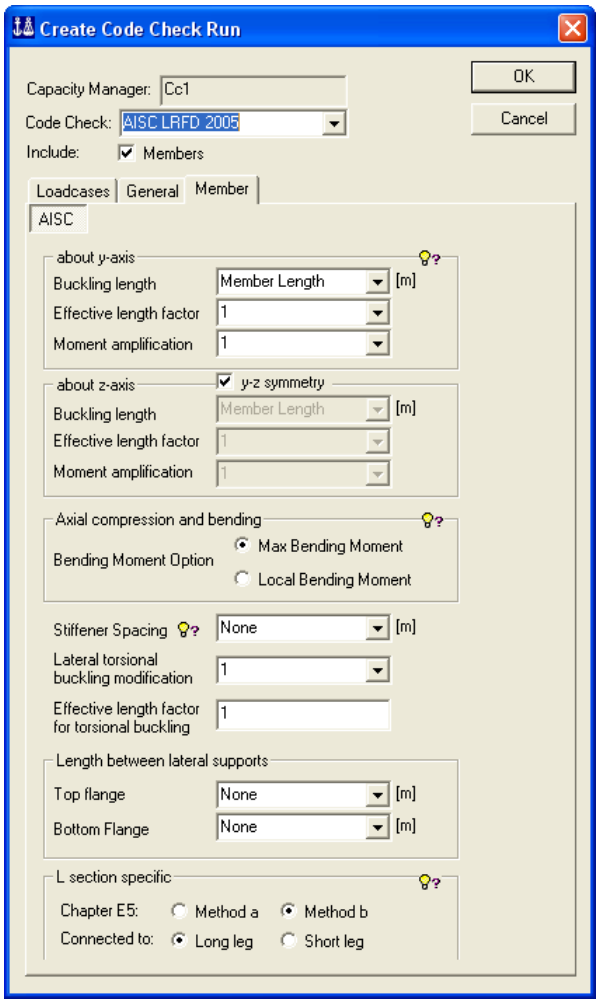
3.2.4.3.2 AISC LRFD 2005

Purpose: To define global and member specific code checking factors according to AISC LRFD 2005.

This command is scripted, typically when using length between lateral supports equal to 1m and 2m for top and bottom flange respectively:

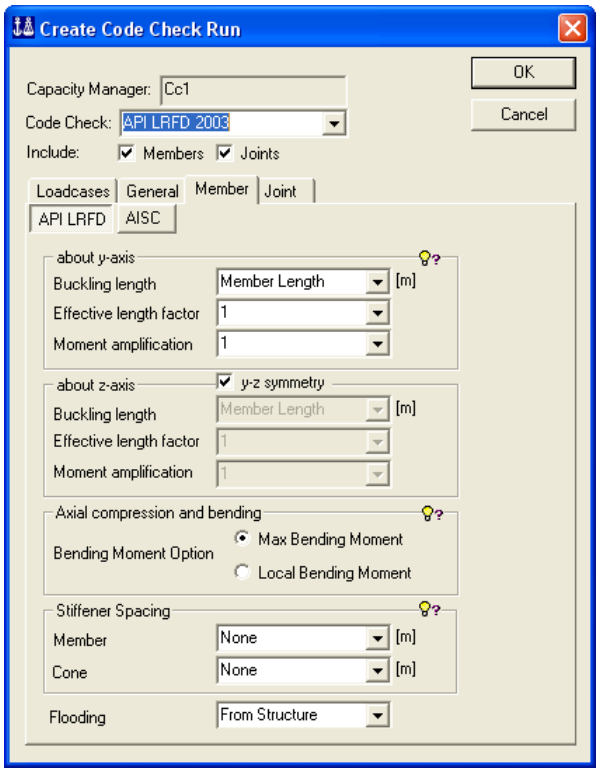
Cc1.run(1).memberOptions.
topFlangeLSupportSpacing = 1;

Cc1.run(1).memberOptions.
bottomFlangeLSupportSpacing = 2;



3.2.4.3.3 API LRFD 2003

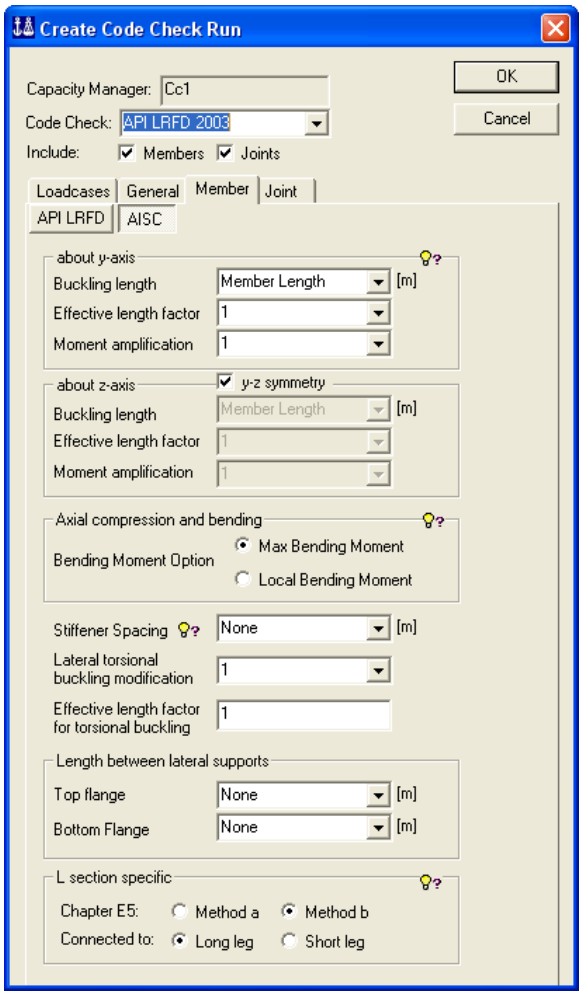
Purpose: To define global and member specific code checking factors according to API LRFD 2003. The factors prevail for tubular and non-tubular members.



This command is scripted, typically when using a moment amplification of 1.2:

Cc1.run(1).memberOptions.momentAmplificationY = 1.2;

Cc1.run(1).memberOptions.momentAmplificationZ = 1.2;



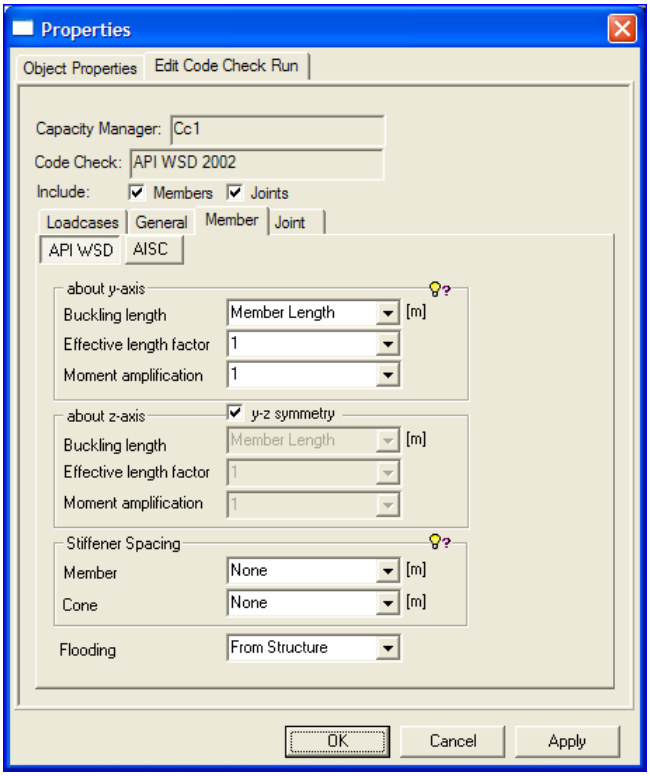
This command is scripted, typically when using "Chapter E5 Method a" connected to "Short leg" for L profiles:

Cc1.run(1).memberOptions.aisc.E5Method = moE5a;

Cc1.run(1).memberOptions.aisc .connectionToLongLeg = false;

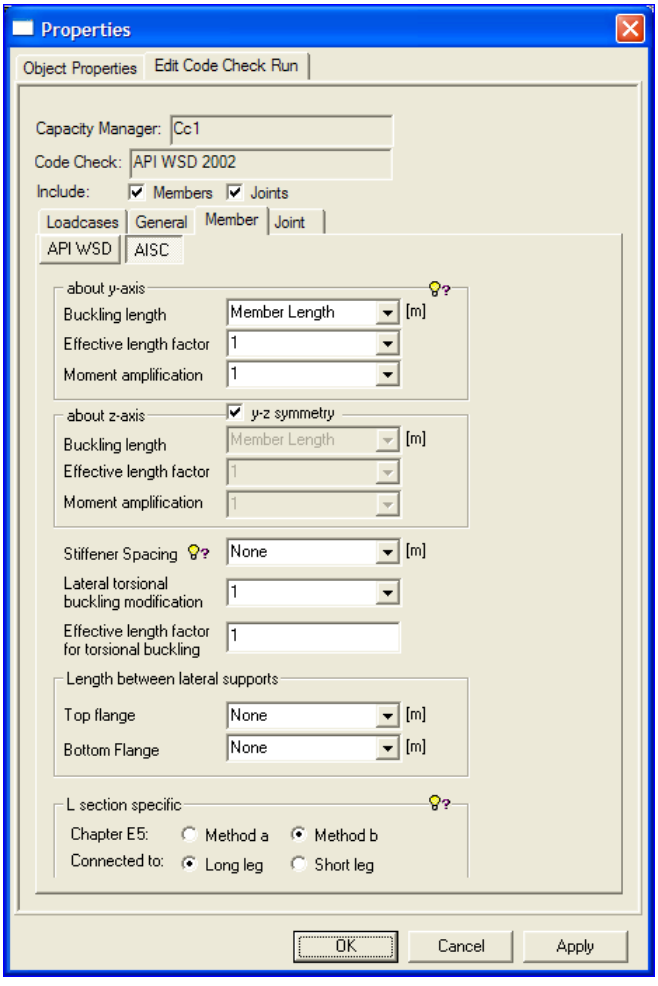
3.2.4.3.4 API WSD 2002

Purpose: To define global and member specific code checking factors according to API WSD 2002. The factors prevail for tubular and non-tubular members.



This command is scripted, typically when specifying flooding to "Not flooded":

Cc1.run(1).memberOptions.flooding = cfNotFlooded;



This command is scripted, typically when specifying the buckling length to 5 m:

Cc1.run(1).memberOptions.aisc.bucklingY = BucklingLength(5 m, 1);

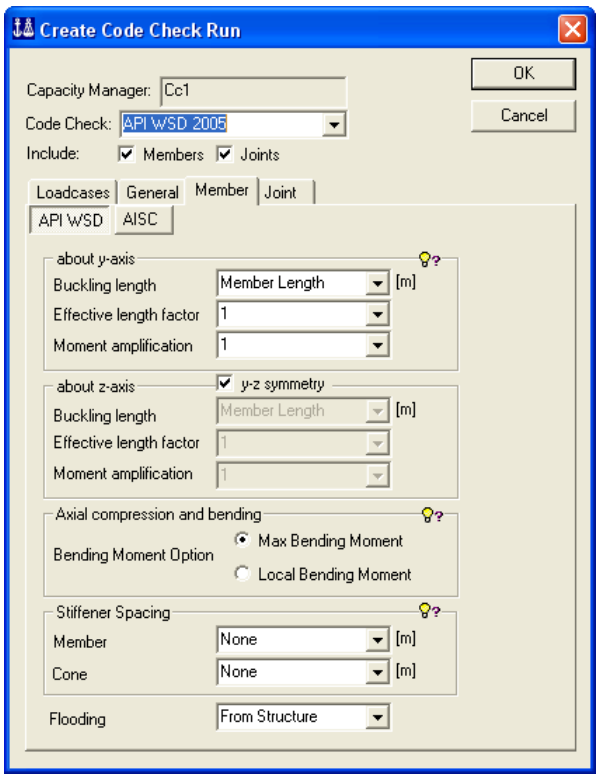
Cc1.run15).memberOptions.aisc.bucklingZ = BucklingLength(5 m, 1);

3.2.4.3.5 API WSD 2005

Purpose: To define global and member specific code checking factors according to API WSD 2005. The factors prevail for tubular and non-tubular members.

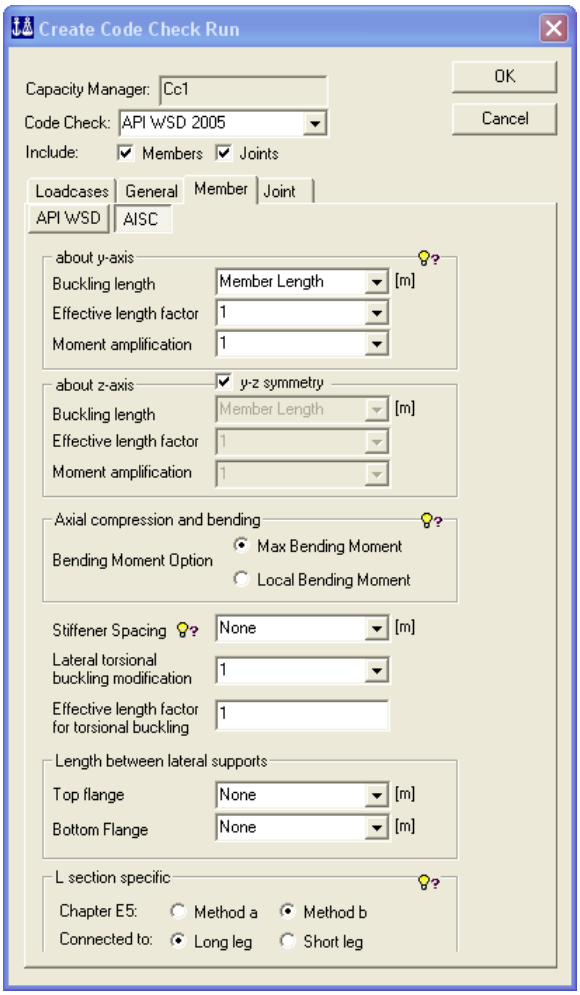
This command is scripted, typically when assigning a different buckling length factor (0.5) about the z-axis:

Cc1.run(1).memberOptions.bucklingZ = BucklingLength(moMemberLength, 0.5);



This command is scripted, typically when assigning a different buckling length 600 c m about the z-axis:

Cc1.run(1).memberOptions.aisc.bucklingZ = BucklingLength(600 cm, 1);



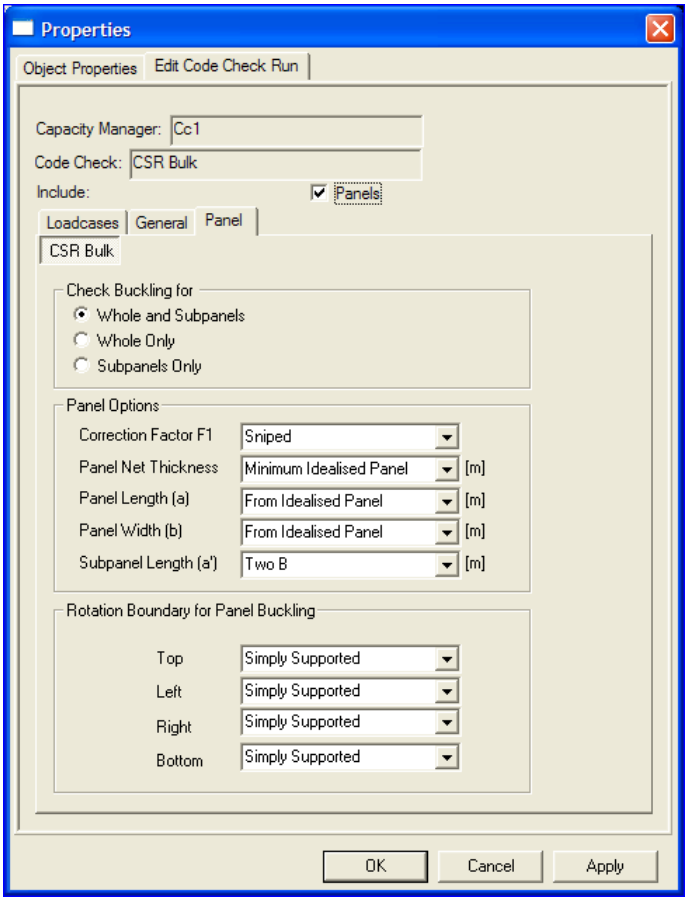
3.2.4.3.6 CSR Bulk

Purpose: To define global and panel specific code checking factors according to CSR Bulk.

This command is scripted, typically when modifying the panel net thickness to "Average Idealised Panel" and the top rotation boundary for panel buckling to "Clamped":

Cc1.run(1).panelOptions.panelThickness = ptAverageIdealisedPanel;

Cc1.run(1).panelOptions.rotationBoundaryTop = rbClamped;



3.2.4.3.7 CSR Tank

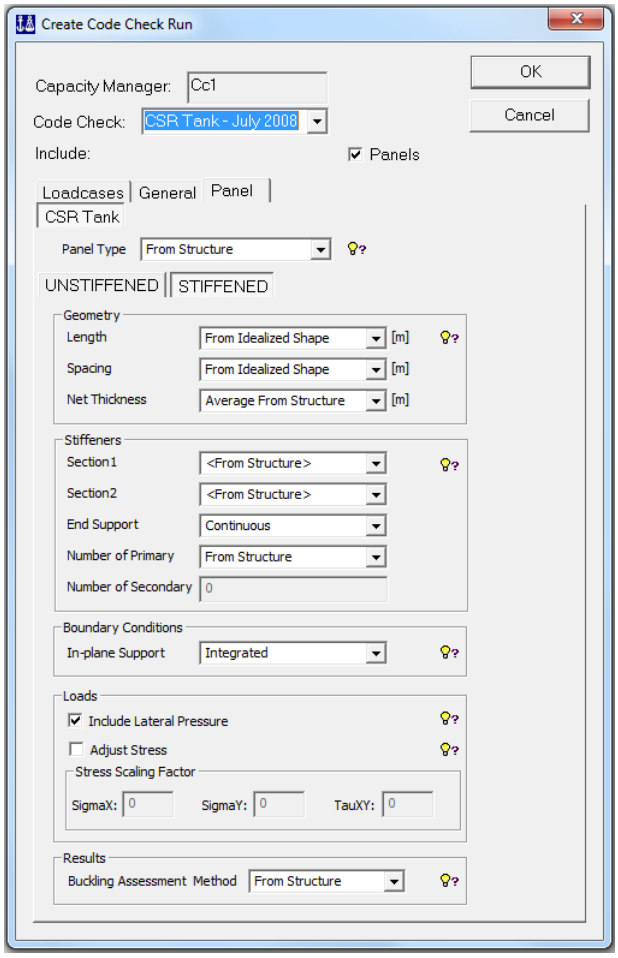
Purpose: To define global and panel specific code checking factors according to CSR Tank.

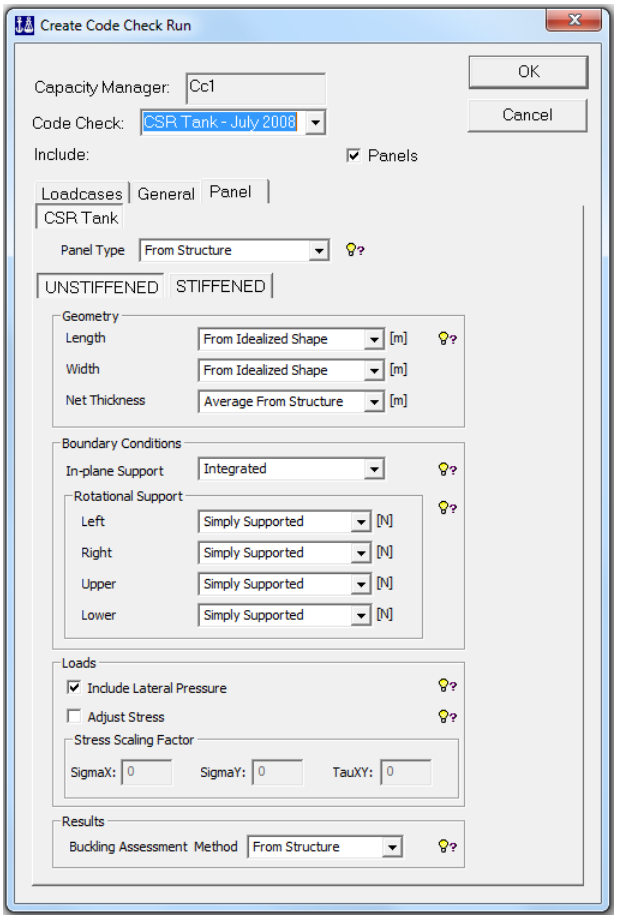
Note that CSR Tank categorizes the panels into two groups, stiffened or unstiffened. You can also force panels to be treated as stiffened or unstiffened.

This command is scripted, typically:

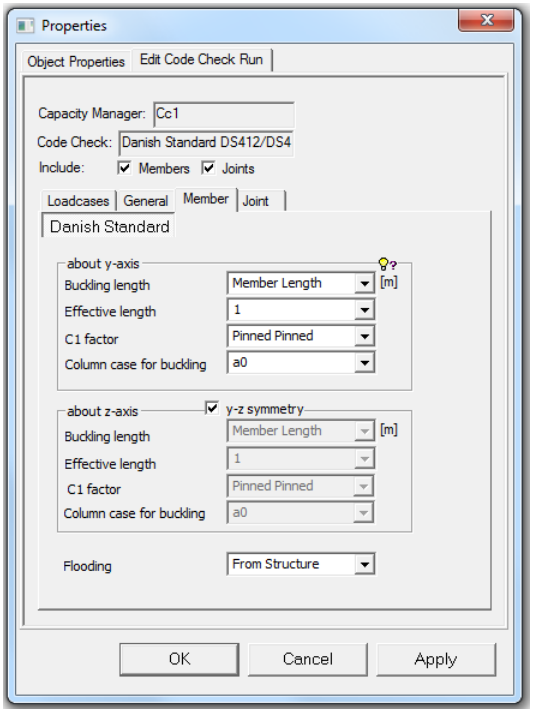
Cc1.AddRun(CSRTankRun());

Cc1.run(1).panelOptions.unstiffened.idealizedLength = 1;





3.2.4.3.8 Danish Standard DS 412 / DS 449



Purpose: To define global and member specific code checking factors according to Danish Standard DS 412 and DS 449

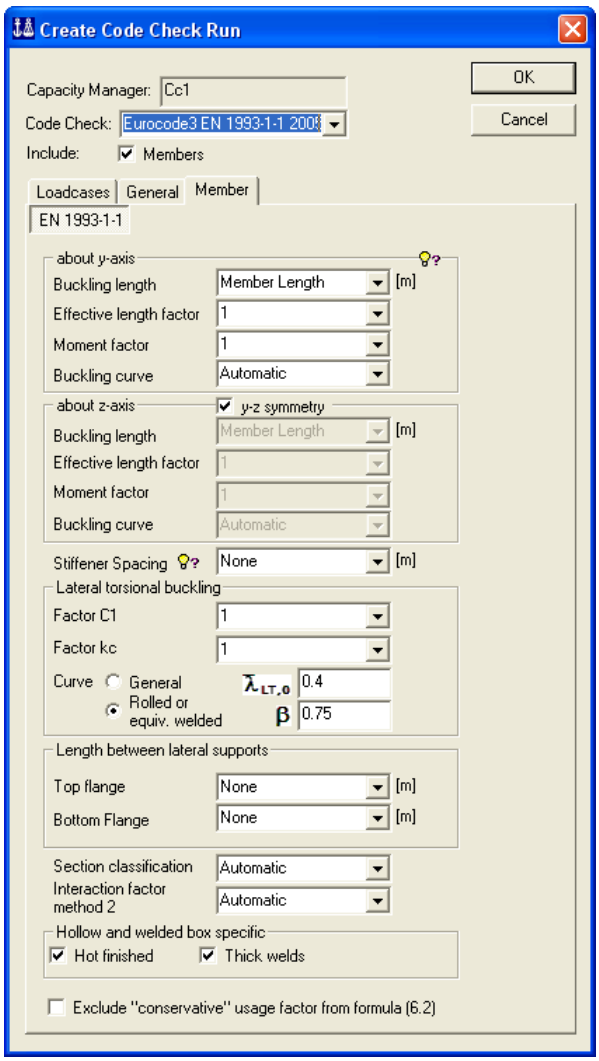
This command is scripted, typicallywhen modifying the buckling length:

Cc1.AddRun(DSRun());

Cc1.run(2).memberOptions.bucklingY = BucklingLength(1, 1);

Cc1.run(2).memberOptions.bucklingZ = BucklingLength(1, 1);

3.2.4.3.9 Eurocode3 EN 1993-1-1 2005



Purpose: To define global and member specific code checking factors according to Eurocode3 EN 1993-1-1 2005.

This command is scripted, typically when modifying the lateral torsional buckling Factor C1 to "3":

Cc1.run(1).memberOptions.LTBFactorC1 = 3;

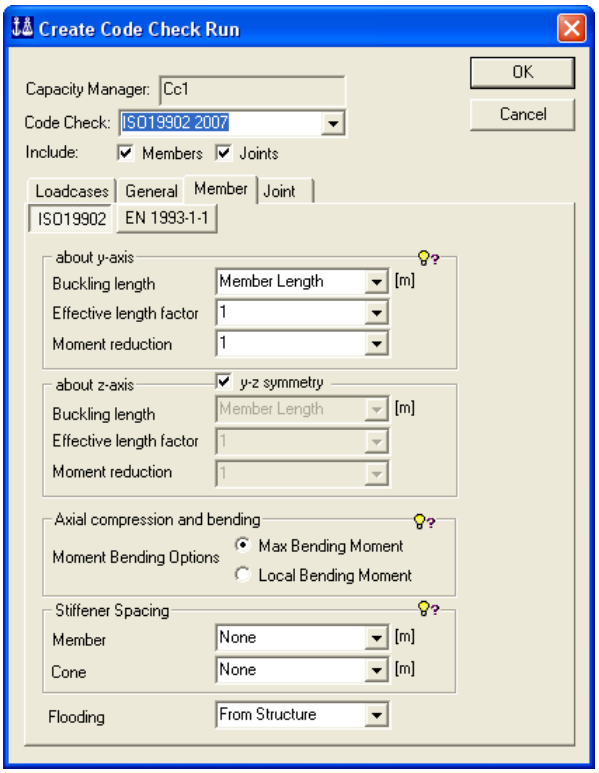
3.2.4.3.10 ISO 19902 2007

Purpose: To define global and member specific code checking factors according to ISO 19902 2007. The factors prevail for tubular and non-tubular members.

This command is scripted, typically when modifying moment reduction factors about y and z-axis to "Case1" and "Case3":

Cc1.run(1).memberOptions.momentReductionY = moCase1;

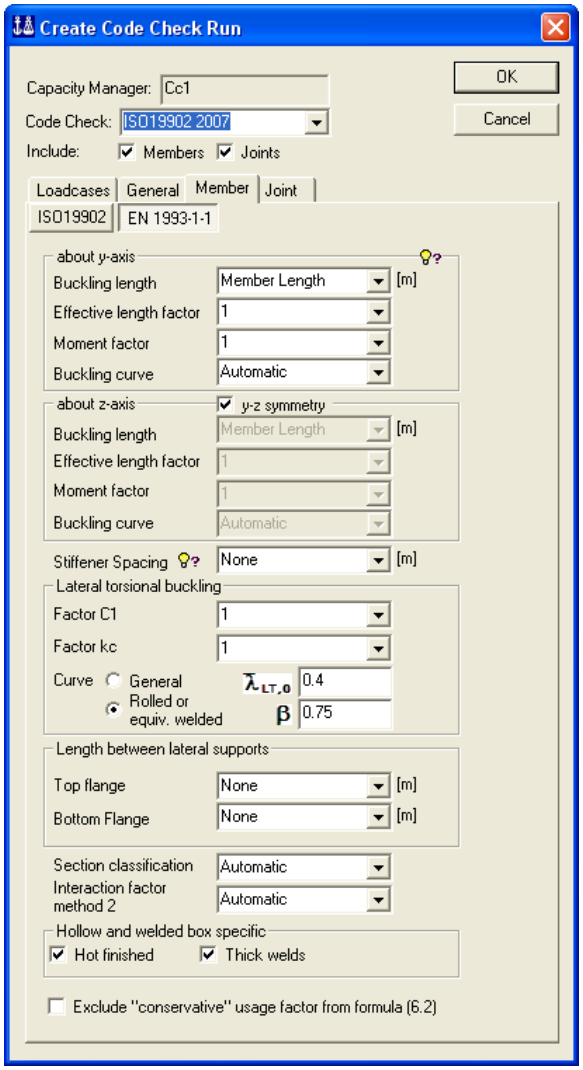
Cc1.run(1).memberOptions.momentReductionZ = moCase3;



This command is scripted, typically when disabling the "Hot finished" and "Thick welds" for hollow and welded box profiles:

Cc1.run(9).memberOptions. EN1993_1_1.hotFinish = false;

Cc1.run(9).memberOptions. EN1993_1_1.thickWelds = false;

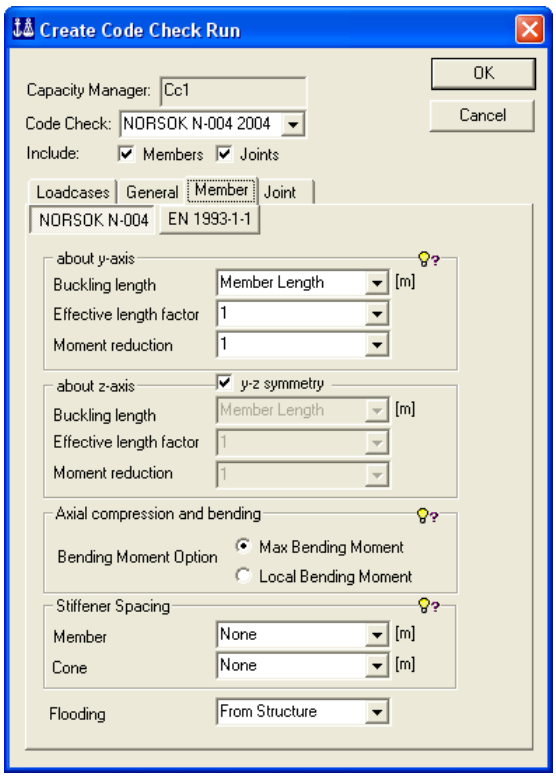


3.2.4.3.11 NORSOK N-004 2004

Purpose: To define global and member specific code checking factors according to NORSOK N-004 2004. The factors prevail for tubular and non-tubular members.

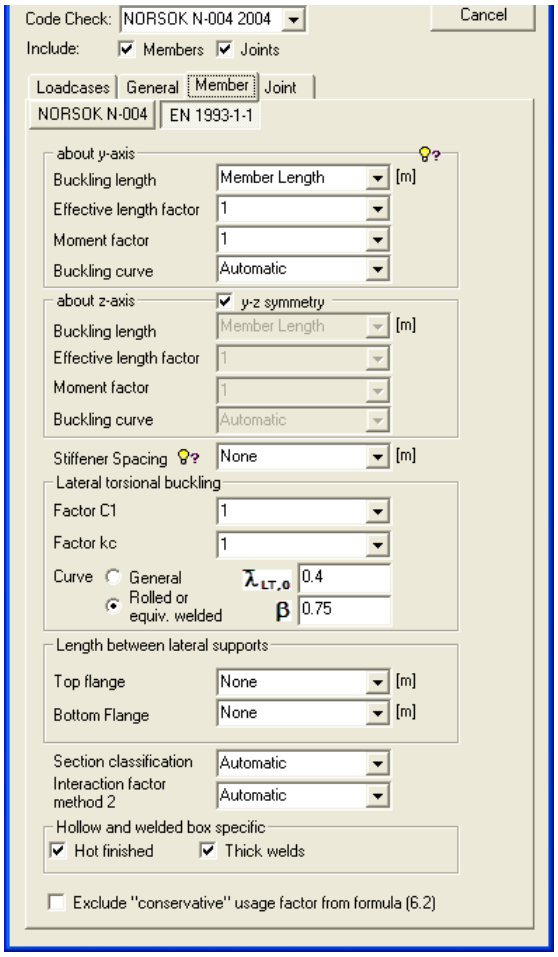
This command is scripted, typically when specifying flooding to status "flooded".

Cc1.run(1).memberOptions.flooding = cfFlooded;



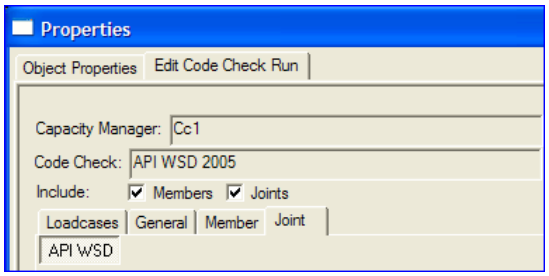
This command is scripted, typically when specifying the factor kc for the lateral torsionl buckling to "1.2":

Cc1.run(1).memberOptions.
EN1993_1_1.LTBFactorKc = 1.2;



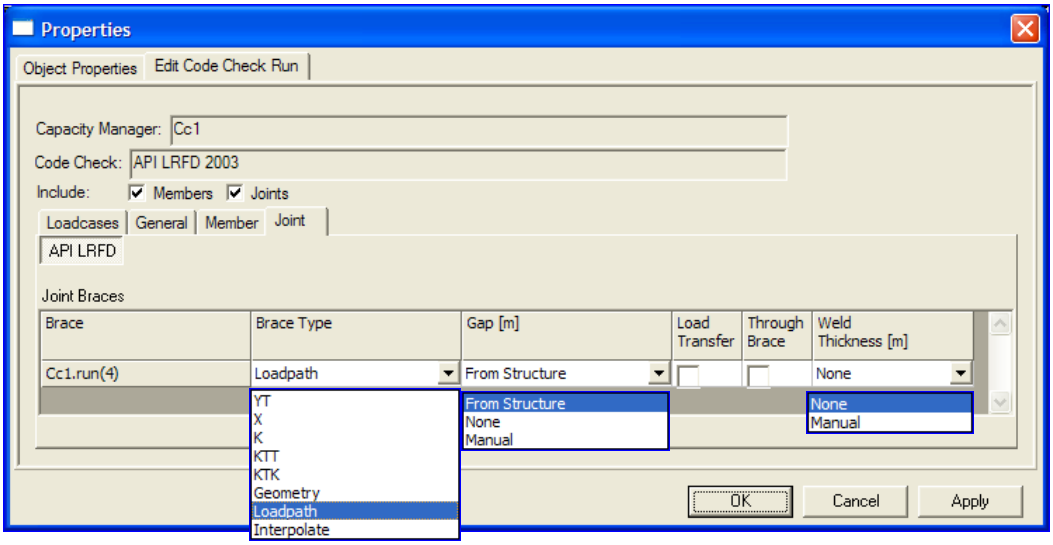
3.2.4.4 Joint

Purpose: To define global and tubular joint specific code checking factors. The factors are different for the various code checking standards. They are listed in the following.



3.2.4.4.1 API LRFD 2003

Purpose: To define global and tubular joint specific code checking factors according to API LRFD 2003.



This command is scripted, typically when modifying brace type to "KTT", a gap value of 2in and a weld thickness of 15mm:

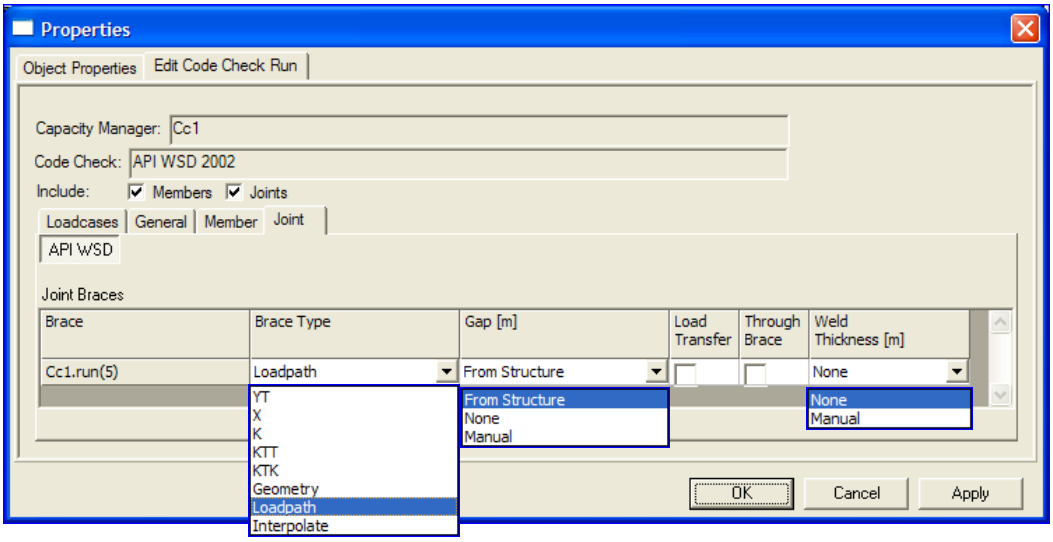
Cc1.run(4).braceOptions.braceType = braceTypeKTT;

Cc1.run(4).braceOptions.gap = 2 in;

Cc1.run(4).braceOptions.weldThickness = 15 mm;

3.2.4.4.2 API WSD 2002

Purpose: To define global and tubular joint specific code checking factors according to API WSD 2002.



This command is scripted, typically when modifying brace type to "X", Load Transfer active and Through Brace active:

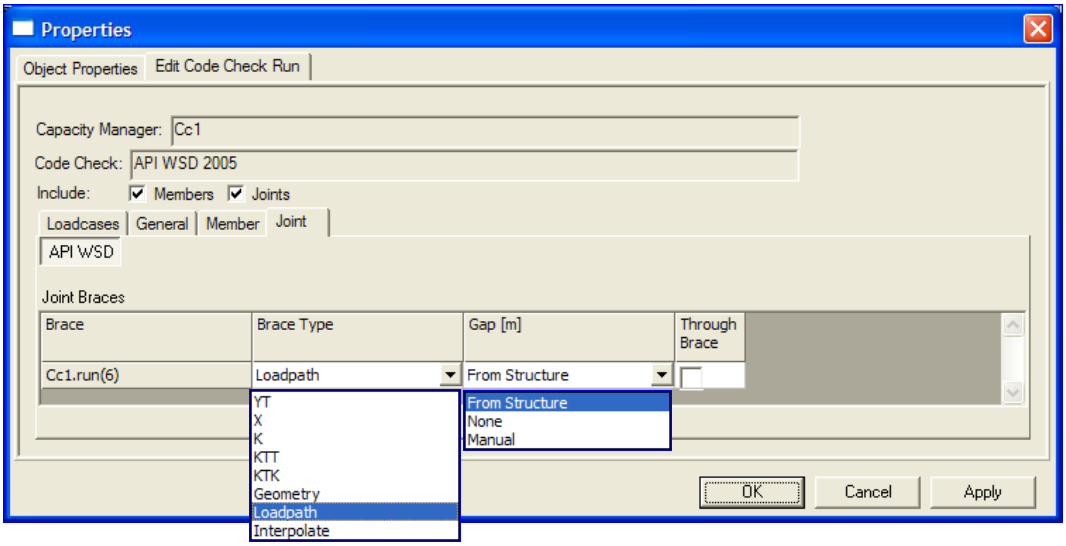
Cc1.run(5).braceOptions.braceType = braceTypeX;

Cc1.run(5).braceOptions.loadTransfer = true;

Cc1.run(5).braceOptions.throughBrace = true;

3.2.4.4.3 API WSD 2005

Purpose: To define global and tubular joint specific code checking factors according to API WSD 2005.

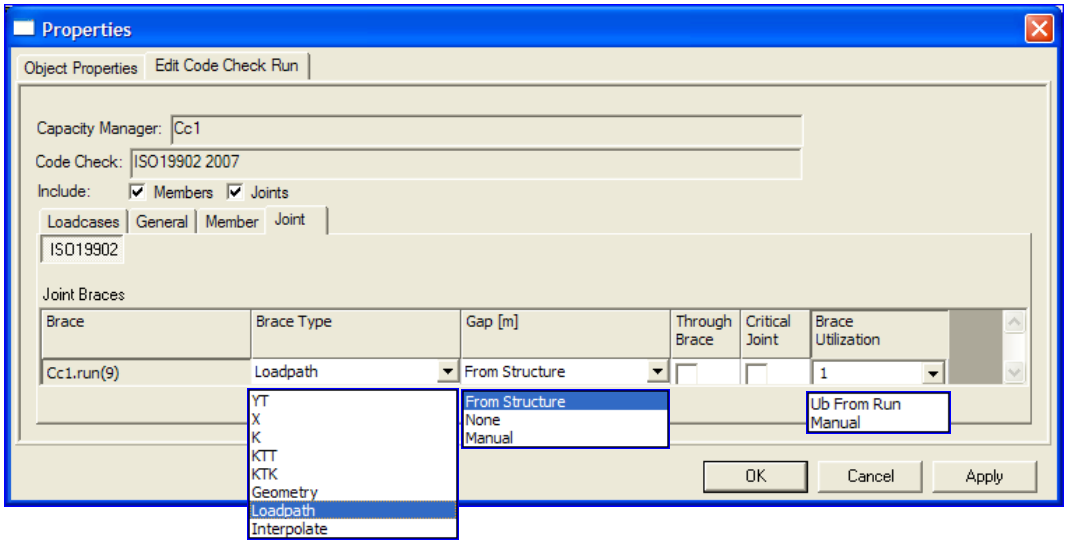


This command is scripted, typically when modifying brace type to Interpolate: 20% YT, 30% X, 30% K, 10% KTT and 10% KTK:

Cc1.run(6).braceOptions.braceType = braceTypeInterpolate(0.2, 0.3, 0.3, 0.1, 0.1);

3.2.4.4.4 ISO 19902 2007

Purpose: To define global and tubular joint specific code checking factors according to ISO 19902 2007.

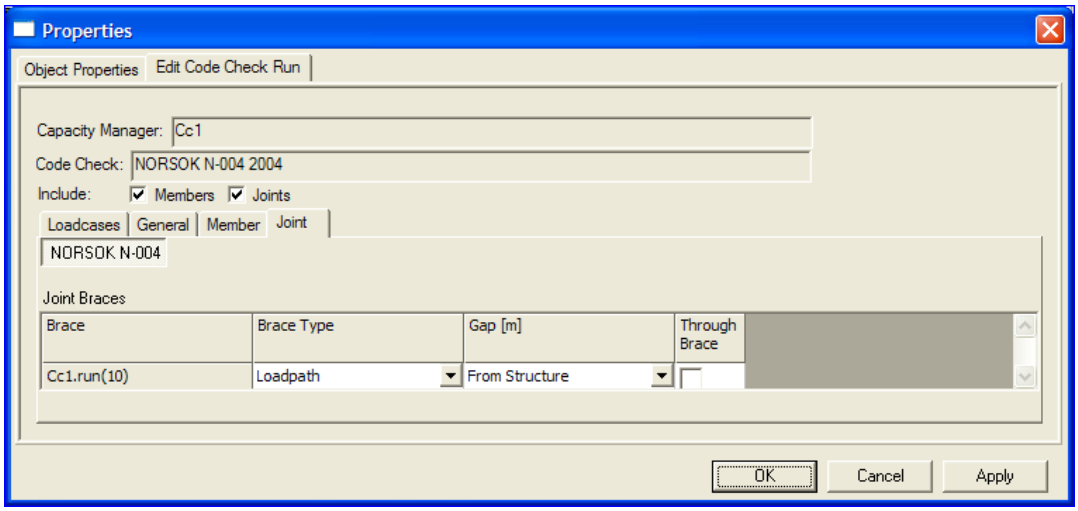


This command is scripted, typically when modifying the brace utilisation to 90% (or 0.90):

Cc1.run(9).braceOptions.braceUtilization = 0.9;

3.2.4.4.5 NORSOK N-004 2004

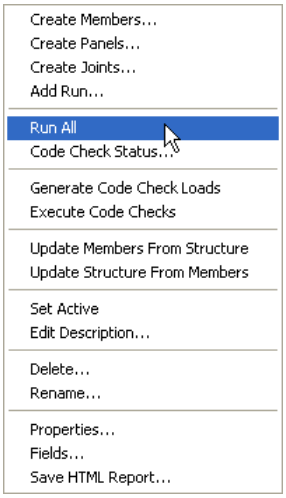
Purpose: To define global and tubular joint specific code checking factors according to NORSOK N-004 2004.



This command is scripted, typically when neglecting any gap values part of the structural model:

Cc1.run(10).braceOptions.gap = gtNone;

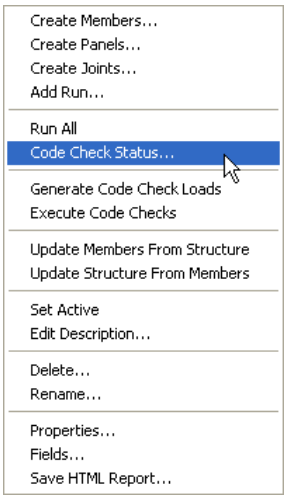
3.2.5 Run All



This option is intended to be used when doing redesign.

It re-runs the analysis, generates the code check loads and executes the code check in one single operation.

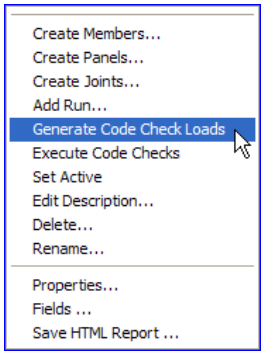
3.2.6 Code Check Status



Checks if the structural connection is ok.

This command is not scripted.

3.2.7 Generate Code Check Loads

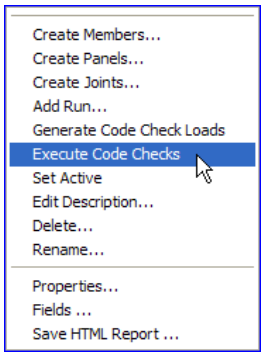


Purpose: To compute forces and moments at system defined locations for use in the subsequent code check. See User Manual Volume 4 and 5 for details.

This command is scripted, typically:

Cc1.run(1).generateCodeCheckLoads();

3.2.8 Execute Code Checks



Purpose: To perform the code check according to the parameters as defined in the Add Run and the code check forces and moments computed by the generate Code Check Loads.

This command is scripted, typically:

Cc1.executeCodeChecks();

3.2.9 Update Members from Structure

This is used when you change something in the concept model, like removing a beam or altering segmentation, and you want this change to be reflected in the capacity model. Information is moved ***from*** the concept model ***to*** the capacity model.

This command is scripted, typically:

Cc1.updateMembersFromStructure();

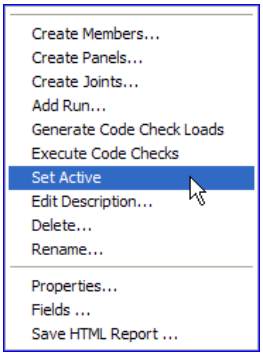
3.2.10 Update Structure from Member

This is used to move information from the capacity model to the concept model.

This command is scripted, typically:

Cc1.updateStructureFromMembers();

3.2.11 Set active



Purpose: To switch between various capacity managers and code checks and set one to active.

This command is scripted, typically when switching from capacity manager Cc2 to Cc1:

Cc1.setActive();

3.2.12 Edit Description

Purpose: To add user defined text for a description of a capacity manager or a code check run.

This command is scripted, typically when adding the text UM Example:

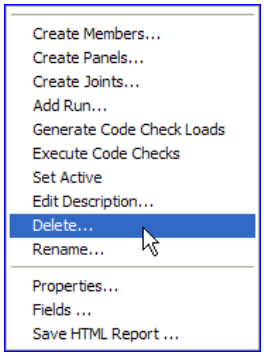
Cc1.description = "UM example";

3.2.13 Delete

Purpose: To delete a capacity manager or a code check run.

This command is scripted, typically:

Delete(Cc1);

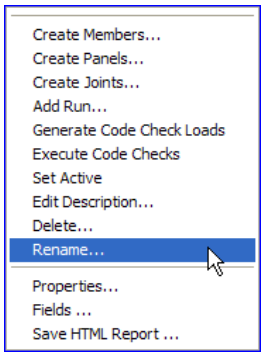


3.2.14 Rename

Purpose: To rename a capacity manager or a code check run.

This command is scripted, typically when renaming to UM_Cap_Manager:

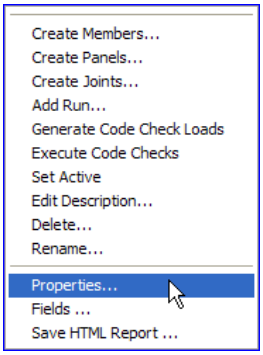
Rename(Cc1,"UM_Cap_Manager");



3.2.15 Properties

Purpose: To modify a capacity manager or a code check run.

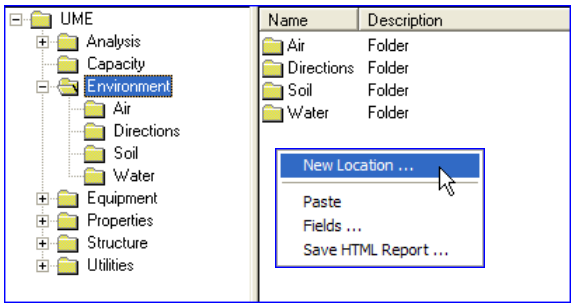
This command is scripted. The scripting depends on which modifications are performed to the capacity manager or a code check run.



3.3 Environment

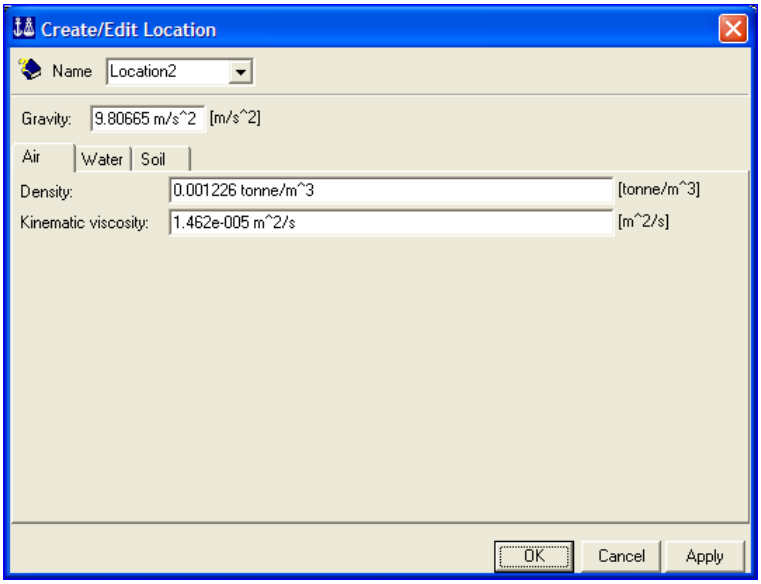
Purpose: To define the environmental data for air (wind), water (wave and current) and soil for use in hydrodynamic and pile-soil analyses. There may be several locations whereby it is possible to run a number of analyses using the different locations.

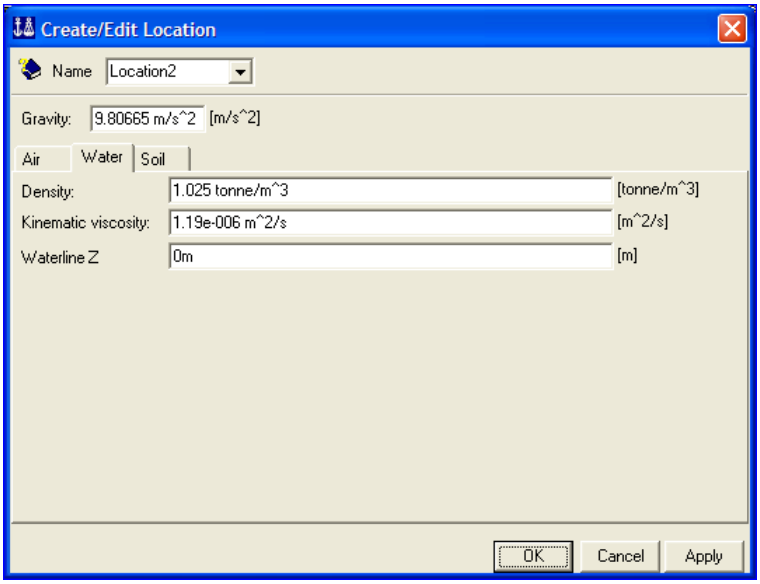
The locations are defined from the Environment Folder; RMB and select New Location.



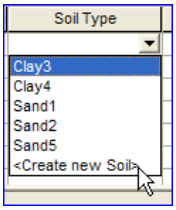
A location refers to data defined in the Environment sub-folders for Air, Directions, Soil and Water.

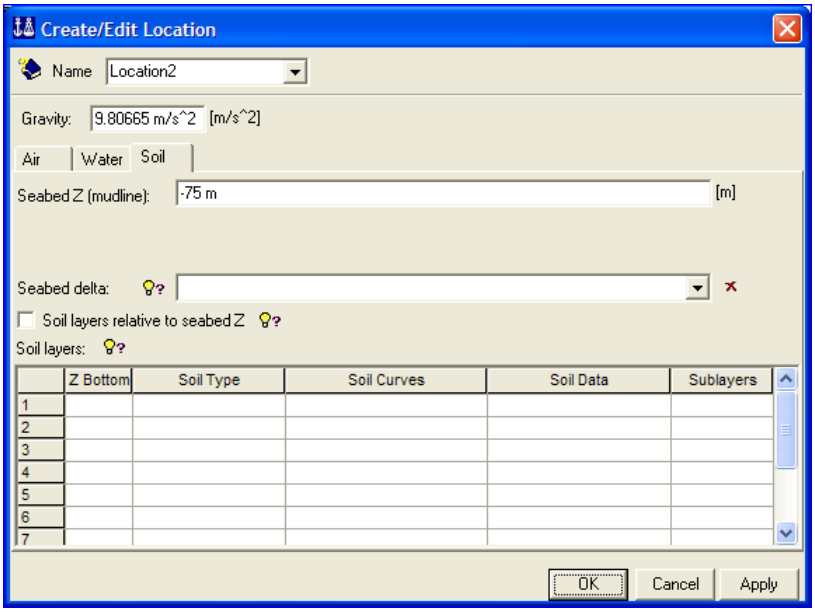
For more details on how to define the environmental data, see User Manual Volume 2.





When filling in a table as shown to the right, data can be created from this view. Alternatively, the data as defined in the subfolders can be used.





This command is scripted. The scripting depends on which data are used to define the location. Typically:

Location1 = Location(124 m, 0 m);

Location1.gravity = 9.80665 m/s^2;

Location1.air.density = 0.001226 tonne/m^3;

Location1.air.kinematicViscosity = 1.462e-005 m^2/s;

Location1.water.density = 1.025 tonne/m^3;

Location1.water.kinematicViscosity = 1.19e-006 m^2/s;

Location1.seabed.normaldirection = Vector3d(0 m,0 m,1 m);

Location1.seabed.seabedDelta = Scour1;

Location1.insertSoilBorder(-1.5 m);

..

Location1.insertSoilBorder(-80 m);

Location1.soil(1).soilCurves = SoilCurves1;

Location1.soil(1).soilData = SoilData1;

Location1.soil(1).soilType = Sand1;

Location1.soil(1).numberOfSublayers = 1;

...

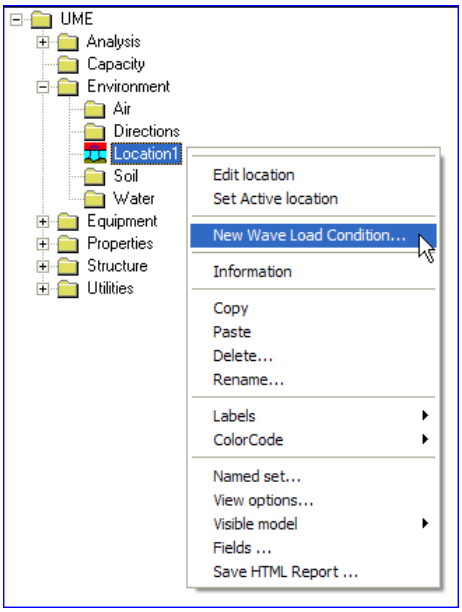
Location1.soil(7).soilCurves = SoilCurves1;

Location1.soil(7).soilData = SoilData6;

Location1.soil(7).soilType = Sand5;

Location1.soil(7).numberOfSublayers = 2;

When a location is defined, the associated wave load condition including current, wind and wave model is defined. The data can be entered in the view below or the data as defined in the subfolders can be used.



This command is scripted. The scripting depends on which data are used to define the condition. Typically:

Condition1 = DeterministicTime(Location1);

Condition1.waterSurface.regularWaveSet = WaveSet1;

Condition1.populate();

Condition1.addCalmSea();

Condition1.component(1).water.current(CurrentProfile1);

Condition1.component(1).waterSurface.waveModel(Airy());

Condition1.component(2).water.current(CurrentProfile1);

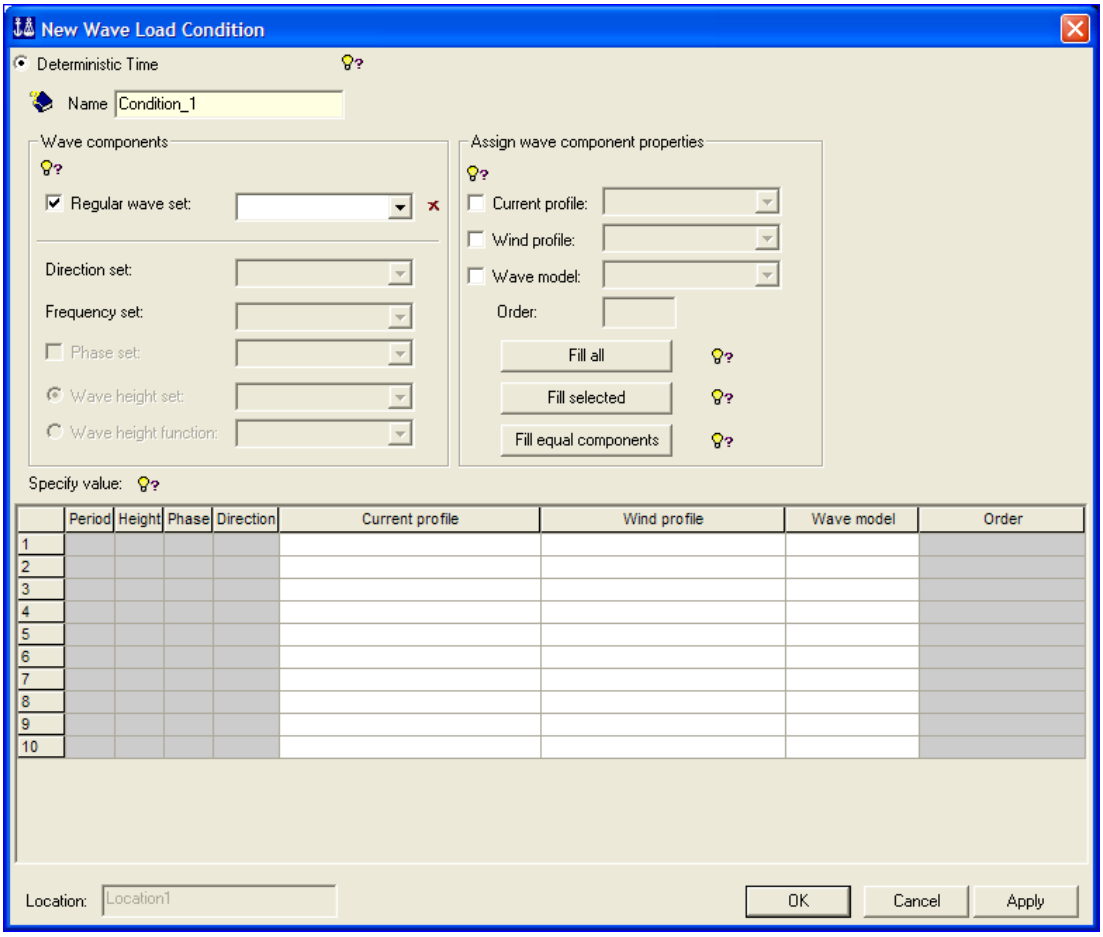
Condition1.component(2).waterSurface.waveModel(Stokes5());

Condition1.component(3).water.current(CurrentProfile1);

Condition1.component(3).waterSurface.waveModel(Stokes5());

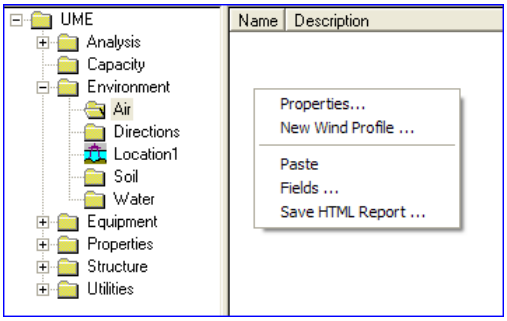
Condition1.component(4).water.current(CurrentProfile1);

Condition1.component(4).waterSurface.waveModel(Stokes5());



3.3.1 Air

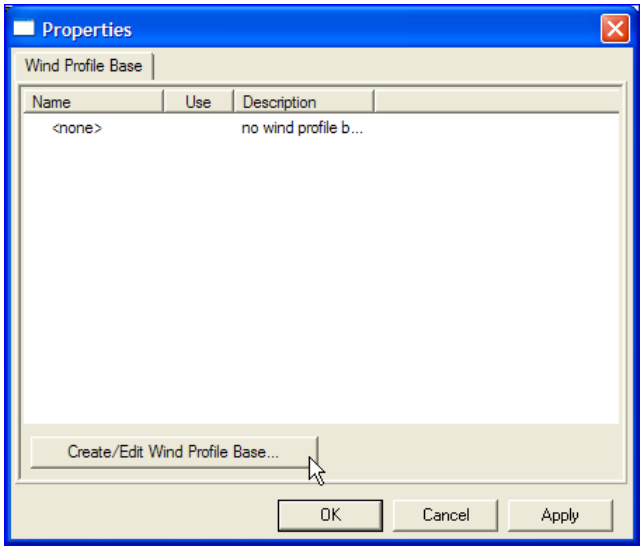
Purpose: To define and modify wind profile data. These data can be used later when defining a wave load condition(s).



3.3.1.1 Properties

Purpose: To define a new wind profile or modify existing wind profiles.

This command is scripted. The scripting depends on which parameters are modified or entered to create a new wind profile, see below.

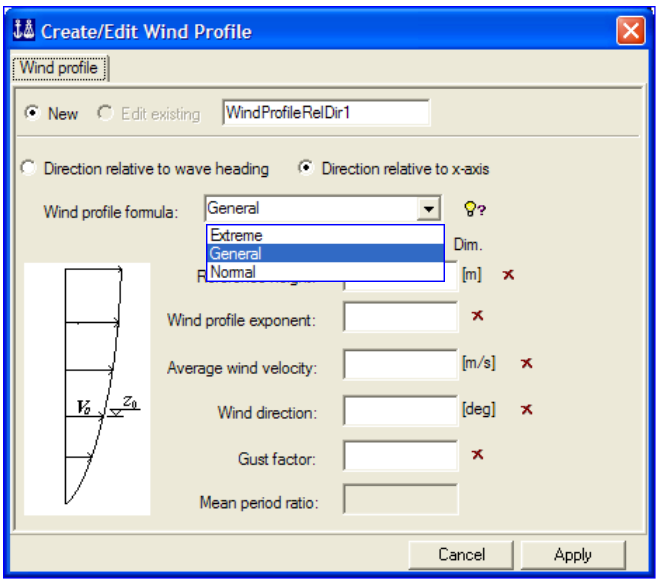


3.3.1.2 New Wind Profile

Purpose: To define a new wind profile. Notice that there are three ways of describing the wind profile formula.

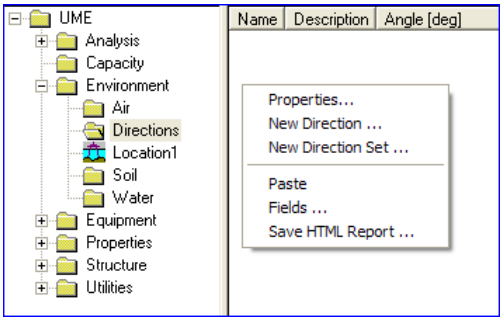
This command is scripted. By using a set of unrealistic wind profile parameters a typical script sequence may be: Reference height 50 m, exponent 1, wind velocity 5 m/s, direction 0 deg and gust factor 2.

WindProfileRelDir1 = WindProfileRelDir(5, 50, 1, dtRelativeX, wpGeneral, 0, 2);



3.3.2 Directions

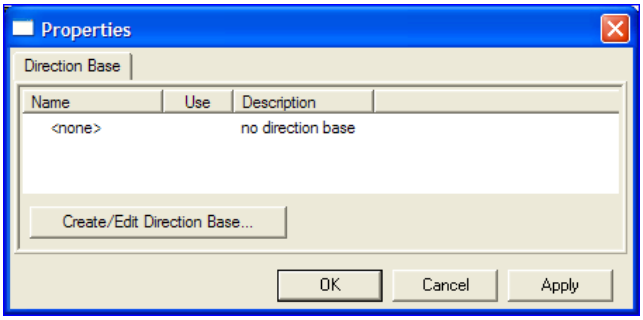
Purpose: To define discrete directions or a set of directions. These data can be used later when defining a wave load condition(s).



3.3.2.1 Properties

Purpose: To define or modify a direction or direction set.

This command is scripted. The scripting depends on which parameters are modified or entered to create a new direction or direction set, see below



3.3.2.2 New Direction

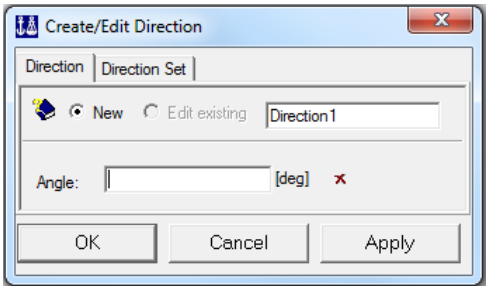
Purpose: To define a new direction.

This command is scripted. Typically when defining three directions with angles 0 deg, 15 deg and 30 deg:

Direction1 = Direction(0 deg);

Direction2 = Direction(15 deg);

Direction3 = Direction(30 deg);

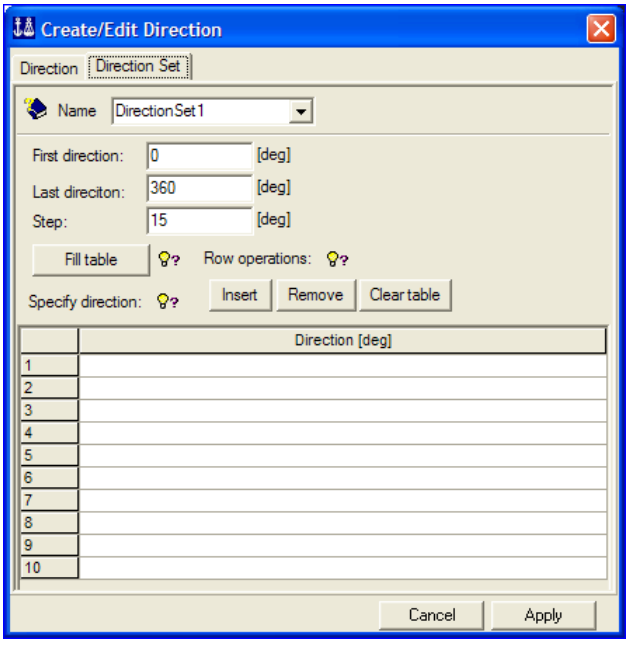


3.3.2.3 New Direction set

Purpose: To define a new direction set.

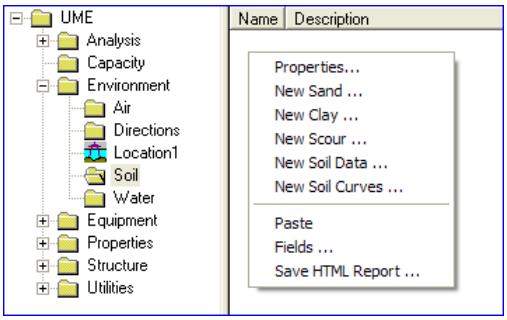
This command is scripted. Typically when using a step of 90 degrees between 0 deg and 360 deg:

DirectionSet1 = DirectionSet(Array(0 deg,90 deg,180 deg,270 deg));



3.3.3 Soil

Purpose: To define soil properties that can be used when defining a location. The soil properties may also be defined from the input dialogue for a location – for both alternatives the soil properties are found under the subfolder Soil.



3.3.3.1 Properties

Purpose: To define or modify soil properties.

This command is scripted. The scripting depends on which parameters are modified or entered to create a new soil property, see below.

3.3.3.2 New Sand

Purpose: To define sand properties.

This command is scripted. Typically angle 40 deg, mass 1.99 tonne/m^3, over-consolidation ratio 1, friction ratio 1 and t-z curve z displacement 1m:

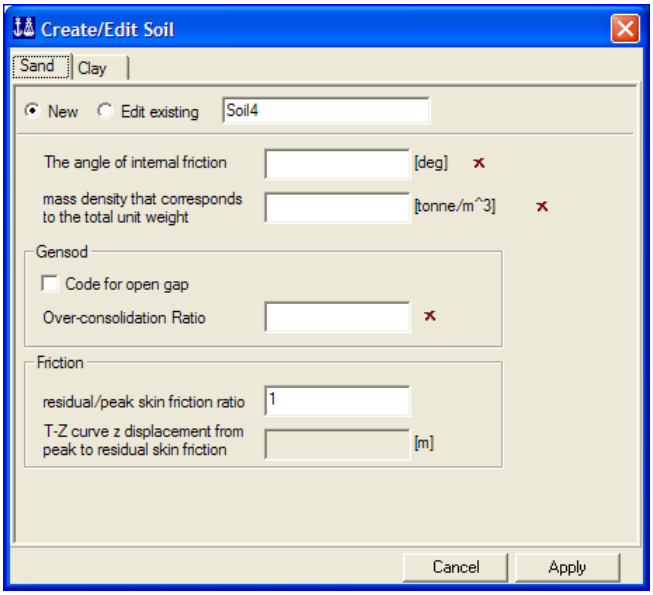
Sand1.ocRatio = 1;

Sand1.frictionRatio = 1;

Sand1.roTotal = 1.99 tonne/m^3;

Sand1.phi = 40 deg;

Sand1.zDisplacement = 0.99 m;



3.3.3.3 New Clay

Purpose: To define clay properties.

This command is scripted. Typically with relevant data as shown below:

Clay3.ocRatio = 1.0;

Clay3.frictionRatio = 1.0;

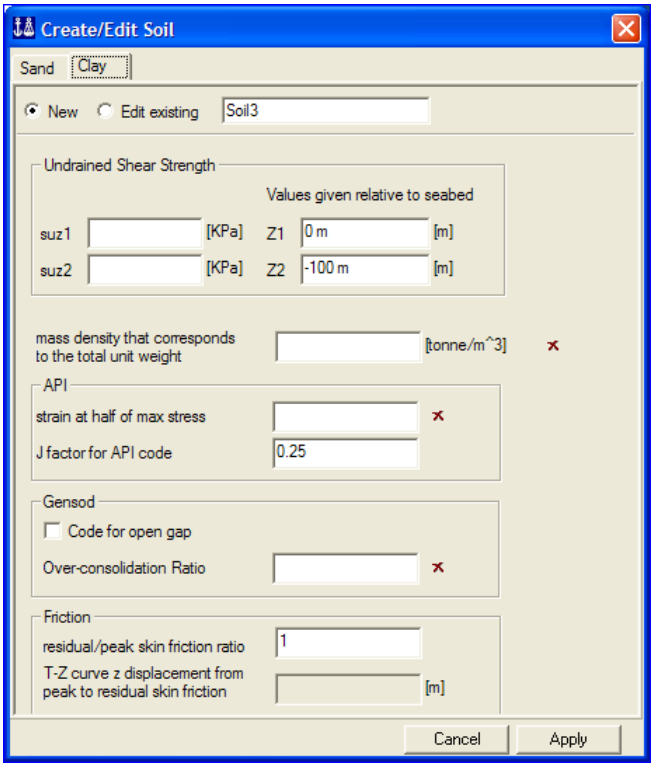
Clay3.zDisplacement = 1.0 m;

Clay3.roTotal = 1.94 tonne/m^3;

Clay3.apiJFactor = 0.5;

Clay3.epsc = 0.01;

Clay3.shearStrengthValues(200 KPa,0 m,100 KPa,-100 m);



3.3.3.4 New Scour

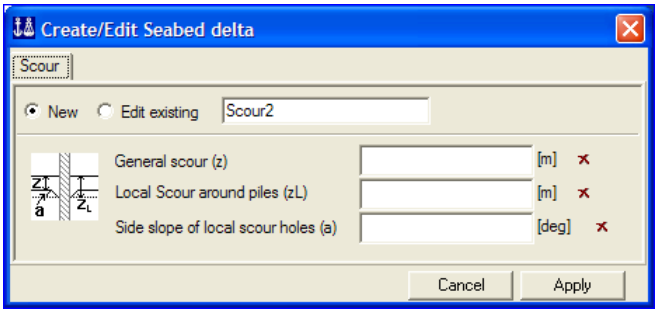
Purpose: To define scour properties.

This command is scripted. Typically with relevant data as shown below:

Scour1.localSlope = 20 deg;

Scour1.localScour = 1.0 m;

Scour1.generalScour = 0.5 m;



3.3.3.5 New Soil Data

Purpose: To define soil data properties.

This command is scripted. Typically:

SoilData1.peakSkinFrictionTension = 3 KPa;

SoilData1.peakSkinFrictionCompression = 5 KPa;

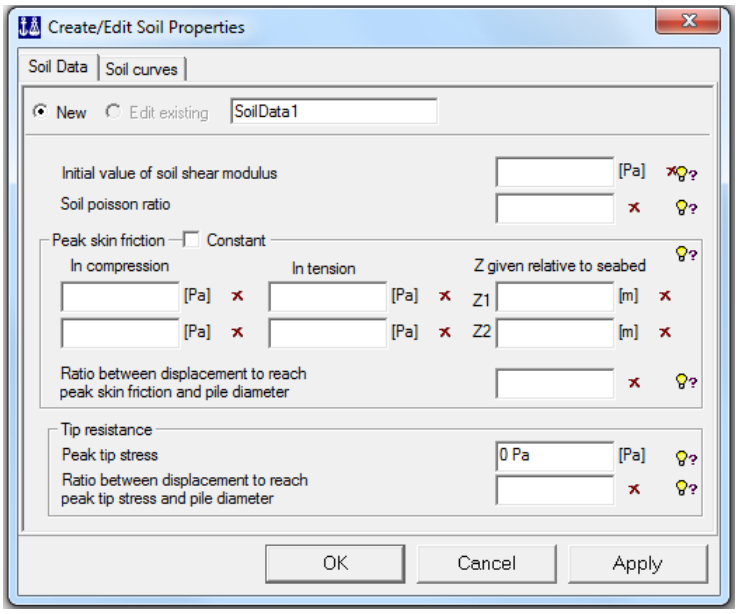
SoilData1.soilShearModulus = -1 KPa;

SoilData1.poissonRatio = 0.5;

SoilData1.peakTipStress = 0 KPa;

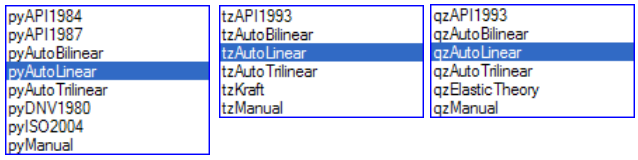
SoilData1.peakTipStressDisplDiamRatio = 0.05;

SoilData1.peakSkinFrictionDisplDiamRatio = 0.01;



3.3.3.6 New Soil Curves

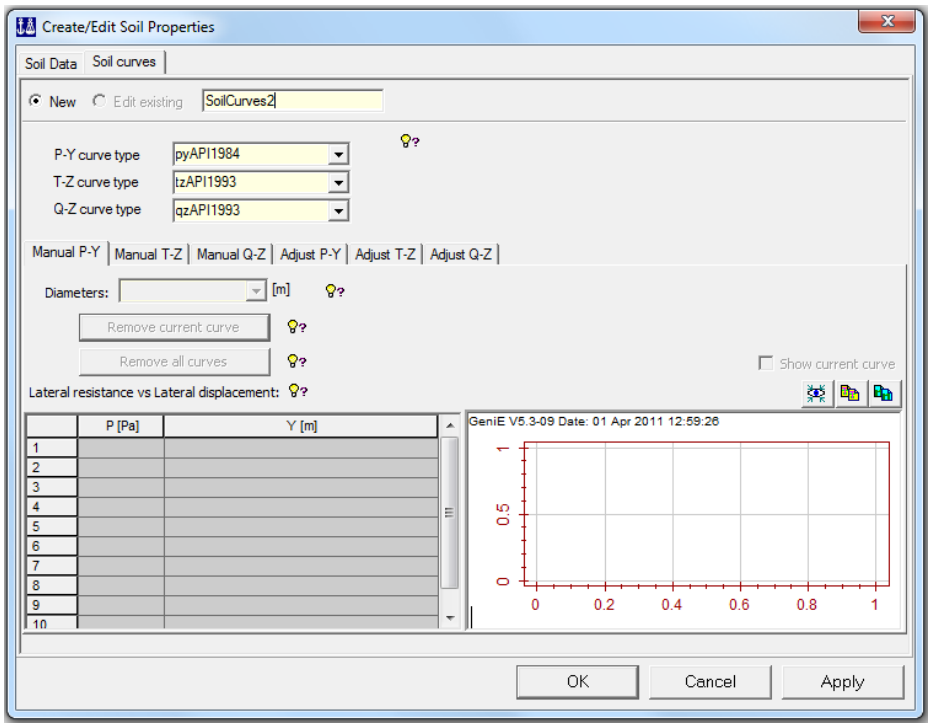
Purpose: To define soil curves properties. There are three sets of curves to choose between:



Notice that the manual option has been disabled.

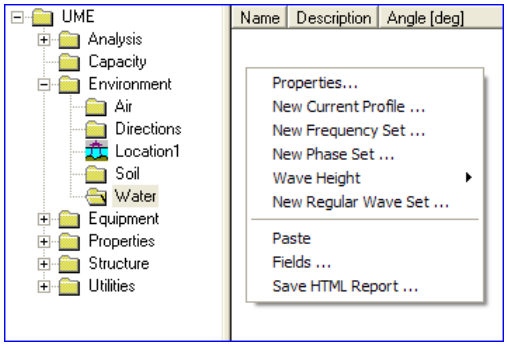
This command is scripted. Typically with relevant data as shown below:

SoilCurves2 = SoilCurves(pyAPI1984,tzAPI1993,qzAPI1993);



3.3.4 Water

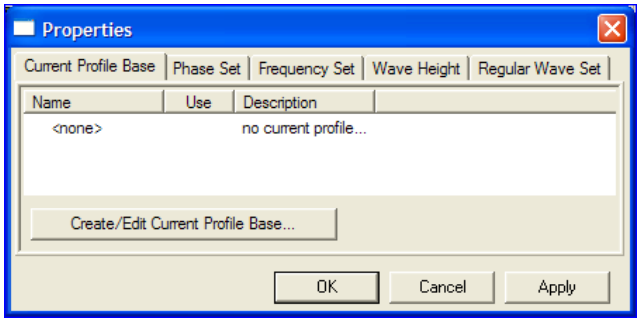
Purpose: To define water properties that can be used when defining a condition. The water properties may also be defined from the input dialogue for a condition – the water properties will be found under the subfolder Water.



3.3.4.1 Properties

Purpose: To define or modify water properties.

This command is scripted. The scripting depends on which parameters are modified or entered to create a new water property, see below.



3.3.4.2 New Current Profile

Purpose: To define a current property.

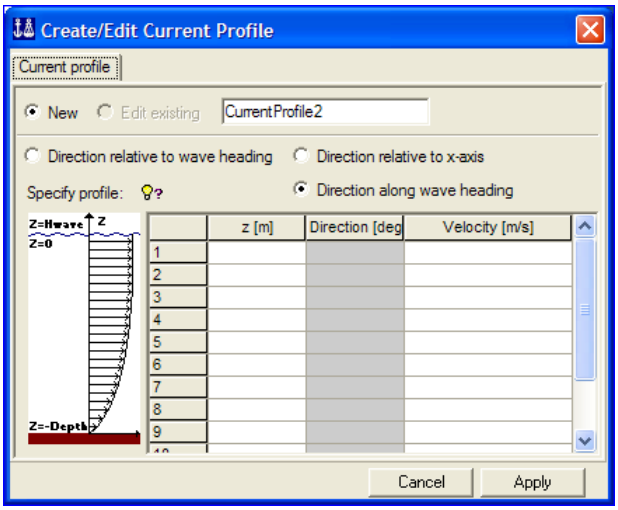
This command is scripted. Typically:

CurrentProfile2_Elevations = Array(30 m,0 m,-42 m,-124 m);

CurrentProfile2_Directions = Array(0 deg,0 deg,0 deg,0 deg);

CurrentProfile2_Velocities = Array(1.1 m/s,1.1 m/s,0.5 m/s,0 m/s);

CurrentProfile2 = CurrentProfileRelDir(CurrentProfile2_Elevations, CurrentProfile2_Directions, CurrentProfile2_Velocities,dtAlongHeading);

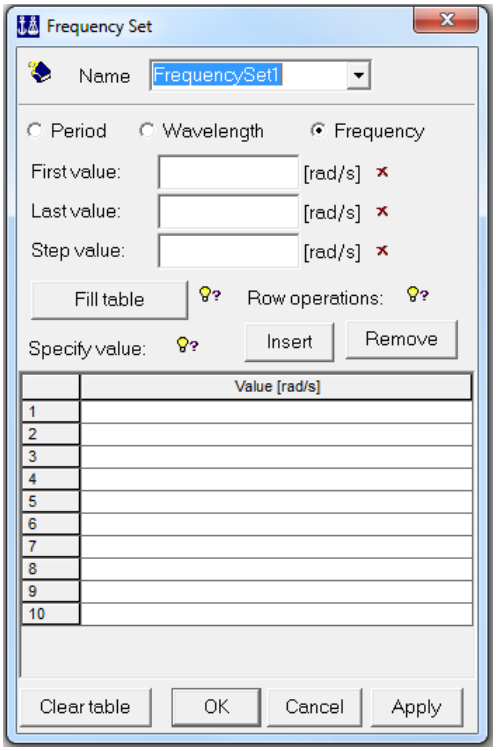


3.3.4.3 New frequency set

Purpose: To define a frequency set. The input may be in period (s), wavelength (m) or frequency (rad/s).

This command is scripted. Typically for period input: First 2 s, last 18 s, step 2 s:

FrequencySet1 = FrequencySet(WavePeriod(),Array(2 s,4 s,6 s,8 s,10 s,12 s,14 s,16 s,18 s));

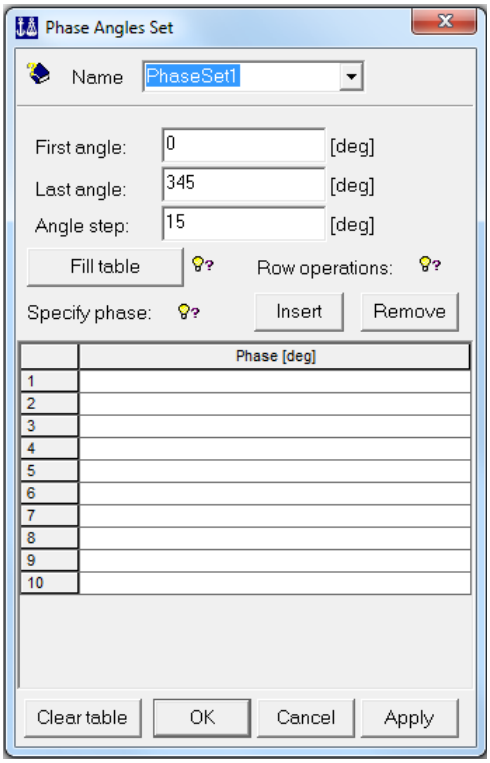


3.3.4.4 New Phase Set

Purpose: To define a phase set. The default values are shown to the right.

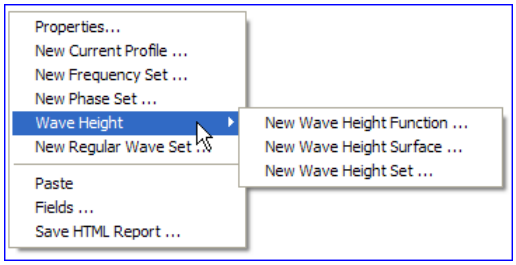
This command is scripted. Typically for first angle 0 deg, last angle 90 deg and angle step 15 deg:

PhaseSet1 = PhaseSet(Array(0 deg,15 deg,30 deg,45 deg,60 deg,75 deg,90 deg));



3.3.4.5 New Wave Height

Purpose: To define the wave height through a function, surface or set.

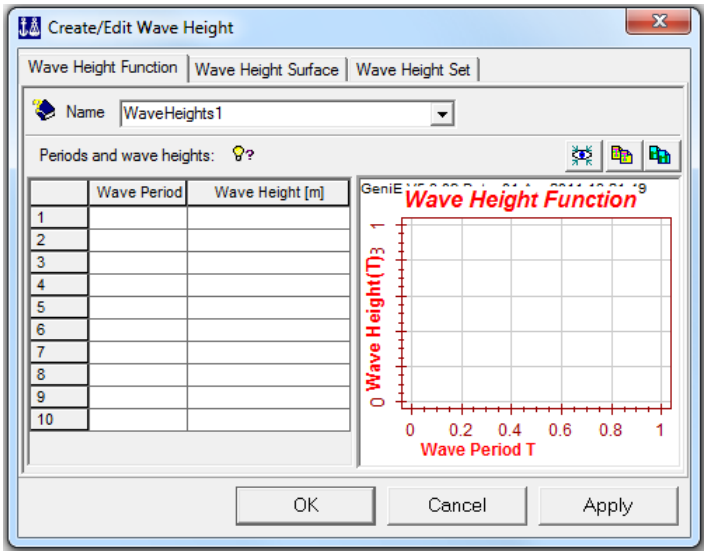


3.3.4.5.1 New Wave Height Function

Purpose: To define the wave height with corresponding wave periods and heights.

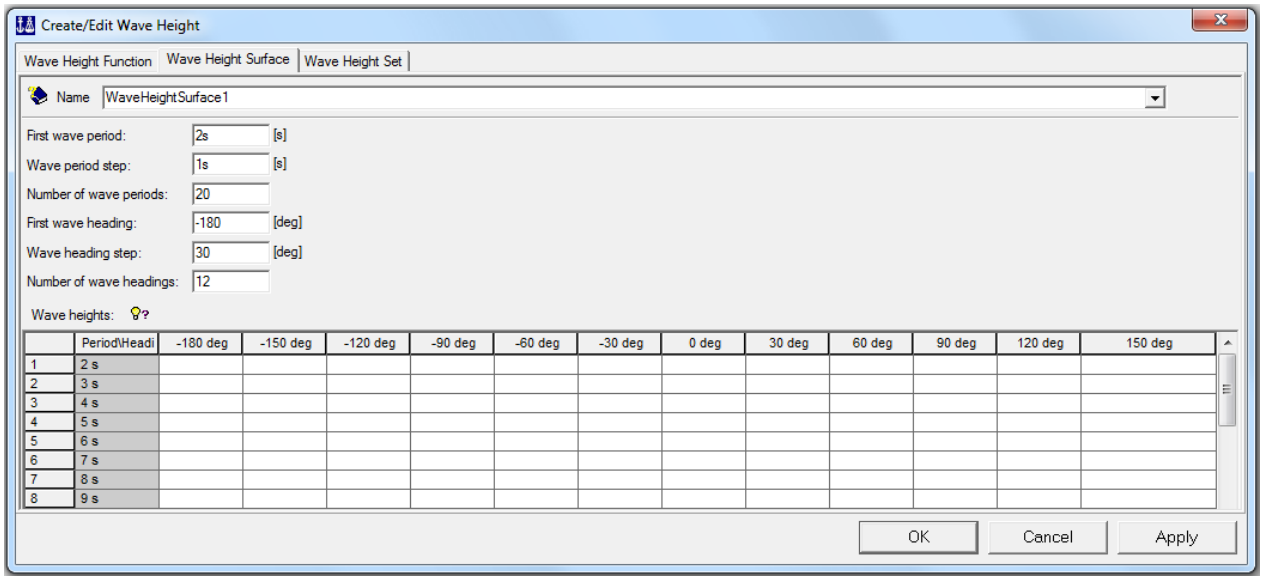
This command is scripted. Typically for three corresponding sets

WaveHeights1 = WaveHeightFunction(Array(5 s,7 s,9 s), Array(10 m,12 m,11 m));

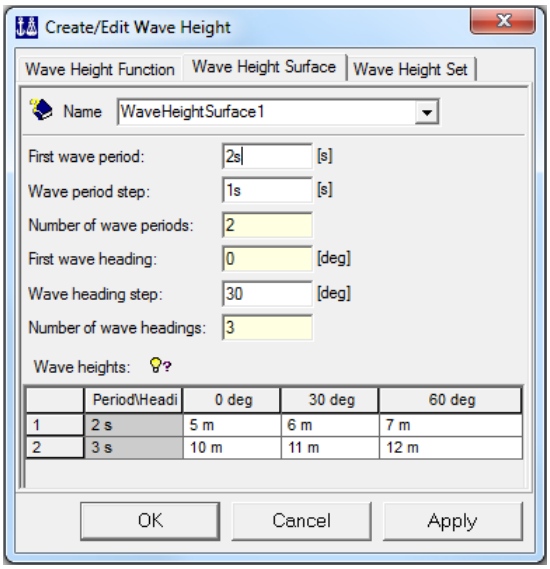


3.3.4.5.2 New Wave Height Surface

Purpose: To specify the wave heights as a function of wave heading and period.



This command is scripted. Typically when using the input parameters as shown to the right:



WaveHeightSurface1 = WaveHeightSurface(2s,1s,2,0,30,3);

WaveHeightSurface1.set(2 s,0 deg,5 m);

WaveHeightSurface1.set(2 s,30 deg,6 m);

WaveHeightSurface1.set(2 s,60 deg,7 m);

WaveHeightSurface1.set(3 s,0 deg,10 m);

WaveHeightSurface1.set(3 s,30 deg,11 m);

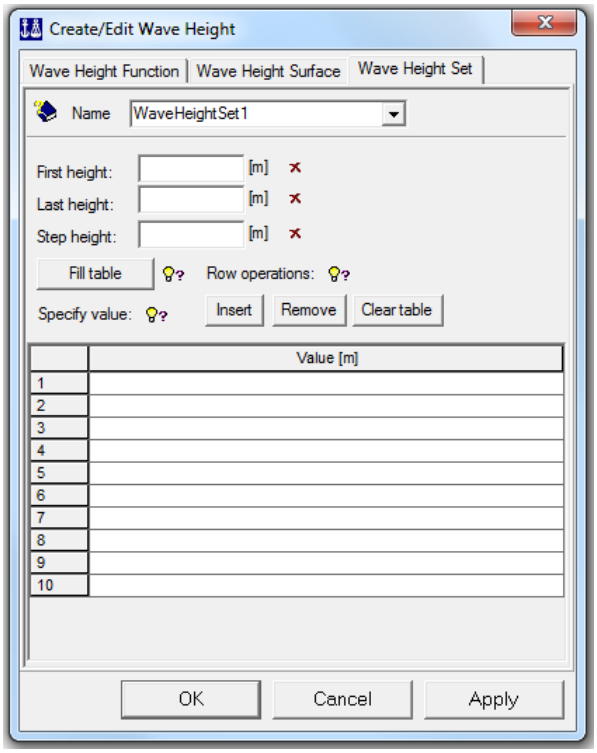
WaveHeightSurface1.set(3 s,60 deg,12 m);

3.3.4.5.3 New Wave Height Set

Purpose: To define a set of wave heights.

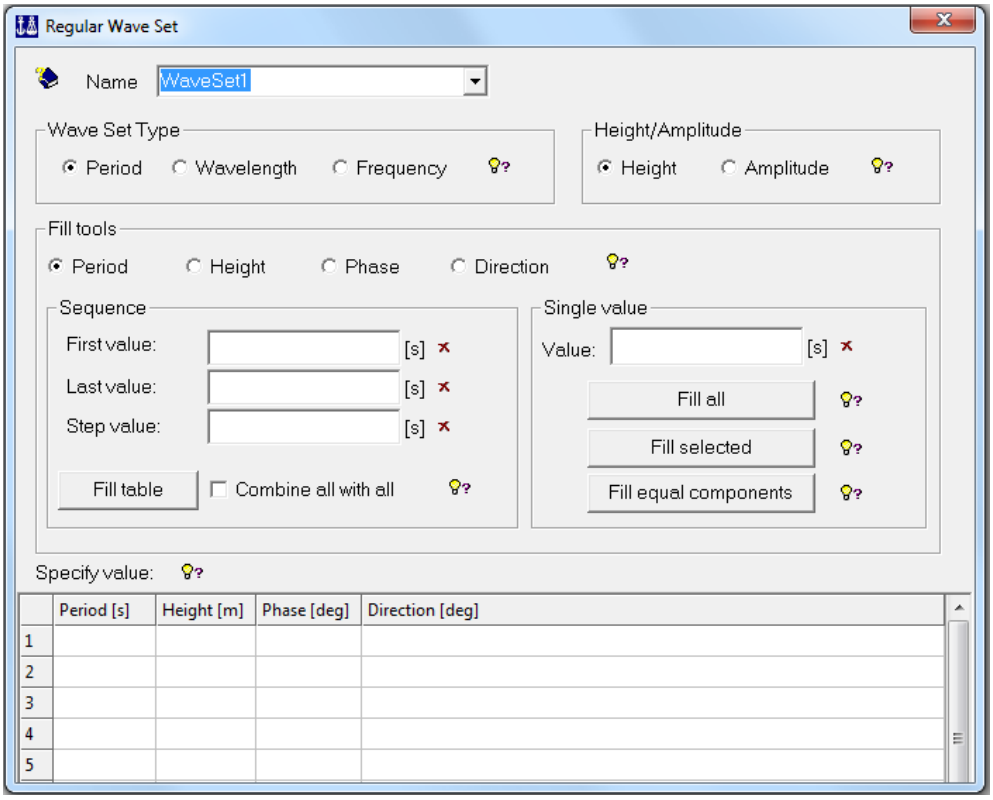
This command is scripted. Typically when using first height 2m, last height 20m and step height 2m:

WaveHeightSet1 = WaveHeightSet(Array(2 m,4 m,6 m,8 m,10 m,12 m,14 m,16 m,18 m,20 m));



3.3.4.6 New Regular Wave Set

Purpose: To define a regular wave set for use when defining a wave condition.



This command is scripted. Typically for input parameters as shown to the right:

	Period [s]	Height [m]	Phase [deg]	Direction [deg]
1	14 s	26 m	-60 deg	0 deg
2	12 s	15 m	-60 deg	90 deg
3	14 s	28 m	-60 deg	180 deg
4	15 s	27 m	-60 deg	270 deg
5				

```
WaveSet2 = RegularWaveSet();
WaveSet2.heightType(rwsHeight);
WaveSet2.add(RegularWave(0 deg,26 m,WavePeriod(14 s),-60 deg));
WaveSet2.add(RegularWave(90 deg,15 m,WavePeriod(12 s),-60 deg));
WaveSet2.add(RegularWave(180 deg,28 m,WavePeriod(14 s),-60 deg));
WaveSet2.add(RegularWave(270 deg,27 m,WavePeriod(15 s),-60 deg));
```

3.4 Equipment

1st level browser	*2nd level browser*	*3rd level browser*	*Available commands*
Equipment			***New Equipment***
	Weight list		***Import weight list***

3.5 Properties

1st level browser	*Available commands*	*Available sub-commands*
Common for all	***Fields***	
	Save HTML Report	
Beam Types	***Beam Types***	
	New Beam Type	
	Colour code all visible properties	
Compartment Content	***Contents***	
	New Content	
	Colour code all visible properties	
Corrosion Addition	***Corrosion Additions***	
	New Corrosion Addition	
	Colour code all visible properties	
Effective Flange	***Effective Flange***	
	New Effective Flange	
	Colour code all visible properties	
Hinges	***Hinges***	
	New Hinge	
Hydro	***Properties***	
	New Flooding	
	Morison	***New Morison Constant***
		New Morison Diameter
		New Morison Reynold
		New Morison KC
		New Morison By Rule
	Marine Growth	***New Marine Growth Constant***
		New Marine Growth Z-level
	Air Drag	***New Air Drag Constant***
		New Air Drag Reynold
	New Hydrodynamic Diameter	***New Hydrodynamic Diameter***

	New Conductor Shielding	***New Conductor Shielding***
	New Element Refinement	***New Element Refinement***
Load Interfaces	***Load Interfaces***	
	New Load Interface	
Mass Density Factors	***New Property***	
Materials	***Materials***	
	New Material	
	Colour code all visible properties	
Mesh	***Mesh Properties***	
	New Mesh Properties	
	Colour code all visible properties	
Mesh Options	***Mesh Options***	
	New Mesh Option	
	Colour code all visible properties	
Permeability	***Permeability***	
	New Permeability	
	Colour code all visible properties	
Pile Characteristics	***Pile Characteristics***	
	New Pile Characteristics	
	Colour code all visible properties	
Plate Types	***Plate Types***	
	New Plate Type	
	Colour code all visible properties	
Reinforcements	***Reinforcements***	
	New Reinforcement	
	Colour code all visible properties	
Sections	***Sections***	
	New Section	
	Colour code all visible properties	
Structure Type	***New Structure Type***	
	Structure Type	
	Colour code all visible properties	
Thicknesses	***Thicknesses***	
	New Thickness	
	Colour code all visible properties	

1st level browser	*Available commands*	*Available sub-commands*
Wet Surface	***Wet Surfaces***	
	New Wet Surface	
	Colour code all visible properties	

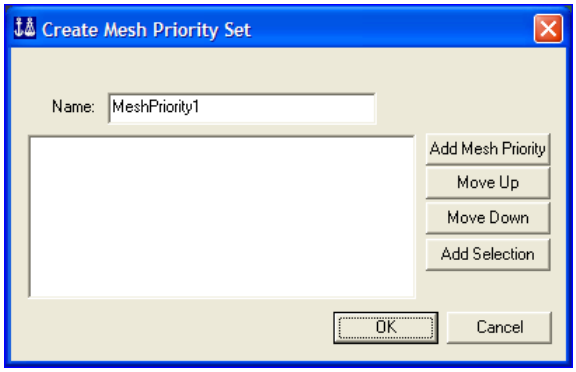
3.6 Structure

1st level browser	*2nd level browser*	*3rd level browser*	*Available commands*
Structure			***New Beam***
			New Plate
			New Support
			New Joint
	Features		***New Feature Edge***
	Point Masses		***New Point Mass***
	Supports		***New Support***
			New Support Rigid Link

3.7 Utilities

1st level browser	*2nd level browser*	*3rd level browser*	*Available commands*
Utilities			
	Evaluators		***New Linear Slicer***
	Guiding Geometry		***New Guide Plane***
			New Guide Point
		Curves	***New Poly Curve***
			New Guide Line
		Points	***New Guide Point***
		Profiles	***New Profile***
		Transformations	***New Transform***
	Mesh Priorities		***New Mesh Priority***
	Model Views		
	Reports		
	Sets		***New Set***

3.7.1 Mesh Priority Sets

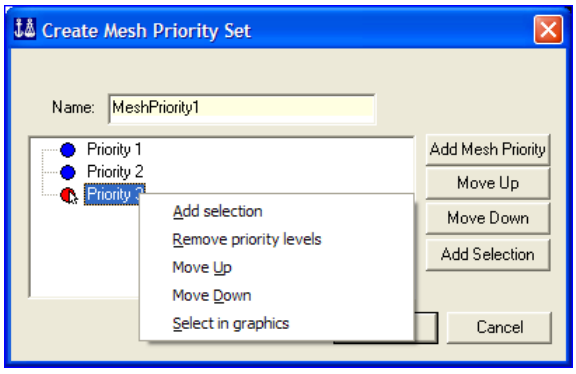


Purpose: To define mesh priority sets that can be used during meshing. A mesh priority set will denote the meshing sequence.

A mesh priority set is defined from the command ***New Mesh Priority*** by right clicking the sub-folder Mesh Priorities

There can be several mesh priority sets, but only one can be used for a meshing activity.

When pushing the button "Add Mesh Priority" a new priority is created. There can be many mesh priorities.



Each priority level contains structural parts and it is possible to move these up and down. The "Add selection" requires that the structural parts are selected in advance. Remember to click the button "Add Selection" at the far right to include the selection in the priority. The "Select in graphics" allows you to select during the priority definitions; the "Add Selection" button must also be used in this case.

This command is scripted. Typically for a mesh priority set with three priorities containing Bm1, Bm10 and Bm100 respectively:

MeshPriority1 = MeshPriority();

MeshPriority1.addMeshPriority();

MeshPriority1.meshPriority(1).add(Bm1);

MeshPriority1.addMeshPriority();

MeshPriority1.meshPriority(2).add(Bm10);

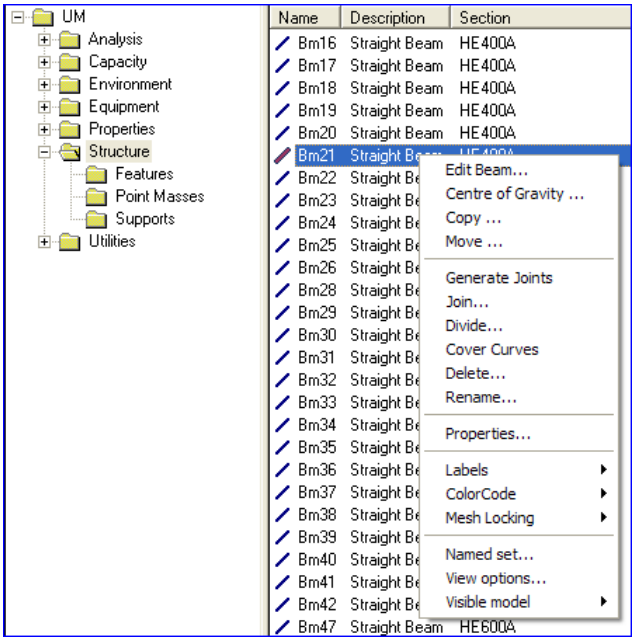
MeshPriority1.addMeshPriority();

MeshPriority1.meshPriority(3).add(Bm100);

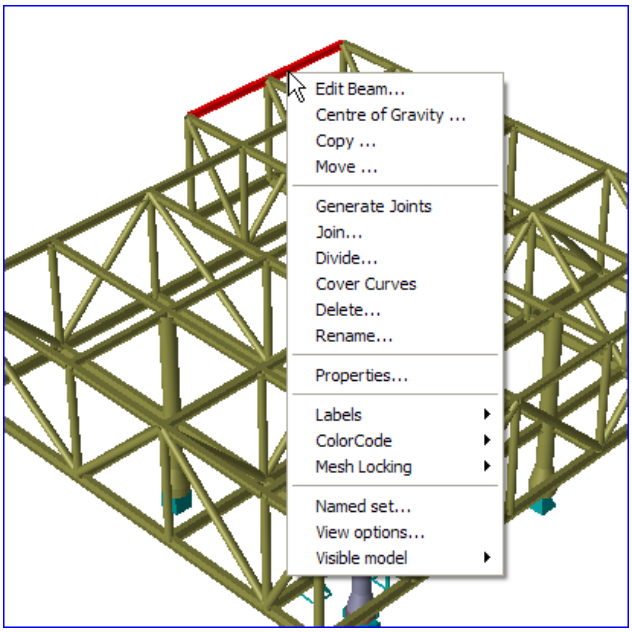
4. CONTEXT SENSITIVE MENU

This Chapter describes how the context sensitive menu works as well as lists those commands only available from this menu. The context sensitive menu is activated by selecting an object(s) from the browser or from the graphics.

Example of context sensitive menu for a beam selected from the browser. Select the beam and right click.



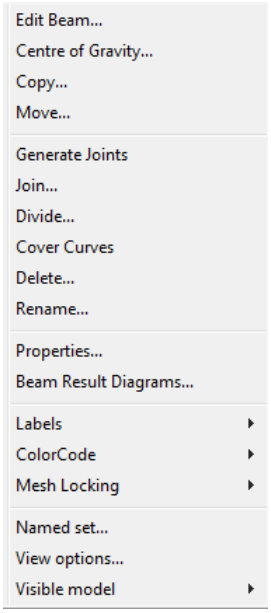
Example of context sensitive menu for a beam selected from the graphics. Select the beam and right click.



The context sensitive menu differs from object to object. But they have several equal commands and the object type “beam” is used to exemplify the context sensitive menu.

4.1 Object type Beam

Note that Edit Beam, Centre of Gravity, Move, Join Beams, Join Segments, Cover Curves, Labels, Mesh Locking and Visible Model are not available from pulldown or from toolbar menus.



4.1.1 Edit Beam

Purpose: To modify beam specific parameters like local system, offset vector, hinges, split points, move ends, translate and buckling factors.

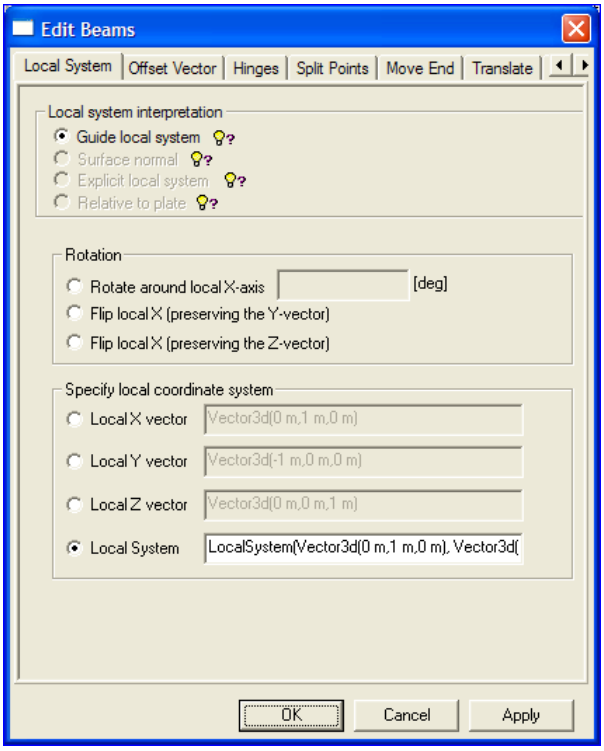
These commands are scripted. The content depends on which operations are carried out. See below for examples.

4.1.1.1 Local system

Purpose: To modify the local system of a beam.

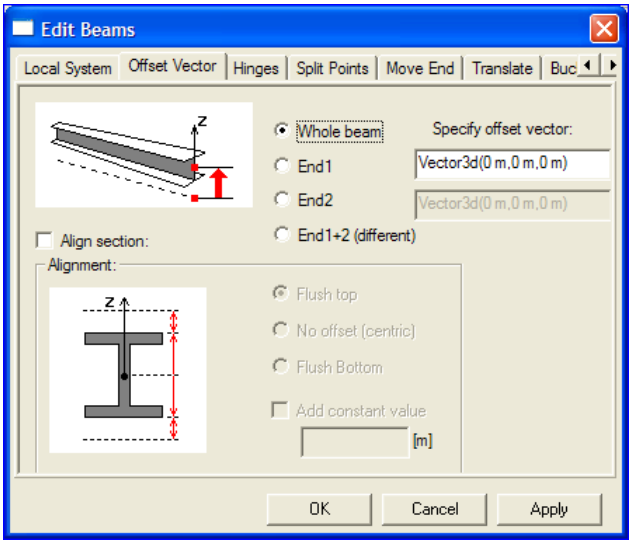
This command is scripted. Typically when rotating the beam Bm21 90 degrees around the beam local x-axis:

Bm21.rotateLocalX(90);



4.1.1.2 Offset vector

Purpose: To add eccentricities to the beam ends.

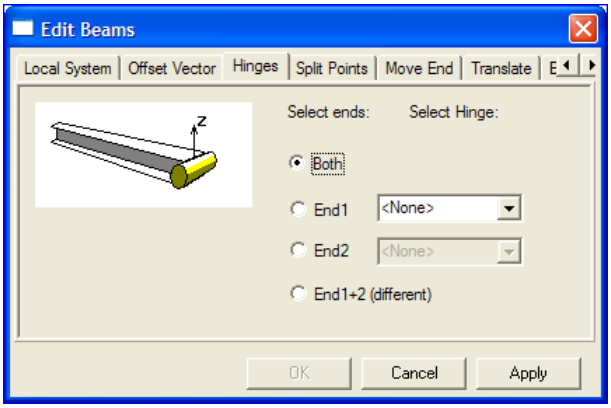


This command is scripted. Typically when specifying an eccentricity of 5cm in negative global z-direction for beam Bm21:

Bm21.setBeamOffset(Vector3d(0 m,0 m,-5cm));

4.1.1.3 Hinges

Purpose: To add a hinge to a beam end. The hinge properties are defined from the ***Edit|Property|Hinge*** or from the browser folder *Properties|Hinges*.

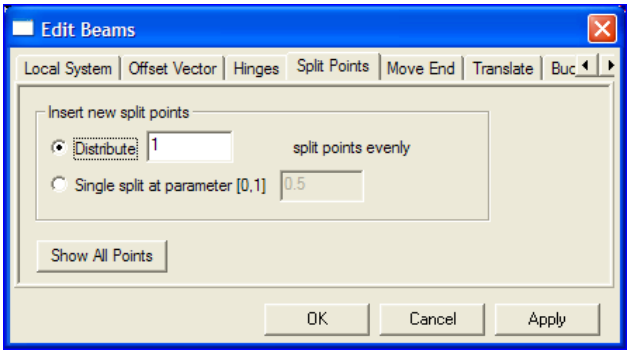


This command is scripted. Typically when adding the hinge "Hinge1" to end1 of Bm1:

Bm21.setEndHinge(1,Hinge1);

4.1.1.4 Split points

Purpose: To add additional vertices (or snap points) to a beam.



This command is scripted. Typically when distributing 4 points evenly of Bm31:

Bm31.splitAt(0.2);

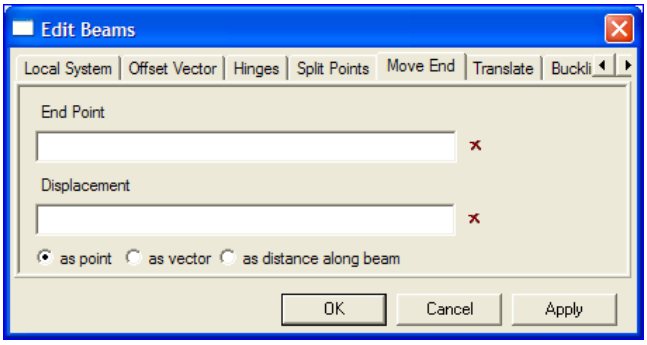
Bm31.splitAt(0.4);

Bm31.splitAt(0.6);

Bm31.splitAt(0.8);

4.1.1.5 Move End

Purpose: To move one beam end. The end point can be moved to another point, by a vector definition or extended/reduced along its own local x-axis. In the latter case a positive value means extension of the beam while a negative value means a reduction of the length.



This command is scripted. Typically when reducing the length with 150 cm relative to the end point 2 of Bm31:

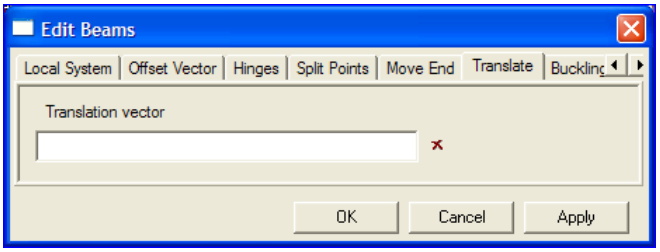
Bm31.extendEnd(2, -150 cm);

Similarly, when moving the end1 a vector of (0,5m,0):

Bm31.moveEnd(1, Vector3d(0 m,5 m,0 m));

4.1.1.6 Translate

Purpose: To move a beam with a translation vector.



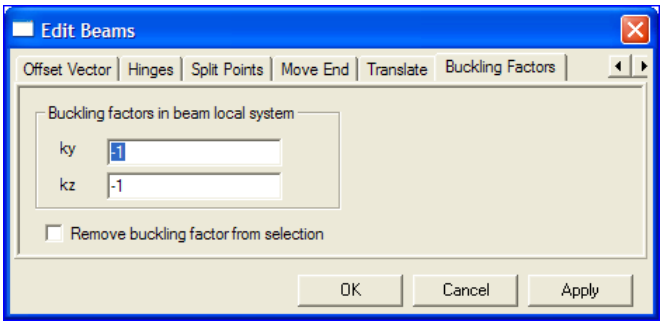
This command is scripted. Typically when moving Bm21 2000mm in positive global z-direction):

Bm31.extendEnd(2, -150 cm);

Bm21.moveTranslate(Vector3d(0m,0m,2000mm));

4.1.1.7 Buckling factors

Purpose: To define local bucking factors for individual beams. These factors can be referred to when defining a capacity model. These factors are also transferred to Framework if you want to do code checking based on older code checking standards. See User Manual Volume 4 for more details.



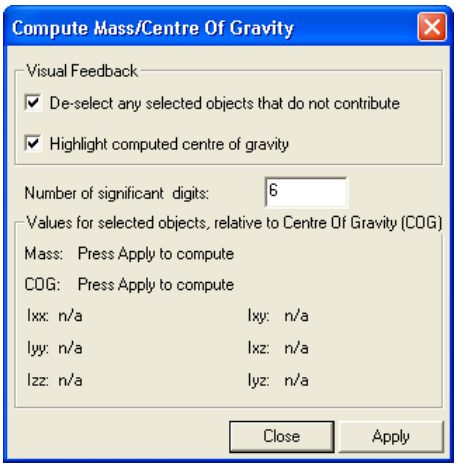
This command is scripted. Typically when specifying buckling factor of 0.9 and 0.8 for buckling around local y and z-axis.

Bm21.buckling = BucklingFactor(0.9, 0.8);

4.1.2 Centre of Gravity

Purpose: To compute the mass and centre of gravity.

This command is not scripted.



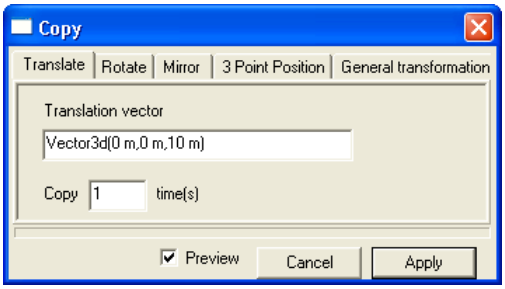
4.1.3 Copy/Move Beam

Purpose: To copy or move a beam using translation, rotation, mirroring, 3 point position or a general transformation.

These commands are scripted. The content depends on which operations are carried out. See below for examples.

4.1.3.1 Translate

Purpose: To copy a beam using a translation vector.

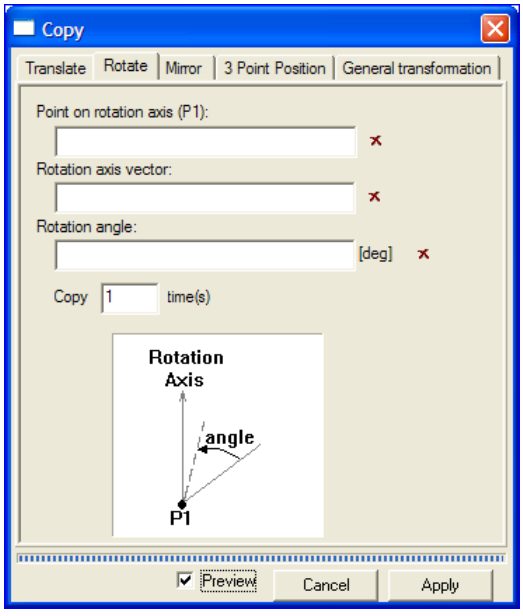


This command is scripted. Typically copying Bm1 with the shown vector:

Bm2 = Bm1.copyTranslate(Vector3d(0 m,0 m,10 m));

4.1.3.2 Rotate

Purpose: To copy a beam by defining a rotation point, a vector and a rotation angle.

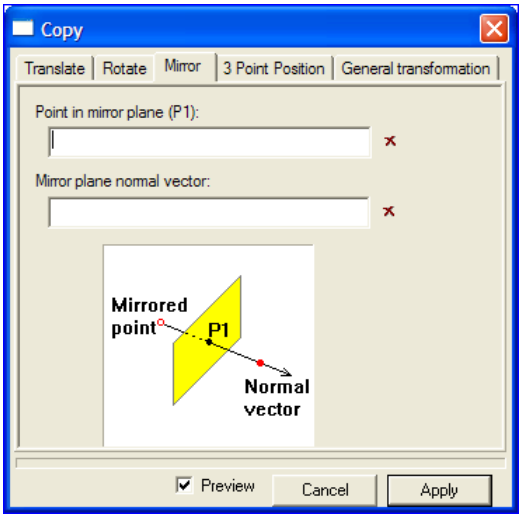


This command is scripted. Typically copying Bm1 around point (0,0,0) with vector (0,0,1) and angle 90 degrees:

Bm103 = Bm21.copyRotate(Point(0 m,0 m,0 m), Vector3d(0,0,1),90 deg);

4.1.3.3 Mirror

Purpose: To copy a beam by defining a mirror plane built up from a point and a normal vector.



This command is scripted. Typically copying Bm1 around point (0,5,0) with vector (0,1,0):

Bm2 = Bm1.copyMirror(Point(0 m,5 m,0 m), Vector3d(0,1,0));

A typical move operation would be

autoMSet = Set();

autoMSet.clear();

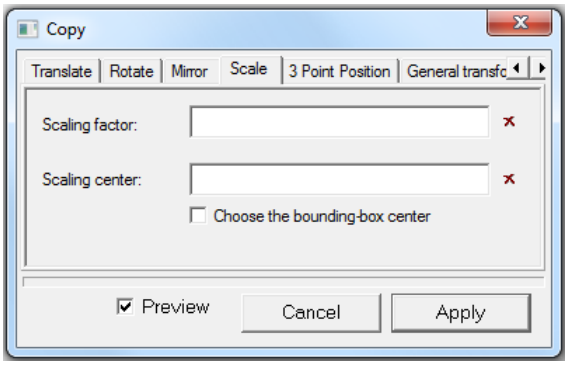
autoMSet.add(Bm1);

autoMSet.moveMirror(Point(0 m,5 m,0 m), Vector3d(0,1,0),geUNCONNECTED);

Delete(autoMSet);

4.1.3.4 Scale

Purpose: To copy a beam by defining a scaling factor and a point to be used as the scaling centre.



This command is scripted

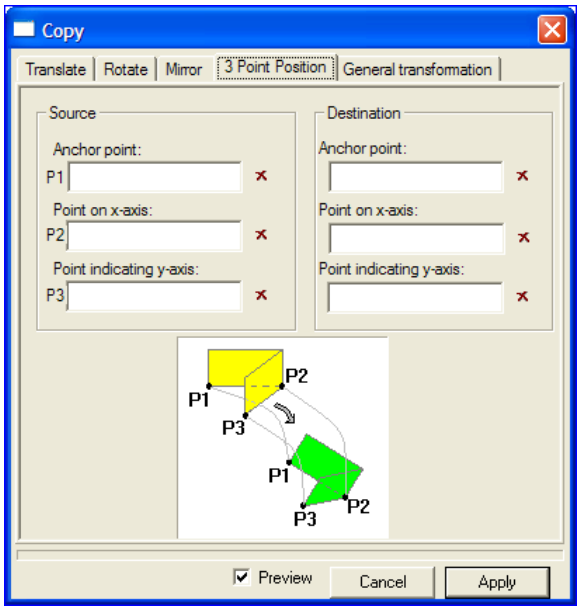
A typical scale operation would be

Beam1_1 = Beam1.copyScale(5,Point(0,0,0));

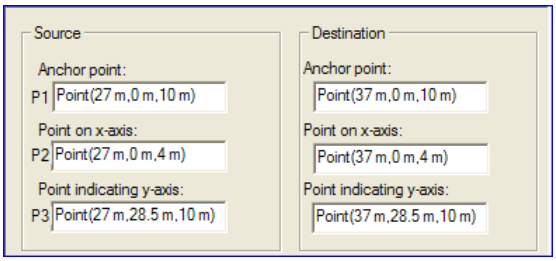
Beam2_1 = Beam2.copyScale(5,Point(0,0,0));

4.1.3.5 3 Point Position

Purpose: To copy beams by defining a source plane and a destination plane. For both planes three points must be defined.



This command is scripted. Typically copying Bm1and Bm2 using the following input parameters

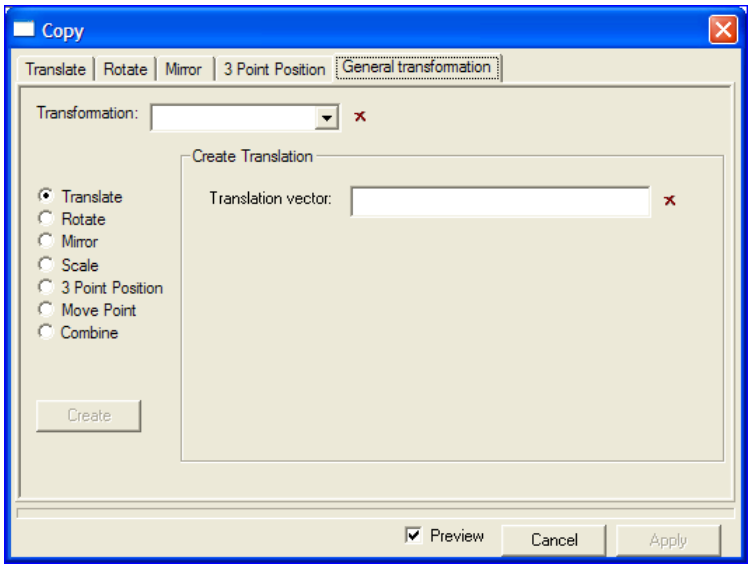


Bm11 = Bm1.copy3Point(Point(27 m,0 m,10 m),Point(27 m,0 m,4 m),Point(27 m,28.5 m,10 m), Point(37 m,0 m,10 m),Point(37 m,0 m,4 m),Point(37 m,28.5 m,10 m));

Bm12 = Bm2.copy3Point(Point(27 m,0 m,10 m),Point(27 m,0 m,4 m),Point(27 m,28.5 m,10 m), Point(37 m,0 m,10 m),Point(37 m,0 m,4 m),Point(37 m,28.5 m,10 m));

4.1.3.6 General transformation

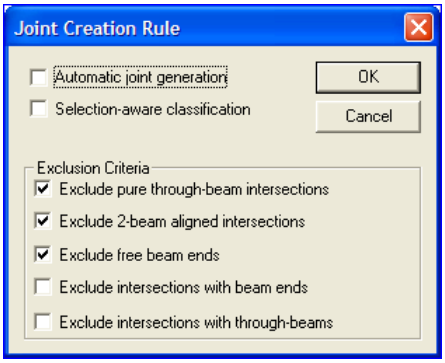
Purpose: To copy a beam by defining a general transformation. For examples, see Section 3.3.12 of User Manual Volume 3.



This command is scripted depending on which operation is carried out.

4.1.4 Generate Joints

Purpose: To generate joints based on the rules as defined in the Edit|Rules|Joint creation.



This command is scripted. Typically when selecting a beam that is subject to definition of four joints based on the rule definition:

Jt1 = Joint(Point(4 m,0 m,10 m));

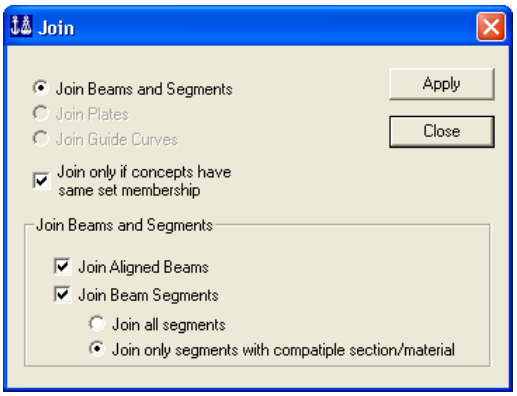
Jt2 = Joint(Point(12 m,0 m,10 m));

Jt3 = Joint(Point(18 m,0 m,10 m));

Jt4 = Joint(Point(24 m,0 m,10 m));

4.1.5 Join

Purpose: To join beams and segments.



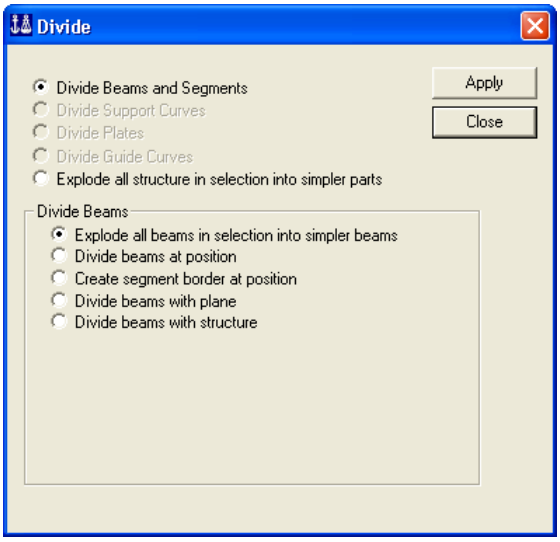
This command is scripted. Typically when joining Bm144 and Bm145:

Bm144.joinBeams(Bm145);

Bm144.joinSegments(1,2);

4.1.6 Divide

Purpose: To divide beams and segments using a number of division criteria.



This command is scripted. Typically when dividing Bm1 with an X-plane at x= 24 m (there are now two beams Bm1 and Bm2):

Bm2 = Bm1.divide(XPlane3d(24 m));

4.1.7 Cover Curves

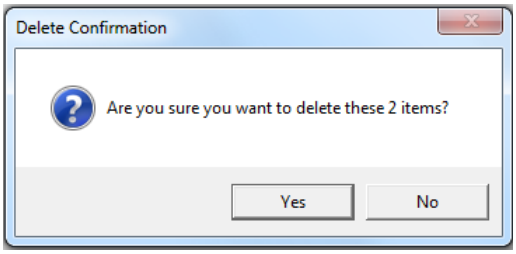
Purpose: To create a plate by covering the area enclosed by the selected beams.

This command are scripted. Typically:

Pl1 = CoverCurves(Beam1,Beam2,Beam3,Beam4);

4.1.8 Delete

Purpose: To delete the selection.



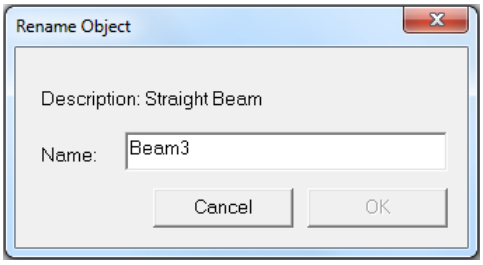
This command is scripted. Typically:

Delete(Bm1);

Delete(Bm2);

4.1.9 Rename

Purpose: To rename the selected item.

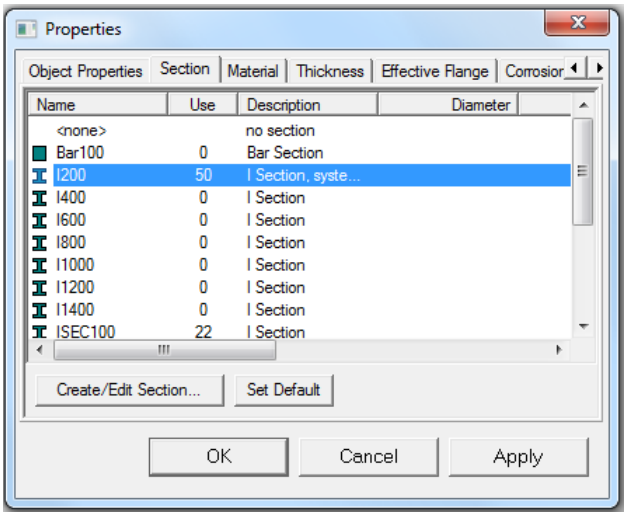


This command is scripted. Typically:

Rename(Beam3,"Beam3_renamed");

4.1.10 Properties

Purpose: To add or modify properties to the selected beam.



These commands are scripted. Typically when adding profile type HE600A to Bm1:

Bm1.section = HE600A;

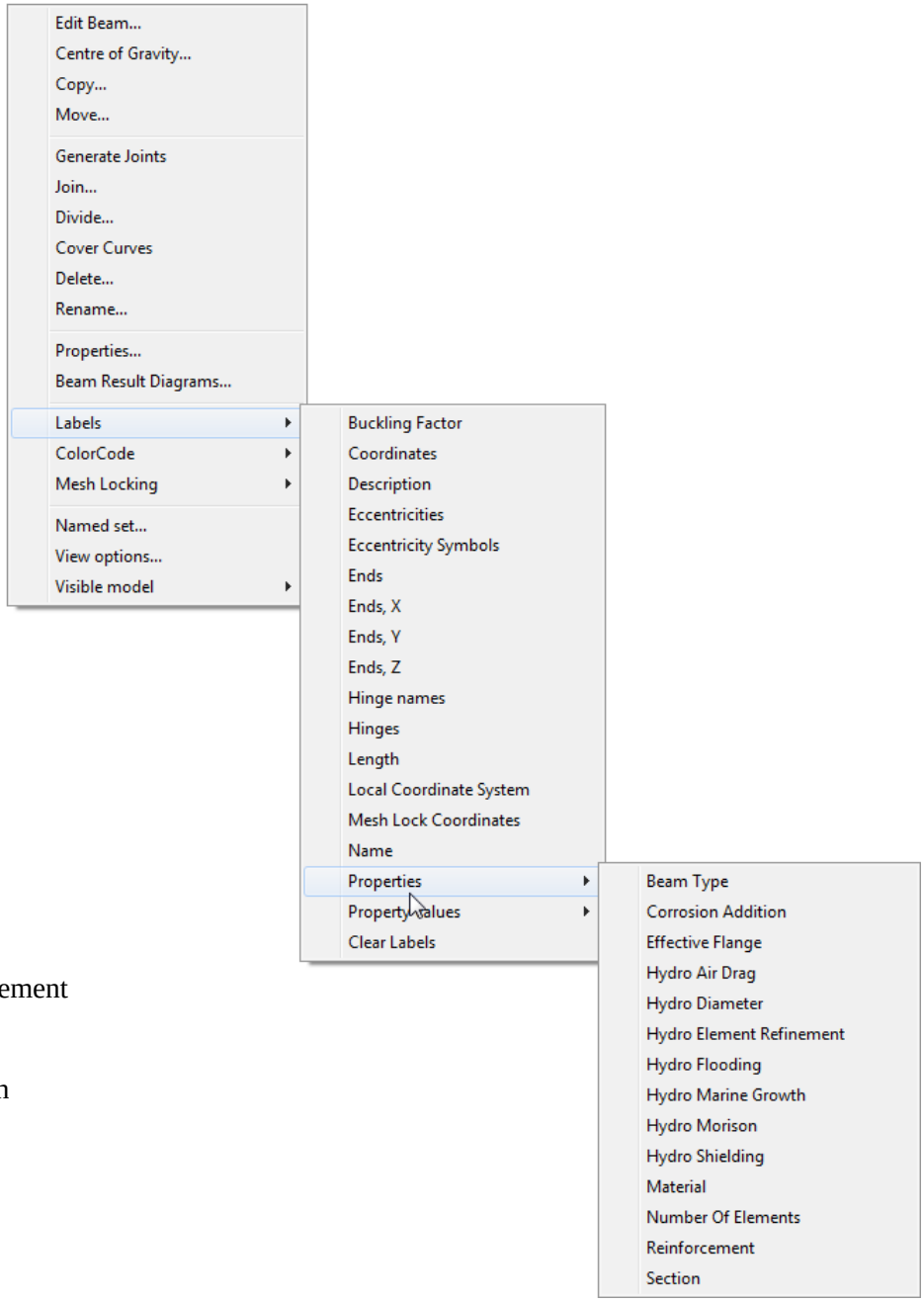
4.1.11 Beam Result Diagrams

Purpose: To open the “Beam Deflection, Force and Stresses Display” dialog.

This command is not scripted.

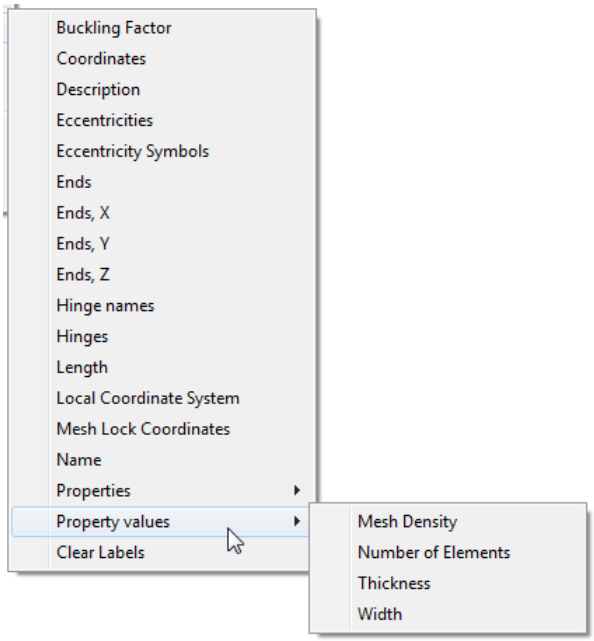
4.1.12 Labels

It is possible to label the following parameters. To remove the labels from the graphic view, use the "Clear labels" option or click the refresh button: 



- Buckling Factor
- Coordinates
- Description
- Eccentricities
- Eccentricity Symbols
- Ends
- Ends, X
- Ends, Y
- Ends, Z
- Hinge names
- Hinges
- Length
- Local Coordinate System
- Name
- Properties
  - Beam Type
  - Corrosion Addition
  - Hydro Air Drag
  - Hydro Diameter
  - Hydro Element Refinement
  - Hydro Flooding
  - Hydro Marine Growth
  - Hydro Morison
  - Hydro Shielding
  - Material
  - Number Of Elements
  - Reinforcements
  - Section

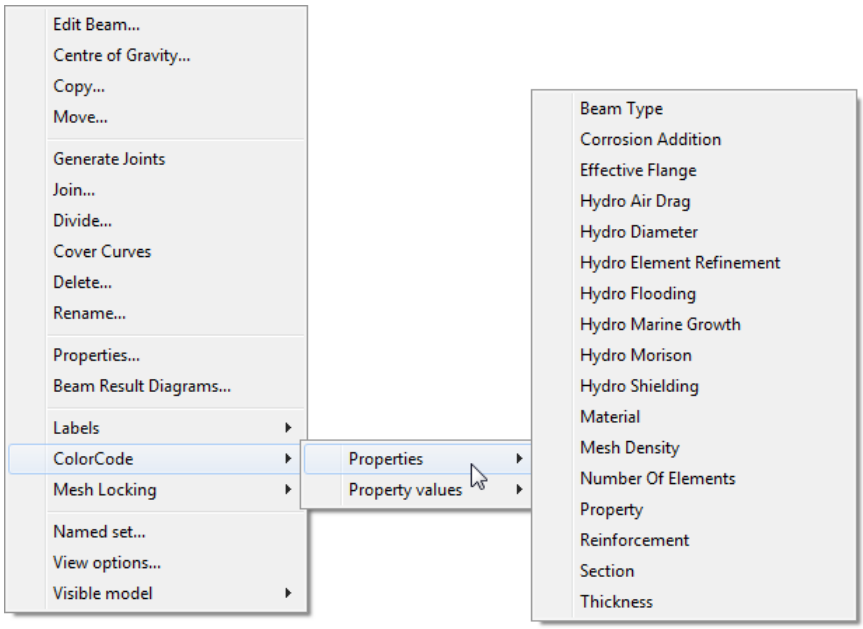
- Property Values
  - Mesh Density
  - Number Of Elements
  - Thickness
  - Width
- Clear Labels



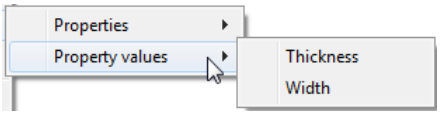
4.1.13 Colour Coding

It is possible to do colour coding of the following parameters. To view the structure without the colour coding you can click on the paint brush symbol to toggle on/off colour coding. 

- Properties
  - Beam Type
  - Corrosion Addition
  - Effective Flange
  - Hydro Air Drag
  - Hydro Diameter
  - Hydro Element Refinement
  - Hydro Flooding
  - Hydro Marine Growth
  - Hydro Morison
  - Hydro Shielding
  - Material
  - Number Of Elements
  - Property
  - Reinforcements
  - Section
  - Thickness



- Property Values
  - Thickness
  - Width

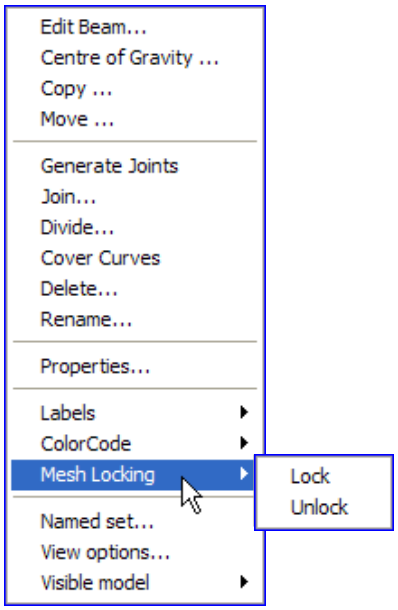


4.1.14 Mesh Locking

It is possible to lock or unlock the mesh.

- Lock
- Unlock

These commands are scripted.



4.1.15 Named Set

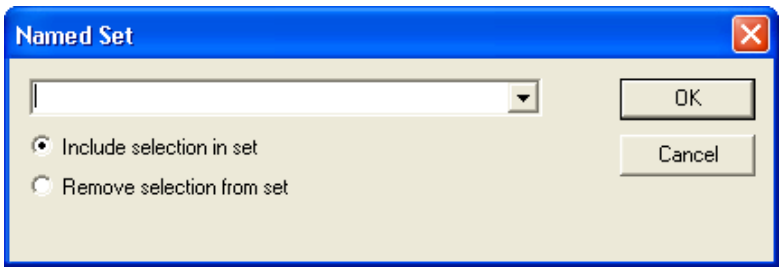
Purpose: To include or remove a selection in a set.

This command is scripted. Typically when including two beams in the set "MySet"

MySet = Set();

MySet.add(Bm1);

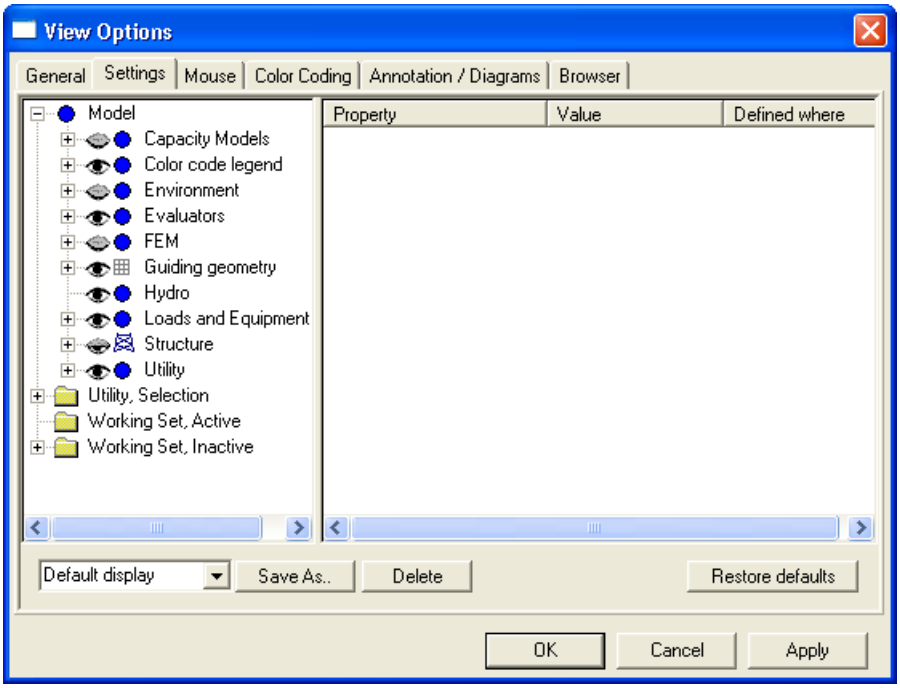
MySet.add(Bm2);



4.1.16 View Options

Purpose: Access to the same view settings as defined under the ***View|Option*** pulldown menu.

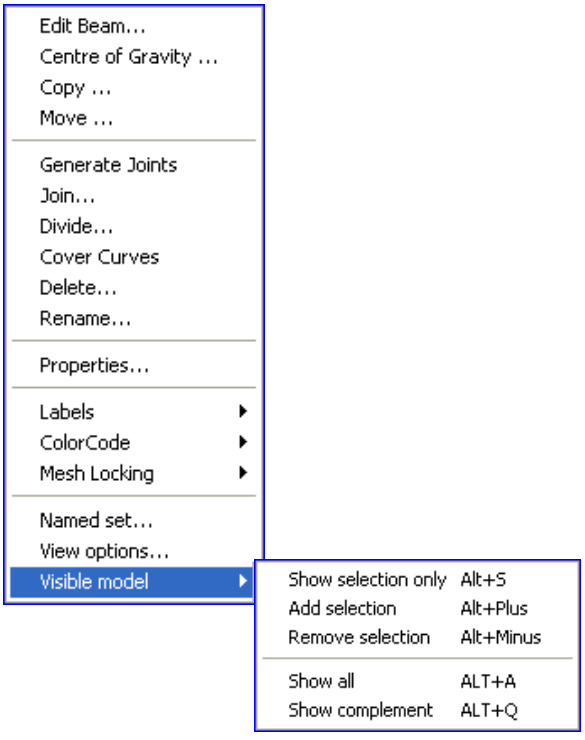
These commands are not scripted.



4.1.17 Visible Model

Purpose: To decide which parts of a model to show in the graphic view. It is possible to:

- Show selection only (Alt + S)
- Add selection (Alt + Plus)
- Remove selection (Alt + Minus)
- Show All (Alt + A)
- Show Complement (Alt + Q)

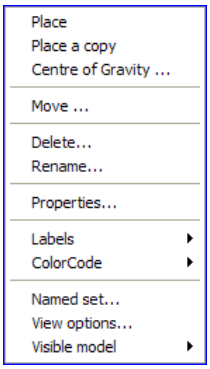


4.2 Object type Plate

Note that Centre of Gravity, Move, Flip Normal, Labels, Named Set, and Visible Model are not available from pulldown or from toolbar menus.

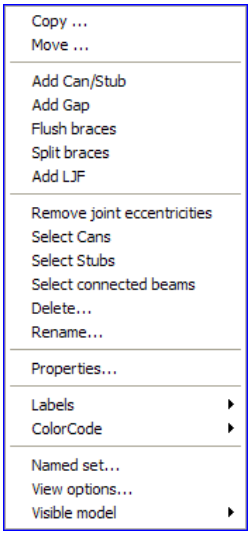
4.3 Object type Equipment

Note that Place, Place a Copy, Named Set, Visible Model are not available from pulldown or from toolbar menus.



4.4 Object type Joint

Note that most of these are not available from pulldown or from toolbar.

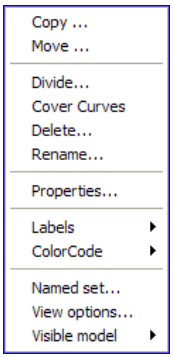


4.5 Object type Support Point

Note that Move, Labels, Named Set, Visible Model are not available from pulldown or toolbar menus.

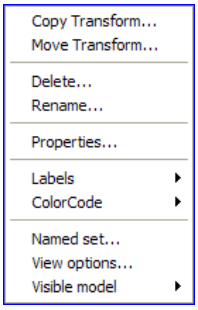
4.6 Object type Support Curve

Note that Move, Labels, Named Set, Visible Model are not available from pulldown or toolbar menus.



4.7 Object type Explicit Load

Note that Move, Named Set, Visible Model are not available from pulldown or toolbar menus

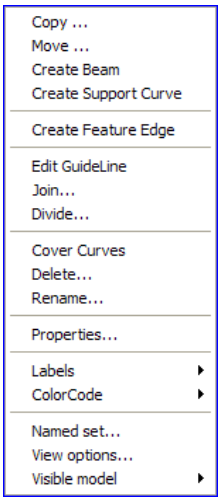


4.8 Object type Guide Plane

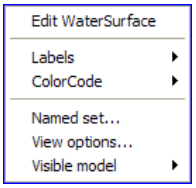
Note that Move, Named Set, Visible Model are not available from pulldown or toolbar menus.

4.9 Object type Guide Line

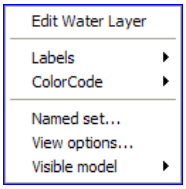
Note that Move, Create beam, Create Feature Edge, Create Support Curve, Join Curves, Cover Curves, Labels, Named Set, Visible Model are not available from pulldown or toolbar menus



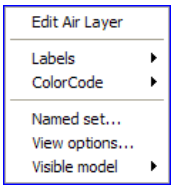
4.10 Object type Water Surface



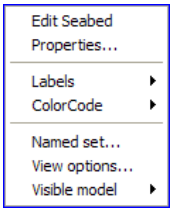
4.11 Object type Water Layer



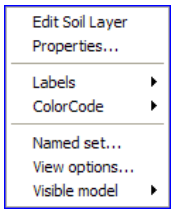
4.12 Object type Air Layer



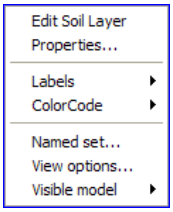
4.13 Object type Seabed



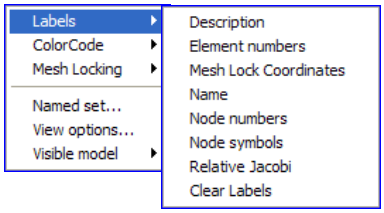
4.14 Object type Soil Layer

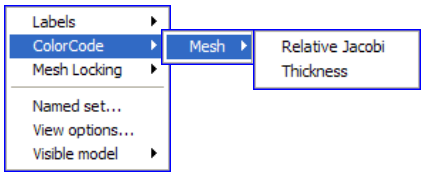


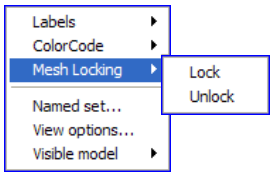
4.15 Object type Soil Border



4.16 Object type Mesh

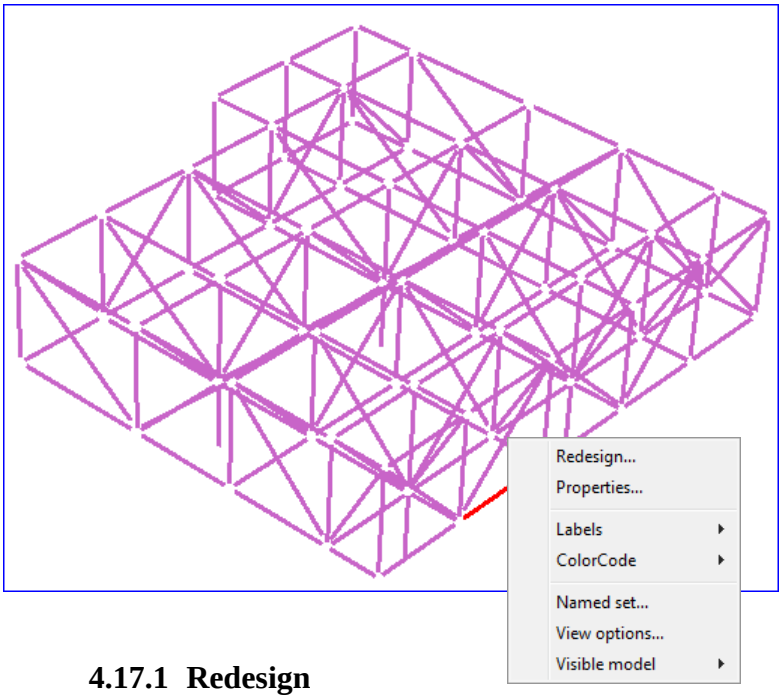




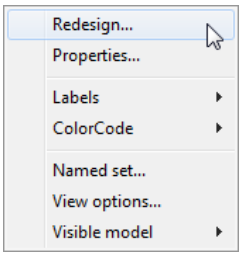


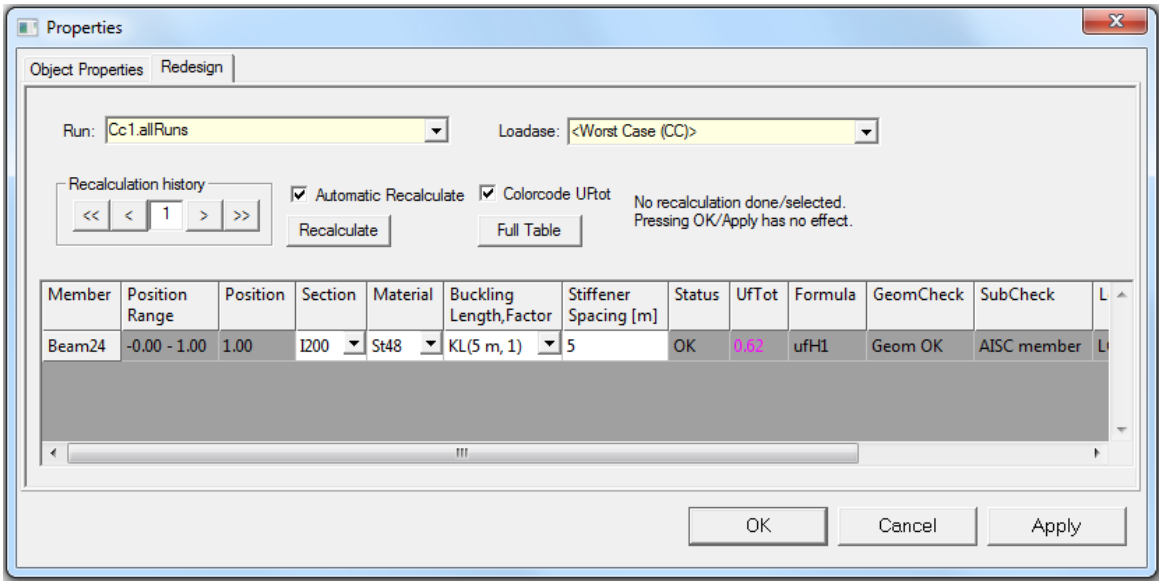
4.17 Object Type Beam Capacity Member

Examples on the context sensitive menu for capacity members are shown in the following:

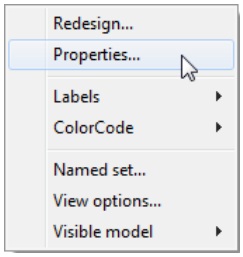


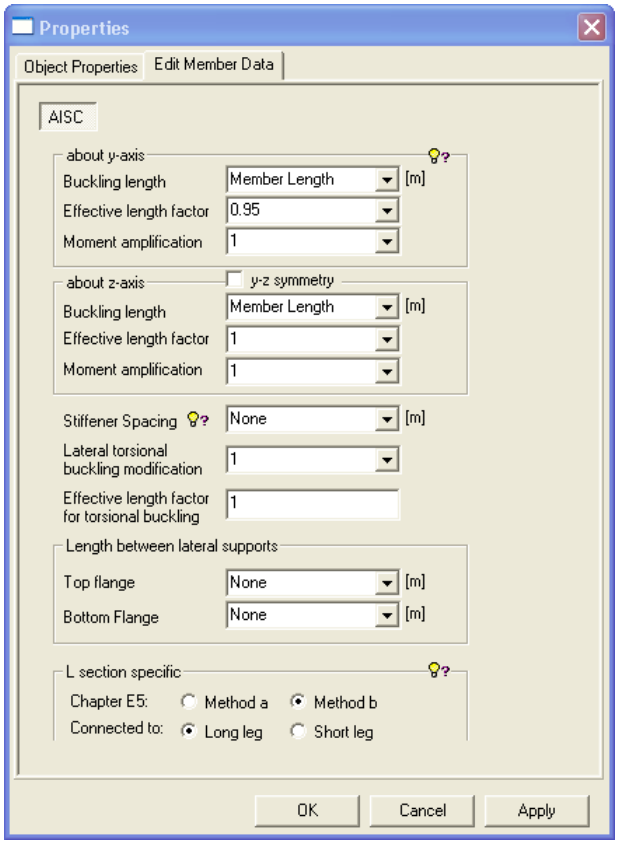
4.17.1 Redesign



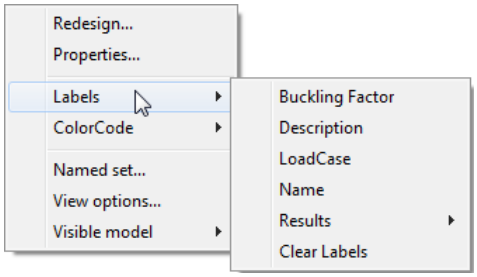


4.17.2 Capacity Member Properties

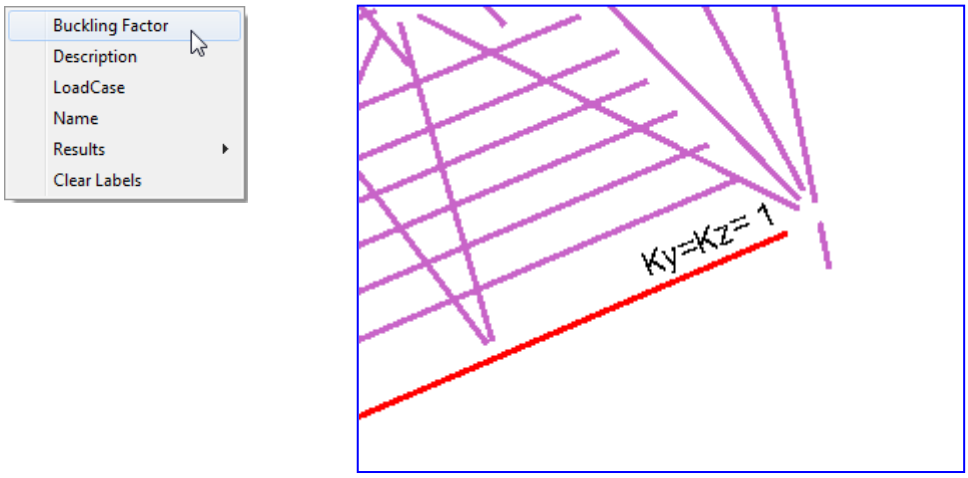




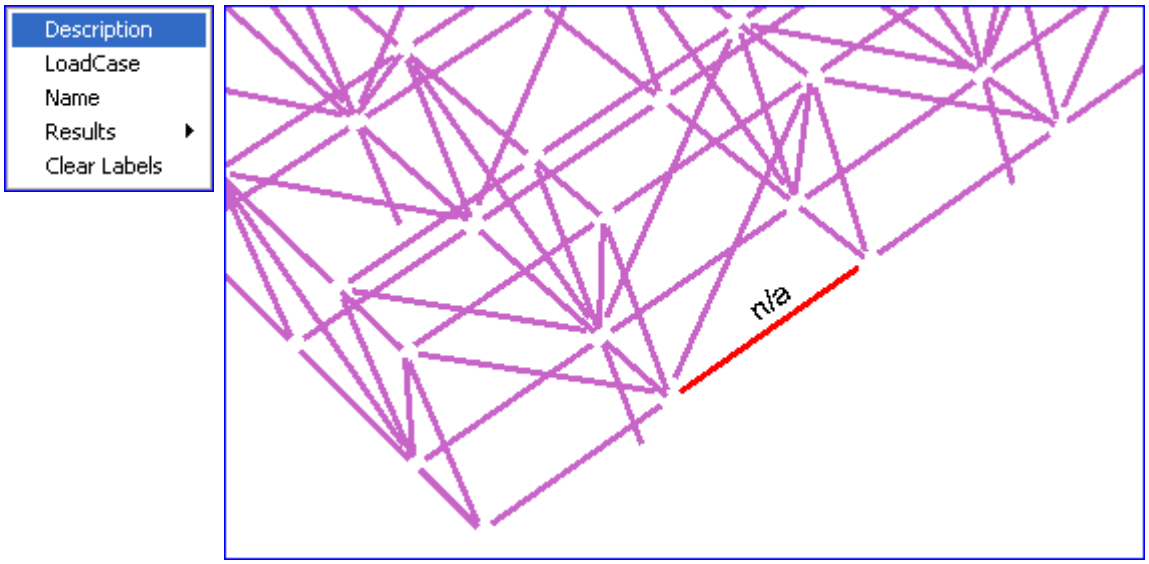
4.17.3 Capacity Member Labels



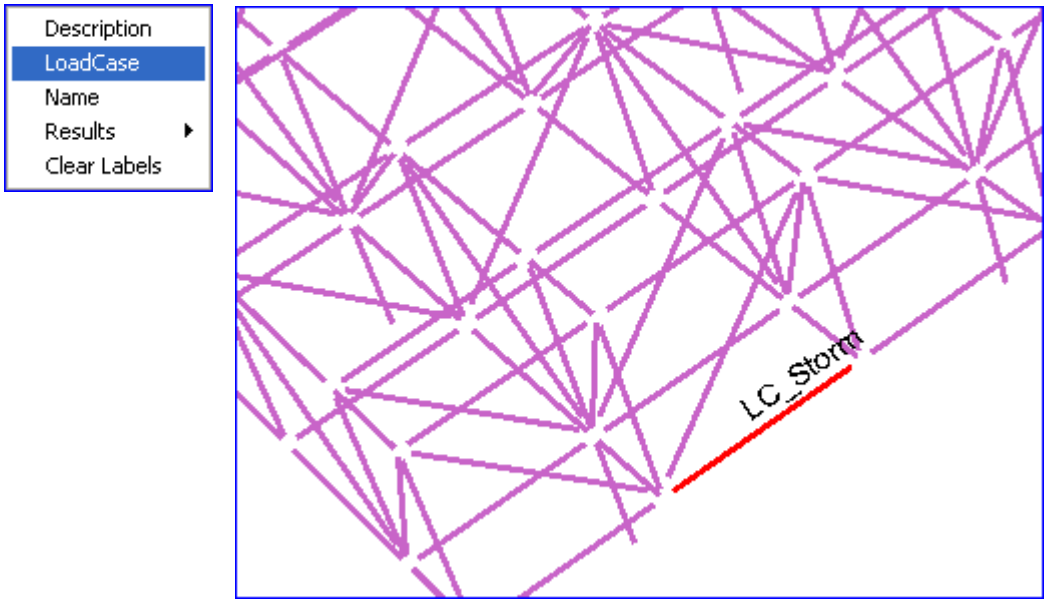
4.17.3.1 Buckling Factor



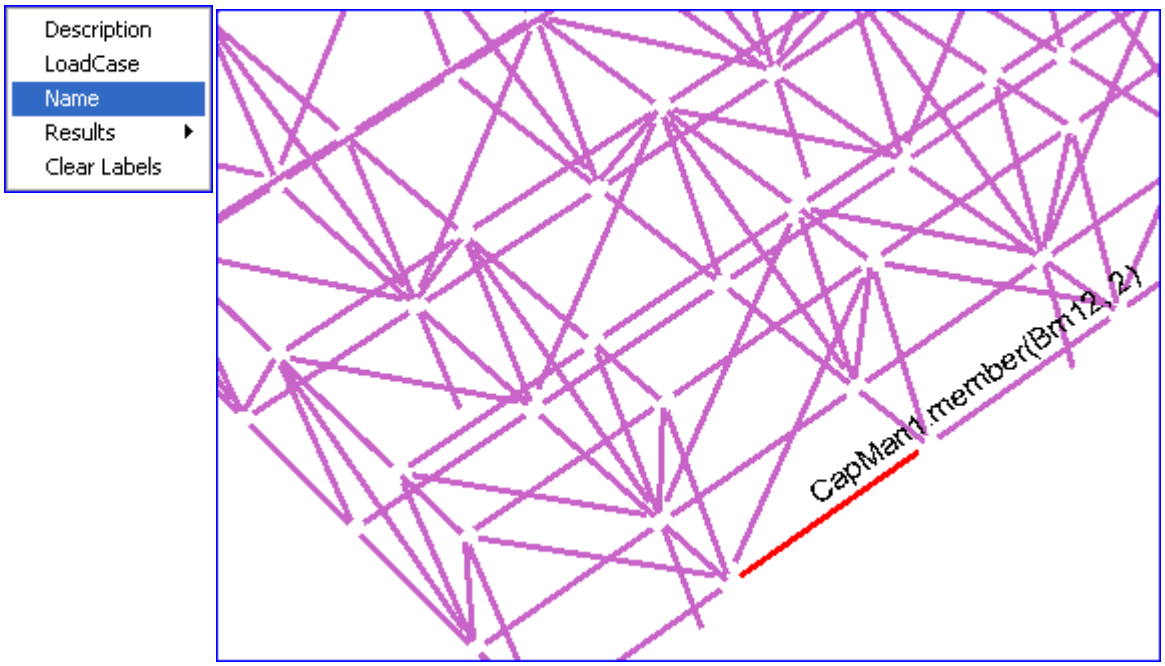
4.17.3.2 Description



4.17.3.3 LoadCase

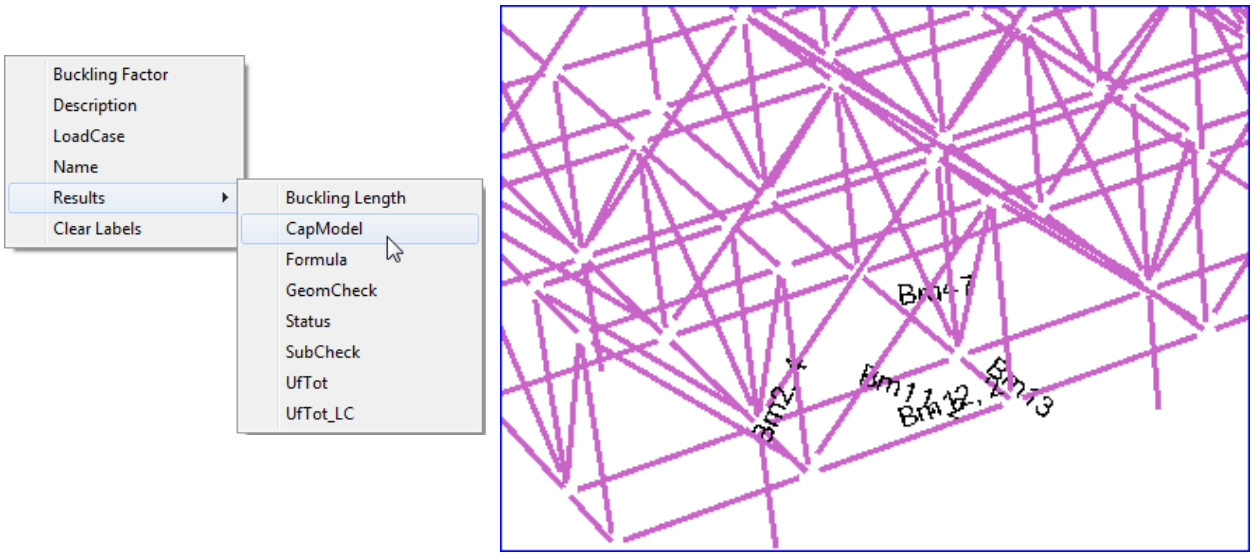


4.17.3.4 Name



4.17.3.5 Results

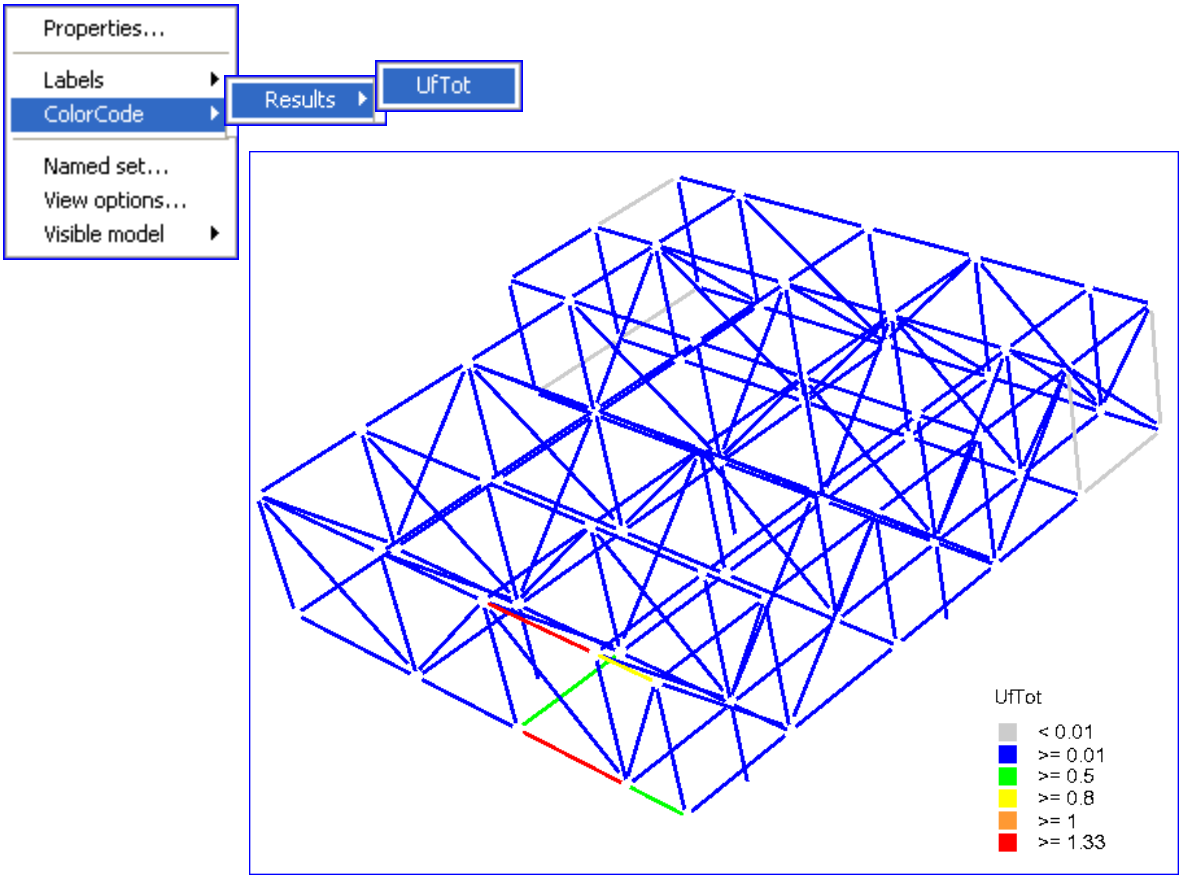
As an example we will take CapModel to show



4.17.3.6 Clear Labels

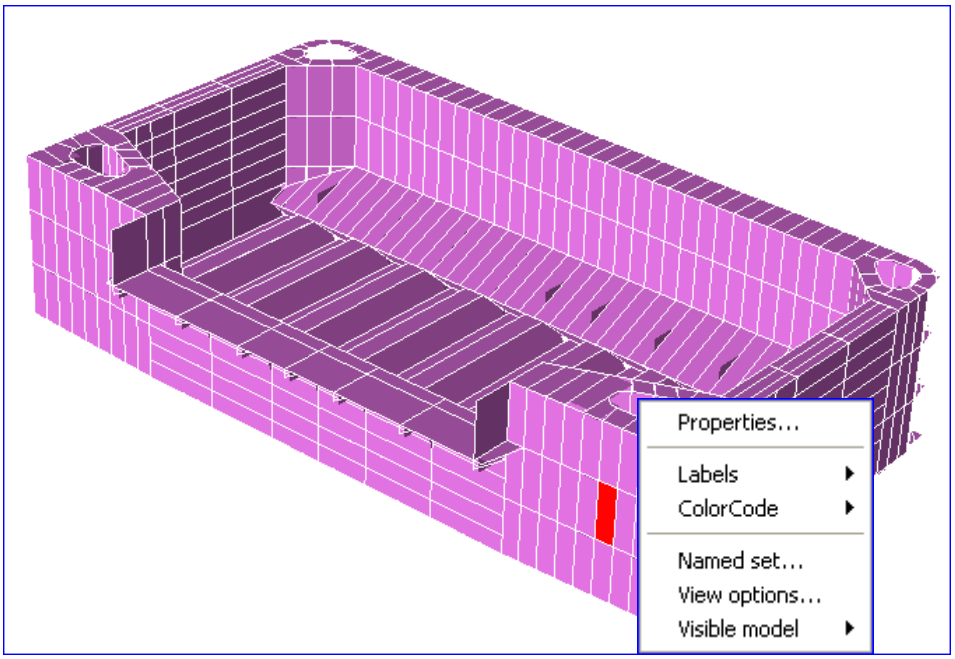
This clears all labels from the graphical display. It gives the same result as clicking the exclamation mark button. 

4.17.4 Capacity Member Color Code

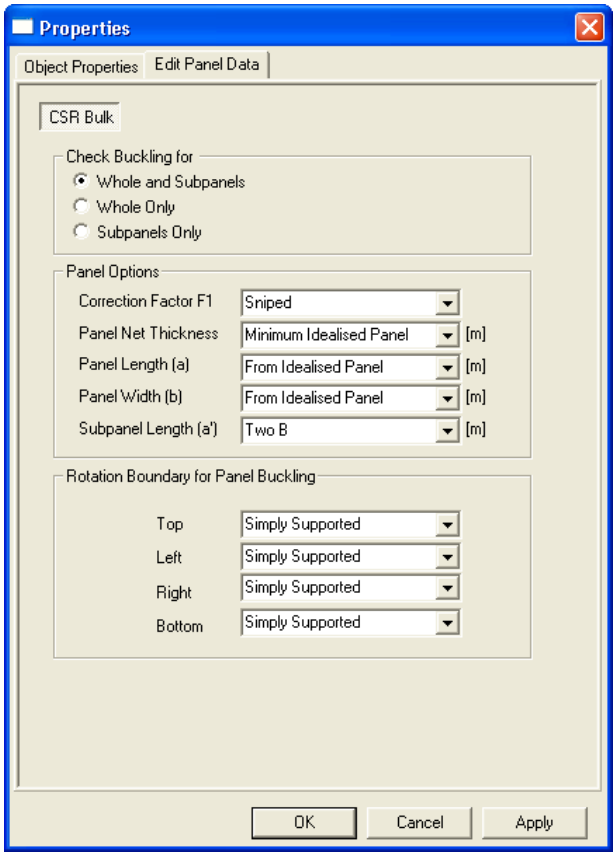


4.18 Object Type Plate Capacity Panel

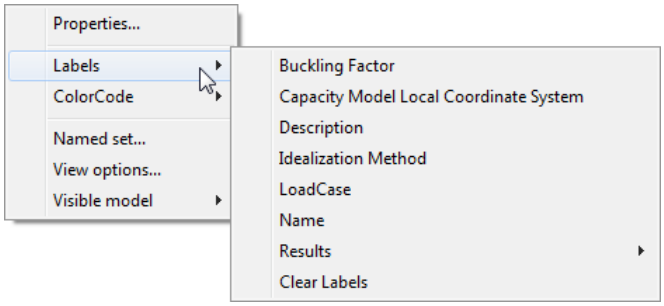
Examples on the context sensitive menu for capacity members are shown in the following:



4.18.1 Capacity Panel Properties

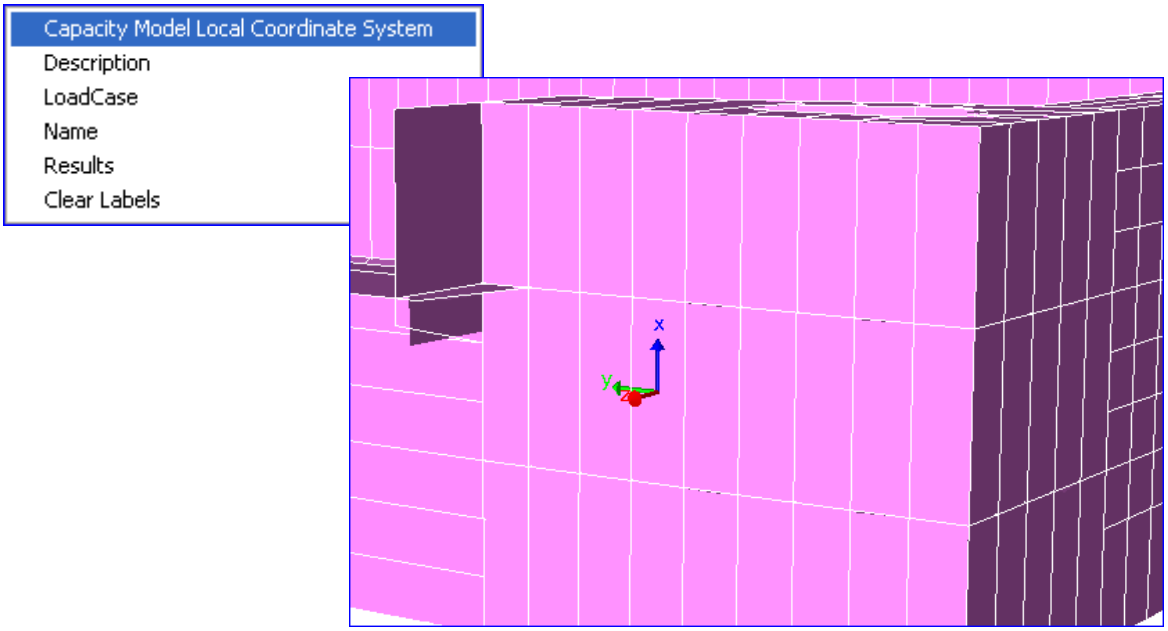


4.18.2 Capacity Panel Labels

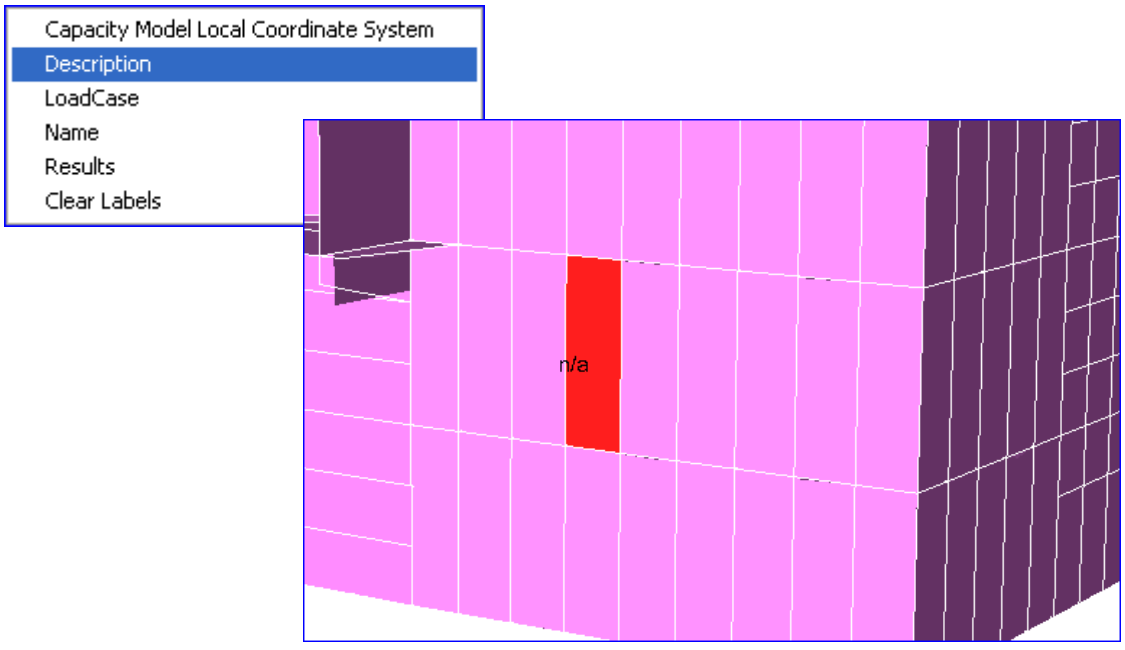


Note: Buckling Factor only applies to beam members.

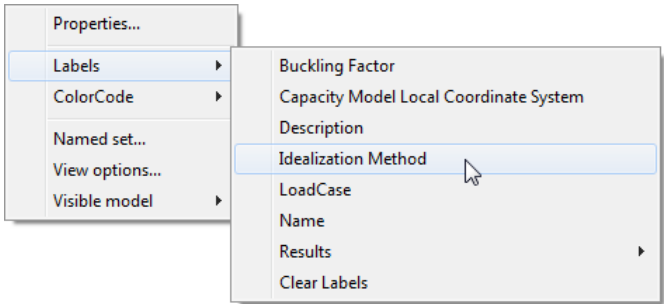
4.18.2.1 Capacity Model Local Coordinate System

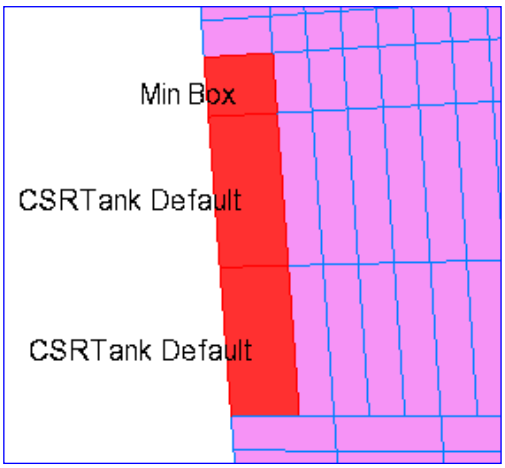


4.18.2.2 Description

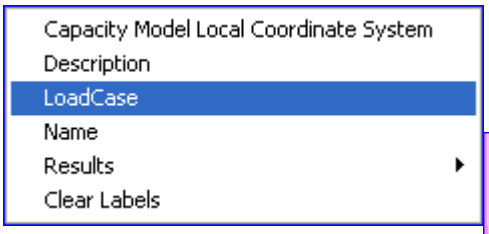


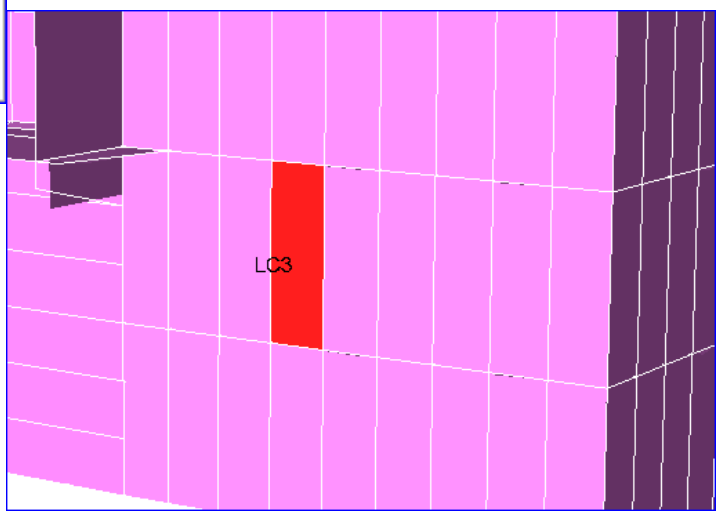
4.18.2.3 Idealization Method



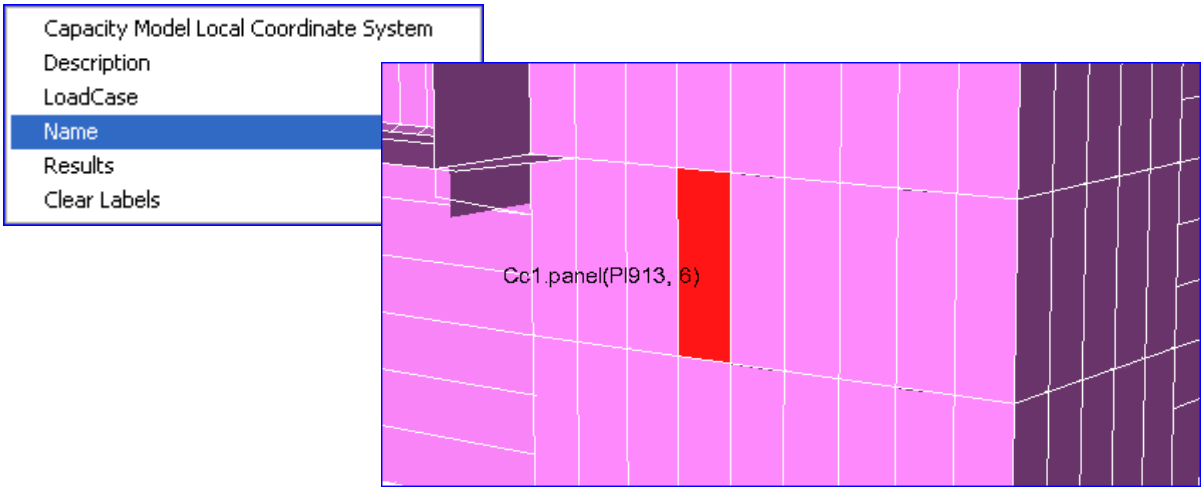


4.18.2.4 LoadCase



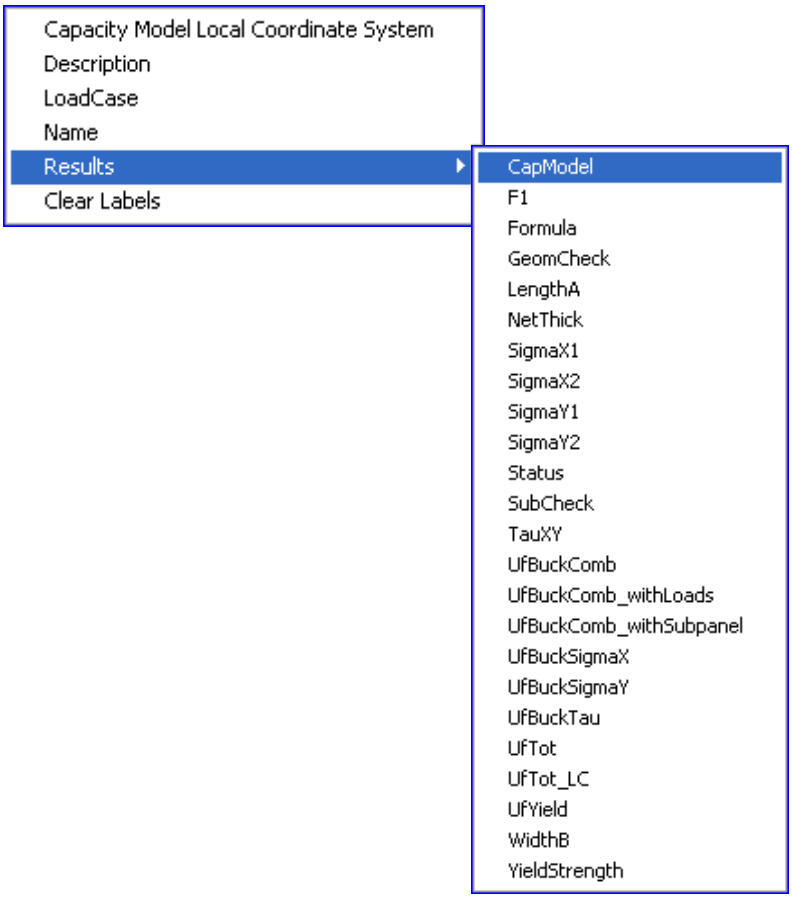


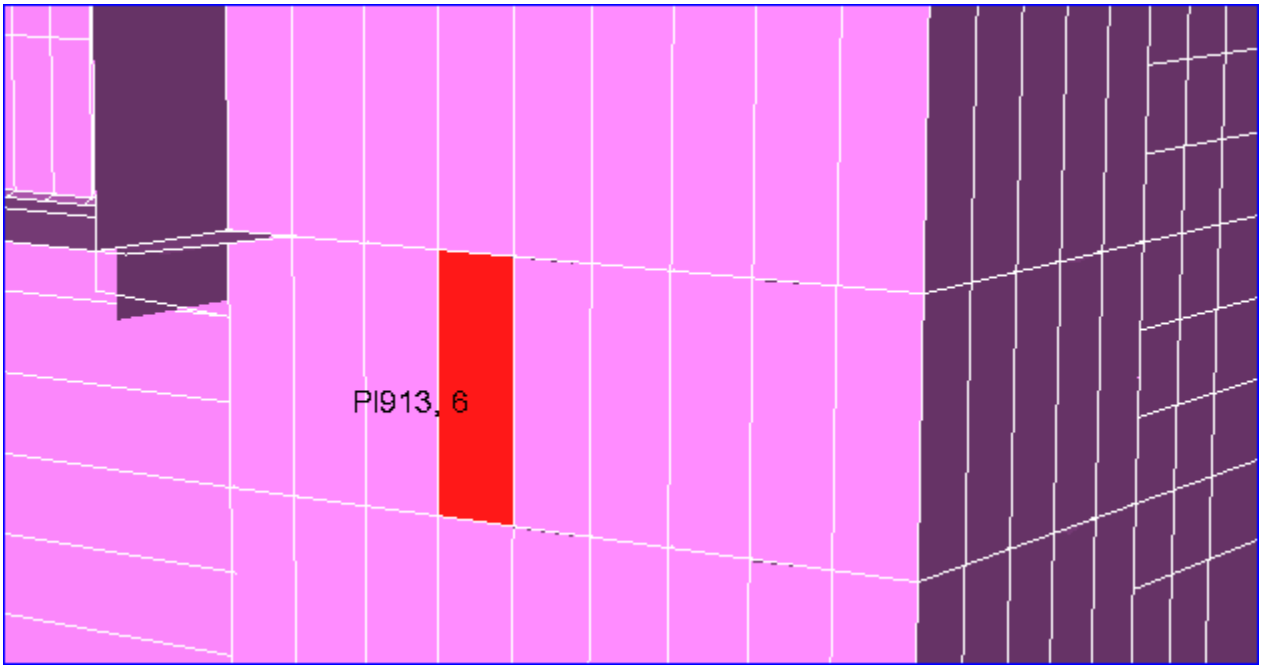
4.18.2.5 Name



4.18.2.6 Results

As an example CapModel is used to show how details can be labelled.

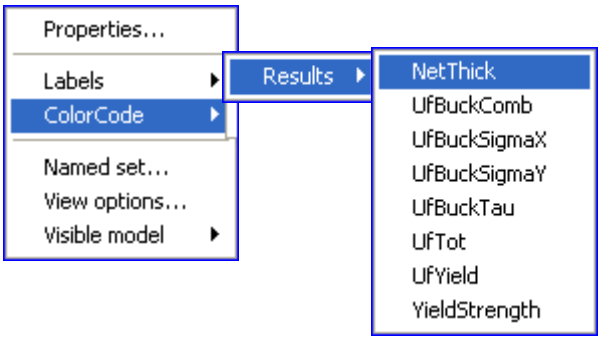


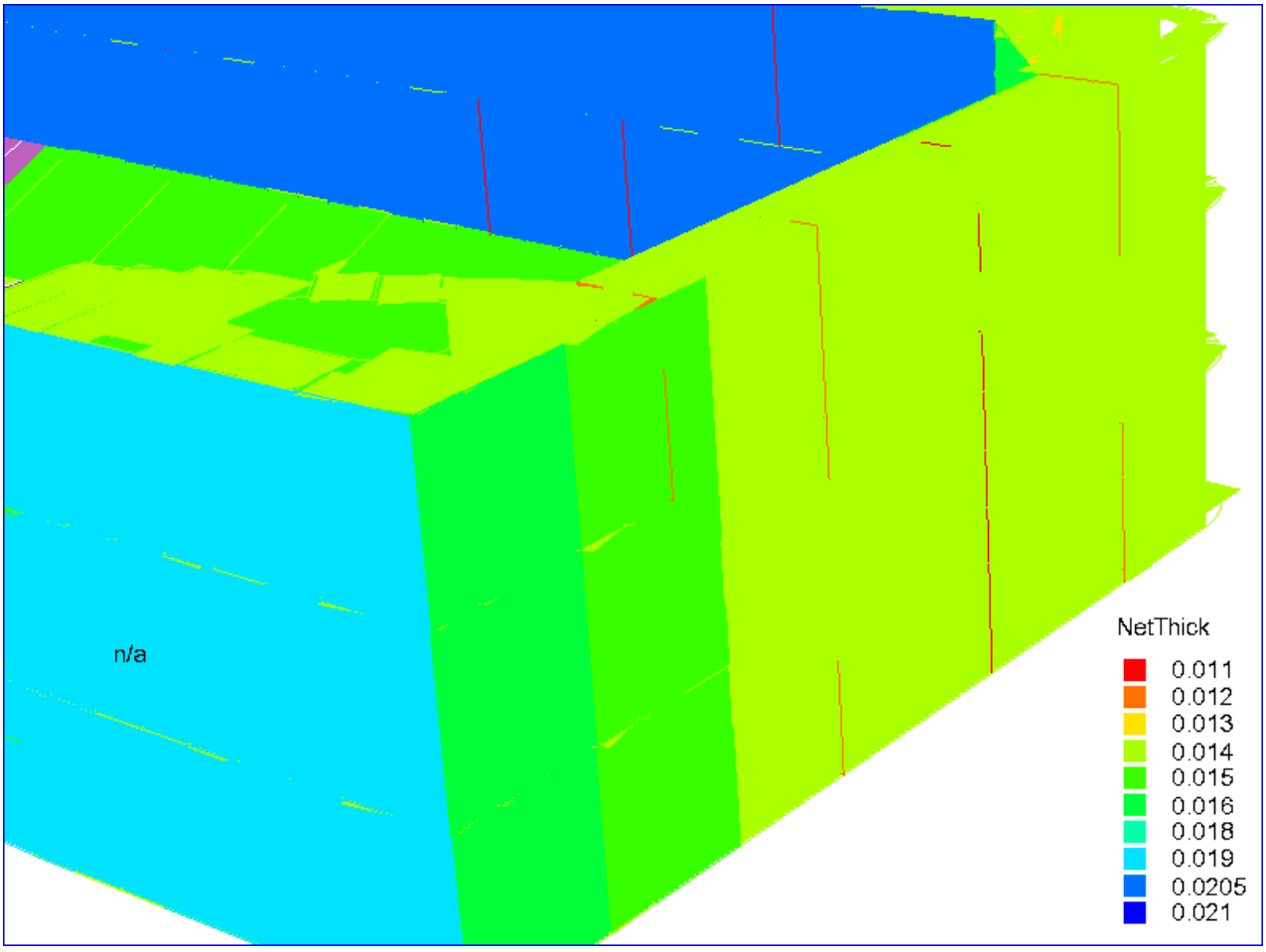


4.18.2.7 Clear Labels

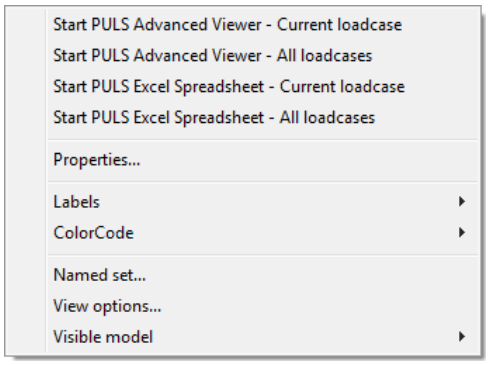
4.18.3 Capacity Panel ColorCode

The result for NetThick is used as an example to show how colour coding works.

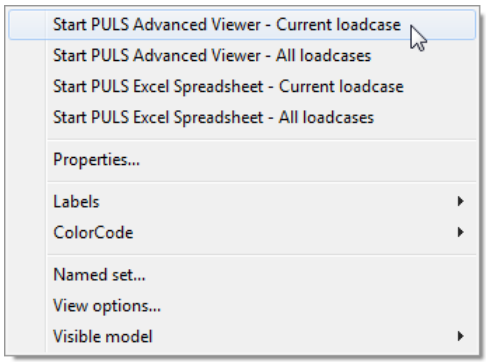


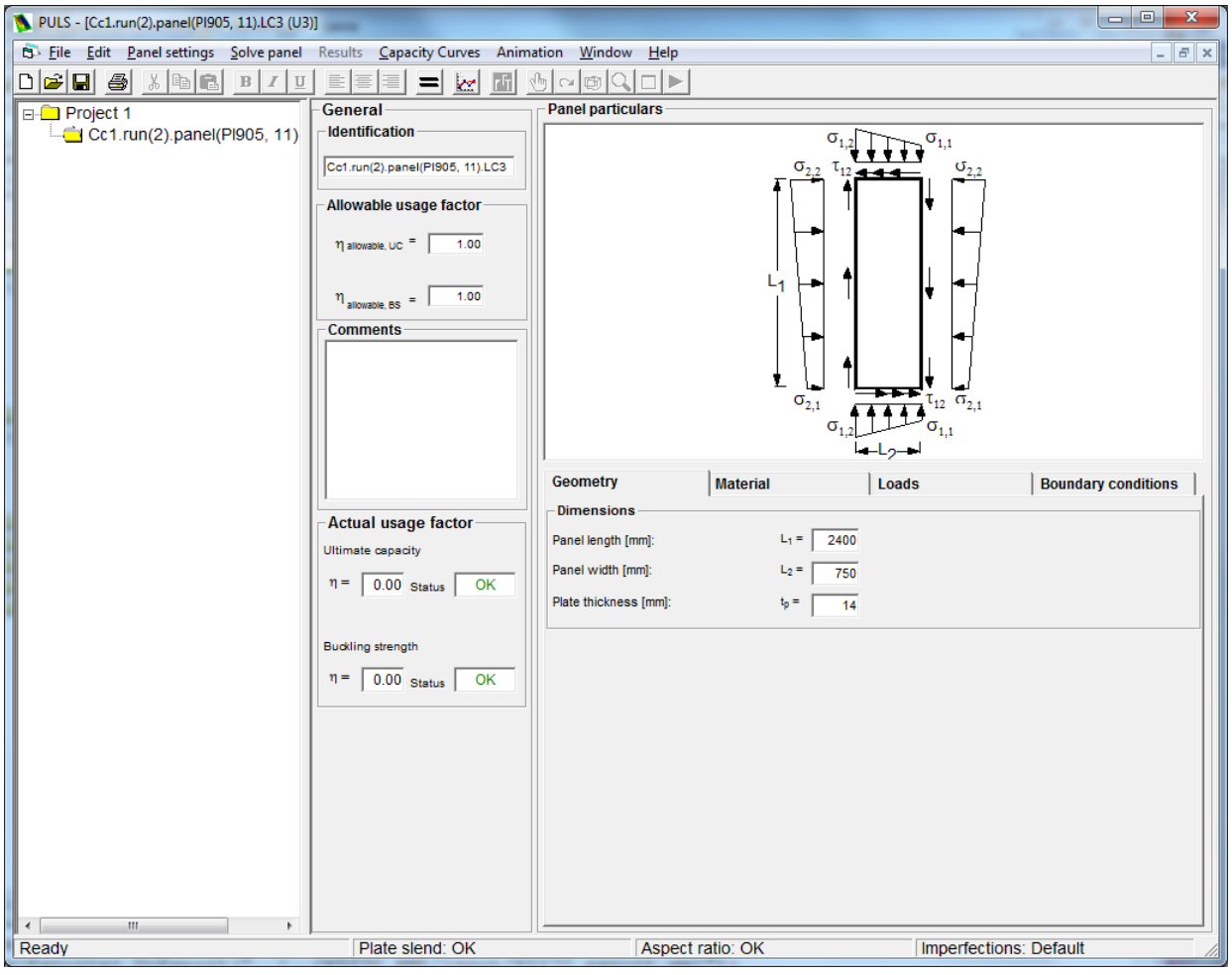


4.18.4 Capacity Panel – CSR Tank Specific

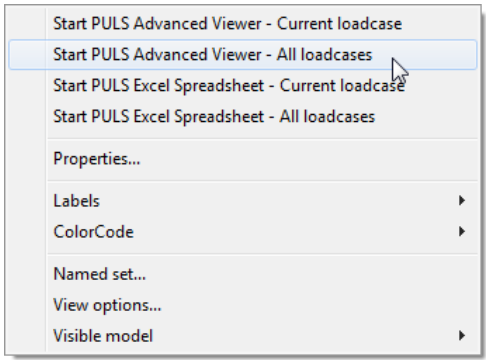


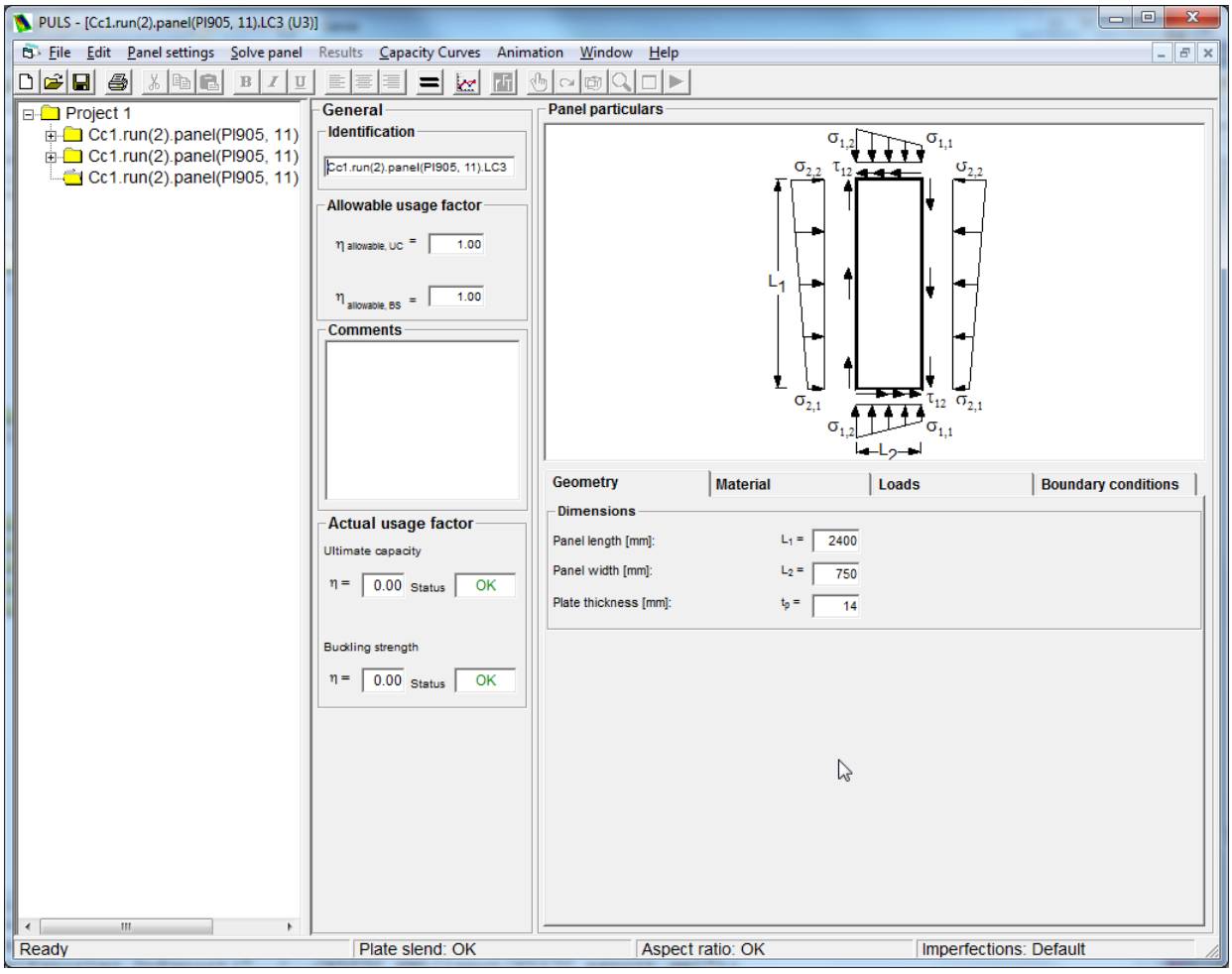
4.18.4.1 Start PULS Advanced Viewer – Current loadcase



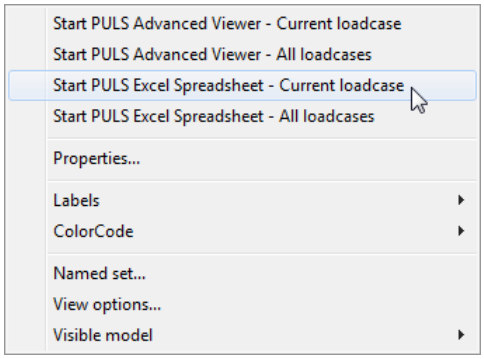


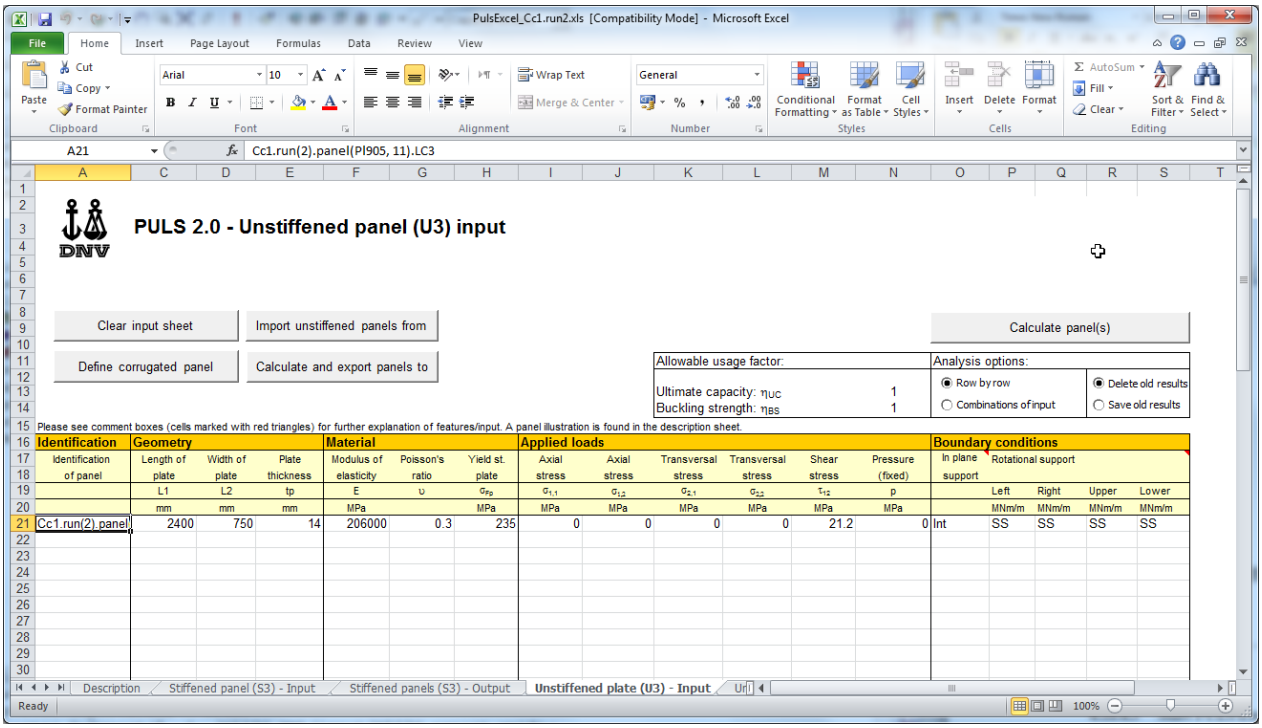
4.18.4.2 Start PULS Advanced Viewer – All loadcases



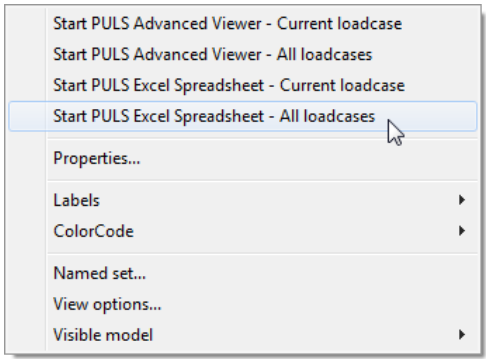


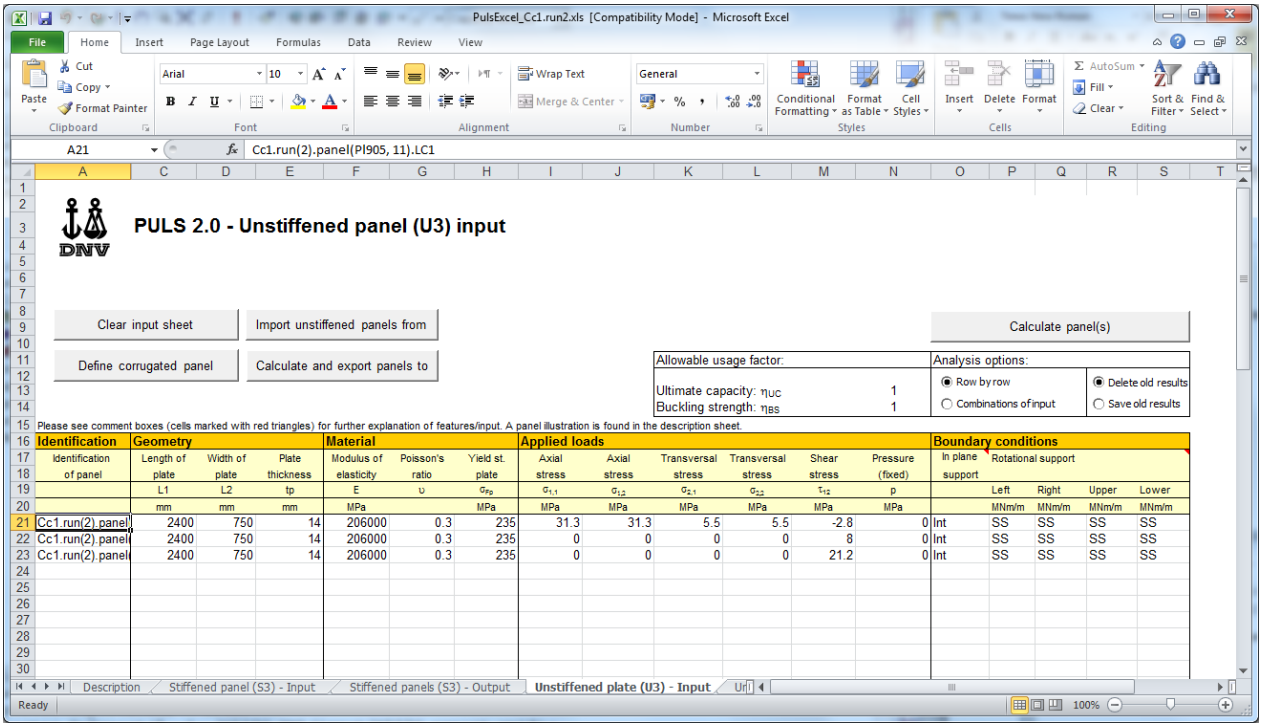
4.18.4.3 Start PULS Excel Spreadsheet – Current loadcase





4.18.4.4 Start PULS Excel Spreadsheet – All loadcases





5. SHORT COMMANDS AND WINDOWS COMPLIANCE

GeniE comes with a number of short commands utilising e.g. ALT, CTRL, and keys F1-F10 like in other Windows applications. Below is a table listing the available ones:

SHORT COMMAND	2nd LEVEL PULLDOWN	3rd LEVEL PULLDOWN
CTRL+N	**File|New Workspace**	
CTRL+O	**File|Open Workspace**	
CTRL+S	**File|Save Workspace**	
CTRL+T	**Edit|Copy**	
ALT+O	**View|Options**	
ALT+C	**Tools|Customize**	
ALT+M		**Tools|Analysis|Create mesh**
ALT+D		**Tools|Analysis|Activity Monitor**
ALT+P		**Tools|Analysis|Presentation**
ALT+G		**Tools|Analysis|Beam Result Diagrams**
ALT+S		**Visible Model|Show selection only**
ALT+Plus		**Visible Model|Add selection**
ALT+Minus		**Visible Model|Remove selection**
ALT+A		**Visible Model|Show all**
ALT+Q		**Visible Model|Show complement**
ALT+B		**View|Options|General:Cur. Background**
Del	**Edit|Delete**	

KEY	TOGGLE COMMAND
F1	**Help**
F2	**Rotate - rotation in all 3 degrees of freedom**
F3	**Zoom - zoom in or zoom out**
F4	**Pan - move model to desired position on display**
F5	**View Iso - view from isometric point**
F6	**View from X - view in negative X-direction**
F7	**View from Y - view in negative Y-direction**
F8	**View from Z - view in negative Z-direction**
F9	**Fit screen**
F10	**Spin - remembers the last rotation and speed of it and makes this a continuous spin**
F11	**Toggles the snap perpendicular, tangential, plane mode**

6. THE COMMAND LINE INTERFACE SYSTEM

GeniE is primarily intended to be operated by the graphical user interface. All program features may, however, also be accessed by GeniE commands.

The GeniE commands are basically used to create journals during interactive sessions. The journal file can now be used to re-create the model (you may also edit and change the journal file). The GeniE commands may also be used to write a model input file directly or to invoke features that have no graphical interface.

There are 3 ways of entering commands into GeniE,

1. by typing or pasting commands into the Command Line tab in GeniE
2. by the "Read command file" option in the File menu of GeniE
3. by starting GeniE with an input command file from the command prompt (DOS window)

E.g. "C:\Program Files\DNVS\GeniE\Program\GeniEr" MyProject /NEW /COM=MY_JOURNAL.JS /EXIT

Please note that if you are using another editor than e.g. MS Notepad, you need to specify that the output format is for PC format and not Unix format.

GeniE supports two kinds of commands

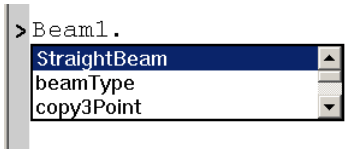
- Specific GeniE commands
- General JScript commands

Specific GeniE commands are typically for creating and editing GeniE model entities such as plates and beams. A simple session of GeniE commands may be:

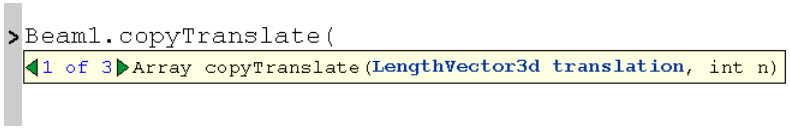
```
// Create 2 points
Point1 = Point(0,0,0);
Point2 = Point(0,0,10);
// A double slash indicates a comment line
// Copy the two points 10 m in x-direction
Point3 = Point1.copyTranslate(Vector3d(10 m,0 m,0 m));
Point4 = Point2.copyTranslate(Vector3d(10 m,0 m,0 m));
// Note that all command must end with ();
// Create a beam between point 3 and 4
BeamA1 = Beam(Point3, Point4);
BeamA1.material = Material1;
BeamA1.section = Section1;
//
// Create a support at Point3, rotate the local Z-axis 30 deg. and define the
boundary conditions
Sp1 = SupportPoint(Point(10 m,0 m,0 m));
Sp1.rotateLocalZ(30);
Sp1.boundary = BoundaryCondition(Free, Fixed, Fixed, Free, Fixed, Free);
```

Note that the objects in the GeniE model have properties and functions that are addressed by a dot like in "BeamA1.material" or "Point1.copyTranslate".

When using the GeniE Command Line tab you will get a list of relevant properties and functions by pressing the Tab key while typing the command;



The command window will also provide a tool tip to assist you with the command syntax;



GeniE also supports the general programming language JScript. By combining GeniE commands with JScript you may use programming features in your model input file.

Typically you may start by defining all basic model data as variables in the beginning of the file and then refer to these variables when creating the model.

```
//  Coordinate arrays
var X1 = new Array();
var Y1 = new Array();
var Z1 = new Array();
//   X1-values
X1[0] = -23;
X1[1] = 12;
X1[2] = 34;
//   Y1-values
Y1[0] = -45;
Y1[1] = -20;
Y1[2] = -5;
//   Z1  elevations
Z1[0] = -128.987;
Z1[1] = -124.987;
//
Point_A = Point(X1[0], Y1[0], Z1[0]);
```

Further you may use arithmetic expressions within the GeniE commands or to do separate calculations;

```
TopElevation = Z1[1] + 2.45 ;
Point_B = Point( 2.54 , 0.0, (TopElevation + 10)) ;
```

Please note that most mathematical functions are addressed through the Math object in JScript.

```
Radius = 5.0;
MyArea = Math.PI()*Math.pow(Radius,2);
Print(MyArea);
 78.539816
```

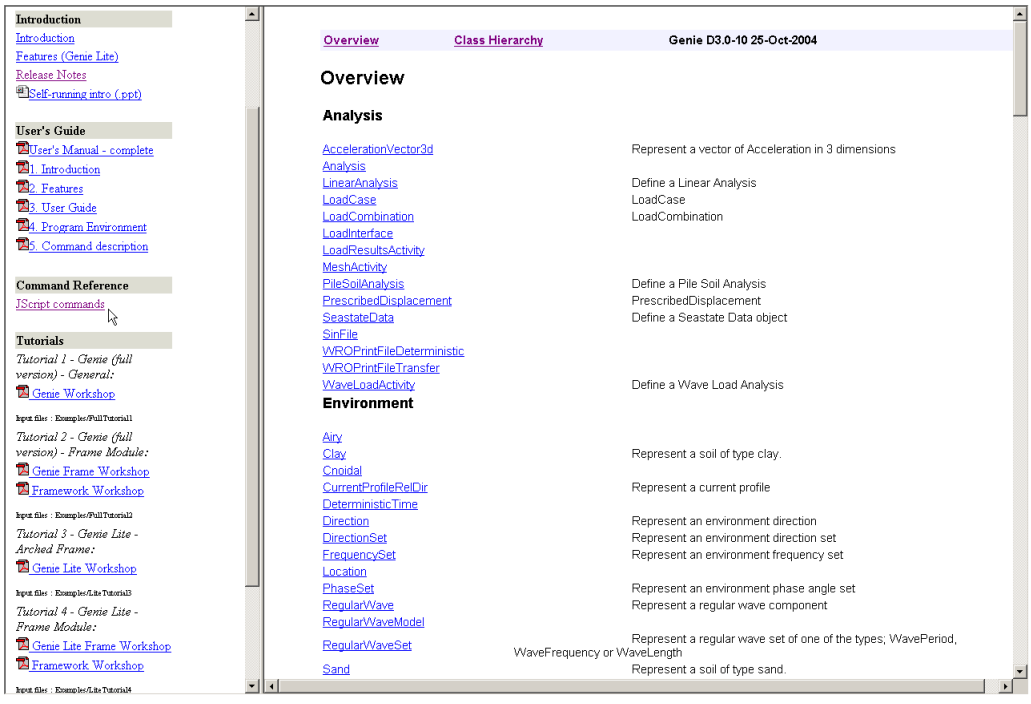
For more sophisticated modelling you may use For loops and If testes to program the creation of your model. You may also invoke other applications that support Automation (e.g. Excel) to exchange data with your GeniE model. Automation is a technology that allows software packages to expose their features to scripting tools and other applications.

Example of creating beams in a loop:

```
//Adding beams to a leg
for (k = 0; k < 5 ; k++)
{
var Bms1 = new Array();
Bms1[k] = Beam(LegPoint[k], LegPoint[k+1]);
Bms1[k].name = "LegBeam" + k ;
Bms1[k].section = LegSection;
Bms1[k].material = LegMaterial;
}
```

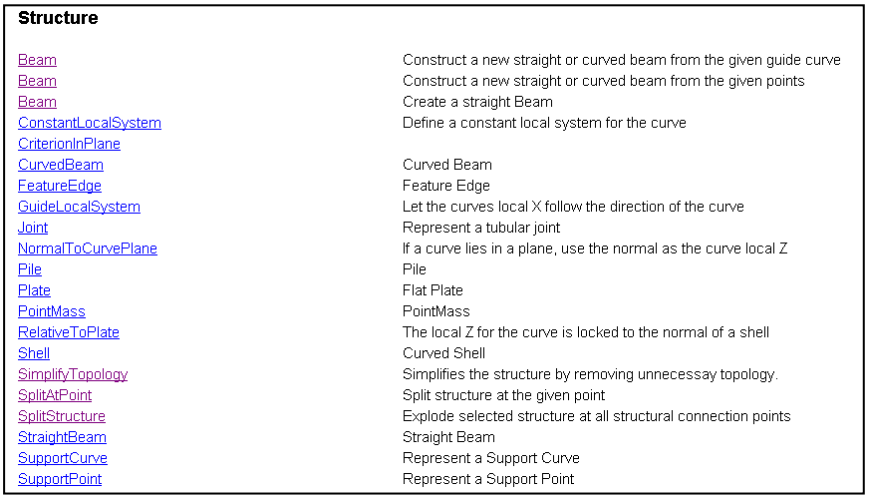
6.1 The GeniE JSript Command Reference

The GeniE commands are described in the JScript Command Reference found in the GeniE help menu. The most feasible approach to writing a command file is rather to create a sample model interactively and then use the generated journal as a template. When you have found the kind of object you are working with you may use the Command Reference to get a full list of relevant features and the command syntax for accessing these. The following pictures show how you find information about the script command for generating a basic beam and which other commands are associated this operation.



The documentation is organised in classes:

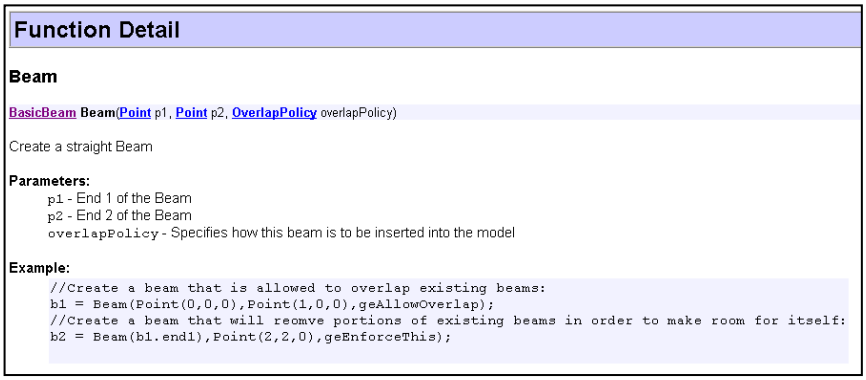
- Analysis
- Environment
- Exchange
- Guiding geometry
- Loads
- Properties
- Sets
- Structure
- Units
- Other



Structure

Beam	Construct a new straight or curved beam from the given guide curve
Beam	Construct a new straight or curved beam from the given points
Beam	Create a straight Beam
ConstantLocalSystem	Define a constant local system for the curve
CriterionInPlane	
CurvedBeam	Curved Beam
FeatureEdge	Feature Edge
GuideLocalSystem	Let the curves local X follow the direction of the curve
Joint	Represent a tubular joint
NormalToCurvePlane	If a curve lies in a plane, use the normal as the curve local Z
Pile	Pile
Plate	Flat Plate
PointMass	PointMass
RelativeToPlate	The local Z for the curve is locked to the normal of a shell
Shell	Curved Shell
SimplifyTopology	Simplifies the structure by removing unnecessay topology.
SplitAtPoint	Split structure at the given point
SplitStructure	Explode selected structure at all structural connection points
StraightBeam	Straight Beam
SupportCurve	Represent a Support Curve
SupportPoint	Represent a Support Point

Looking up straight beam found under the class Structure.



Function Detail

Beam

BasicBeam **Beam**(Point p1, Point p2, OverlapPolicy overlapPolicy)

Create a straight Beam

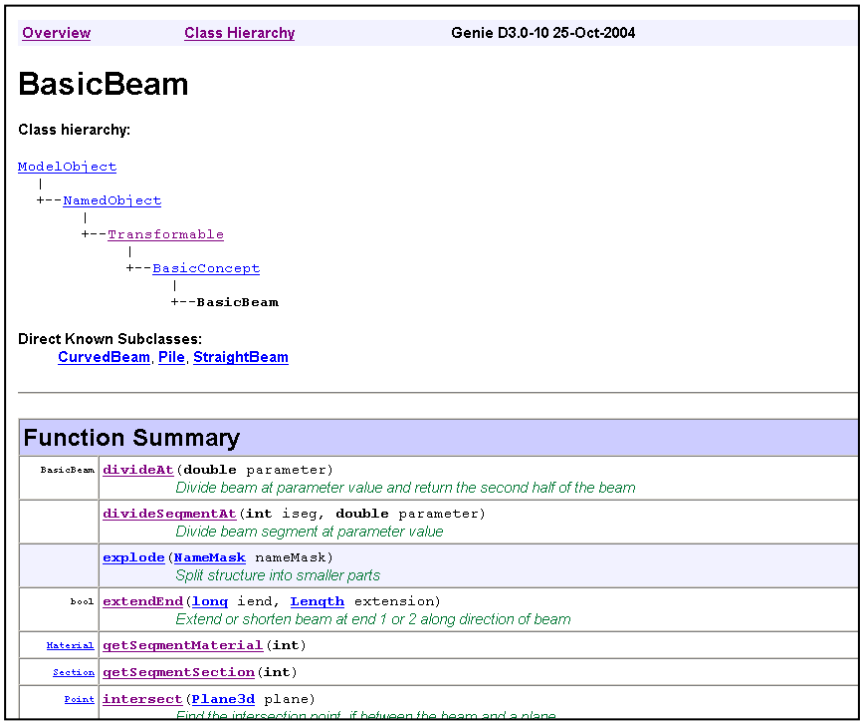
Parameters:

`p1` - End 1 of the Beam
`p2` - End 2 of the Beam
`overlapPolicy` - Specifies how this beam is to be inserted into the model

Example:

```
//Create a beam that is allowed to overlap existing beams:
b1 = Beam(Point(0,0,0),Point(1,0,0),geAllowOverlap);
//Create a beam that will reomve portions of existing beams in order to make room for itself:
b2 = Beam(b1.end1),Point(2,2,0),geEnforceThis);
```

The *Function Detail* now documents the command itself and practical examples.



Overview Class Hierarchy **Genie D3.0-10 25-Oct-2004**

BasicBeam

Class hierarchy:

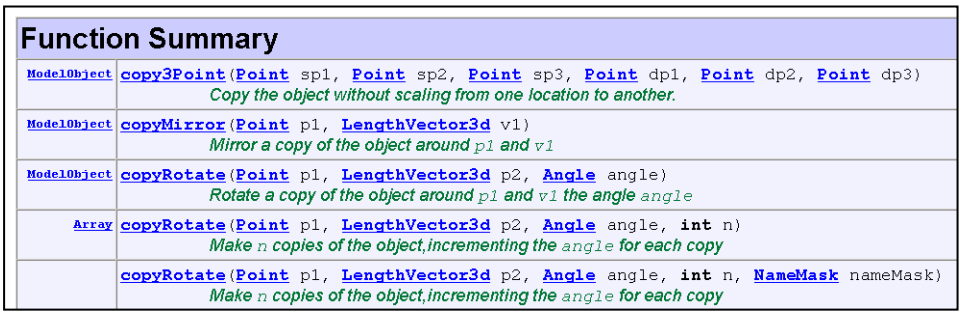
```
ModelObject
  |
  +--NamedObject
       |
       +--Transformable
            |
            +--BasicConcept
                 |
                 +--BasicBeam
```

Direct Known Subclasses:
CurvedBeam, Pile, StraightBeam

Function Summary

BasicBeam	**divideAt**(**double** parameter) *Divide beam at parameter value and return the second half of the beam*
	divideSegmentAt(**int** iseg, **double** parameter) *Divide beam segment at parameter value*
	explode(**NameMask** nameMask) *Split structure into smaller parts*
bool	**extendEnd**(**long** iend, **Length** extension) *Extend or shorten beam at end 1 or 2 along direction of beam*
Material	**getSegmentMaterial**(**int**)
Section	**getSegmentSection**(**int**)
Point	**intersect**(**Plane3d** plane) *Find the intersection point, if between the beam and a plane*

You may look at further details belonging to the BasicBeam. When scrolling down the Function Summary the additional commands (ore features) are listed.



Function Summary

ModelObject	**copy3Point**(**Point** sp1, **Point** sp2, **Point** sp3, **Point** dp1, **Point** dp2, **Point** dp3) *Copy the object without scaling from one location to another.*
ModelObject	**copyMirror**(**Point** p1, **LengthVector3d** v1) *Mirror a copy of the object around `p1` and `v1`*
ModelObject	**copyRotate**(**Point** p1, **LengthVector3d** p2, **Angle** angle) *Rotate a copy of the object around `p1` and `v1` the angle `angle`*
Array	**copyRotate**(**Point** p1, **LengthVector3d** p2, **Angle** angle, **int** n) *Make `n` copies of the object,incrementing the `angle` for each copy*
	copyRotate(**Point** p1, **LengthVector3d** p2, **Angle** angle, **int** n, **NameMask** nameMask) *Make `n` copies of the object,incrementing the `angle` for each copy*

Scroll down to *copyRotate* to find out how this command works

By clicking the link “copyRotate” you will get a detailed description of this function as follows:

ModelObject copyRotate(**Point** p1, **LengthVector3d** p2, **Angle** angle)

Rotate a copy of the object around `p1` and `v1` the angle `angle`

Decription:

Make one copy of the object. The transformation is defined as counterclockwise rotation around the given axis vector at the anchor point.

Parameters:

`p1` - Point on rotation axis

`p2` - Rotation axis vector

`angle` - Rotation angle

Returns:

the copied object

Example:

```
//Rotate Bm1 45 degrees around Bm1.end1 and the axis (0,0,1):
Bm2=Bm1.copyRotate(Bm1.end1,Vector3d(0,0,1),45deg);
```

The description shows the syntax of the command with the type and name of each parameter. By clicking the link for the type like **LengthVector3d** you will get all valid forms for entering a vector in GeniE.

More about general JScript commands may be found e.g. at http://msdn.microsoft.com/scripting.

6.2 Useful script commands

GeniE creates a journal file including all the operations you perform except a few – typically related to making a finite element model (when not part of analysis activity), export the FE model or graphical interactions. These commands may be executed directly from the Command Line Interface or when importing a journal file containing the commands. You may find some of the following script commands useful.

Command	**Description**
CreateMesh();	Force the creation of a mesh (same as **Alt+D** or ***Tools|Analysis|Create Mesh***)
FemExporter = ExportMeshFem(); FemExporter.DoExport(name);	Export a finite element model to the default working directory (same as ***File|Export|FEM File***), e.g. FemExporter.DoExport("T1.FEM");
GenieRules.Meshing.preference(mpUseDrillingElements,true);	Activation of 3 and 4 noded drilling elements (FTAS and FQAS)
Graphics.move(vector);	Move the view of the model along the given vector in the global system, e.g. Graphics.move(Vector3D(1,1,0));
Graphics.pan(x,y);	Pan a model. Same as the graphical operation 'Pan (F4)', e.g. Graphics.pan(1,10); will move the model 1 pixel to the right and 10 pixels up. Note that x and y are measured in pixels.
Graphics.rotate(rotationAxis,angle);	Rotate the view of the model a given angle around the centre of the model with a rotation axis given in the global system, e.g. Graphics.rotate(Vector3d(0,0,1),45); The model is now rotated 45degrees around global z-axis.
Graphics.rotate(rotationCenter,rotationAxis,angle);	Rotate the view of the model a given angle around rotationCenter with a rotation axis given in the global system, e.g. Graphics.rotate(Point(2,5,3),Vector3d(0,0,1),45); The model is now rotated 45degrees around global z-axis around the point with coordinate values (2,5,3)
Graphics.rotationCenter;	Return the models centre of rotation, i.e. when you rotate the model graphically (using the default rotation scheme, the model is rotated around the centre of the model. Graphics.rotationCenter);: will return this point.
Graphics.saveImage(name);	Save the image in a given format (given by the filename), e.g. Graphics.saveImage("Picture.jpg");

Command	**Description**
Graphics.saveImage(name,width,height);	Save the image in a given format, scaled to the size in pixels given by width and height, e.g. Graphics.saveImage("Picture.jpg",2000,2000);
Graphics.zoomArea(left,bottom,right,top);	Rubberband zoom. Same as the graphical operation 'Zoom Rubberband'. The input (left,bottom,right,top) are given in pixels. The bottom left pixelcoordinate of the 3d-view is 0,0. The top right pixelcoordinate of the 3d-view is dependent on your screen size. (You can find this value on your computer by File|Save Graphics As...|<Save> Under Pixels|Width Pixels|Height Pixels|Width is the rightmost pixel coordinate of the screen. Pixel|Height is the top coordinate of the screen. If the width and height of the screen are 600,600, the size of our 3d viewport is (0,0,600,600). If you want to zoom in on the middle portion of the screen, you may use *Graphics.zoom Area(150,150,450,450);*. If you want to zoom out, you may use *Graphics.zoomArea(-150,-150,750,750);*.
Graphics.setEye(eyePos);	The command gives a view of the model focused on its origin,e.g. Graphics.setEyePosition(Point (-30,30,5),Vector3d(1,-1,0));. To be used to recreate a specific view.
Graphics.viewISO();	Same as the graphical operation 'View ISO (F5)' Note, you need to refresh graphics to yield immediate screen update
Graphics.viewFromX();	Same as the graphical operation 'View from X (F6)' Note, you need to refresh graphics to yield immediate screen update
Graphics.viewFromY();	Same as the graphical operation 'View from Y (F7)' Note, you need to refresh graphics to yield immediate screen update
Graphics.viewFromZ();	Same as the graphical operation 'View from Z (F8)' Note, you need to refresh graphics to yield immediate screen update
Graphics.fitModel();	Same as the graphical operation 'Fit (F9)' Note, you need to refresh graphics to yield immediate screen update
JsExporter = ExportModelJS(); JsExporter.DoExport(name)	Export a js file to the default working directory (same as ***File|Export GeniE Journal File***), e.g. JsExporter.DoExport("C:/Location/yourname.js")

Command	**Description**
<platename_A>.join(<platename_B>);	Command for joining plate B into plate A. The new plate keeps name plate A. Example: Joining Pl124 with Pl121 has the command Pl121.join(Pl124);
Math.timer(long t0);	Useful feature that will return elapsed seconds from t0. You need to specify start time (t0), when to measure elapsed time and print to the journal file window. Example returning elapsed time at t1 and t2 (to be edited in the journal file): *t0=Math.timer(0);* *t1=Math.timer(t0);* *print(t1);* *-> returns time since t0 in seconds* *t2=Math.timer(t0);* *print(t2);* *->returns time since t0 in seconds*
SinExporter = ExportResultsSin(); SinExporter.DoExport(name);	Export a results file to the default working directory (same as ***File|Export SIN File***), e.g. SinExporter.DoExport("R1.SIN");
XmlExporter = ExportConceptXml(); XmlExporter.exportLoads = true/false; XmlExporter.DoExport(name);	Export a xml file to the default working directory (same as ***File|Export XML File***), e.g. XmlExporter.DoExport("Semi.xml");

By using these commands together with commands automatically created during GeniE sessions it is possible to run GeniE from batch mode to do among others

- Create structure
- Apply loads
- Model environment (wave, current, air, soil)
- Perform analysis – structural, wave and pile-soil (or export FEM model for later usage in e.g. a superelement analysis)
- Specify view settings, create and save graphics images

---------- o ----------