MANAGING RISK DNV

Sesam User Course

JScript in GeniE command input
Parametric modelling

Revised: 1 July 2009

Contents

- Applications of JScript in GeniE
- Examples on parametric models in GeniE
- A practical example - workshop
- GeniE & JScript – what you need to know
- Documentation of GeniE JScript
- For the advanced user
  - Basic JScript
  - Variables and Data Types
  - GeniE Data Types
  - Understanding the structure of GeniE commands
  - Basic JScript within GeniE commands
  - Objects
  - Accessing the objects in a model
  - Controlling Program Flow
    - if ... else
    - do ... while
    - for ...
    - for ... in
  - Functions
  - User defined functions
  - JScript objects – the Math object
  - Working with Office documents from GeniE
  - User defined JScript objects
  - Parametric modelling
  - Generating Pictures and Reports
  - Load Intensity by Jscript
  - Limitations
- Some of the content require that you have access to the file “Frame.js”

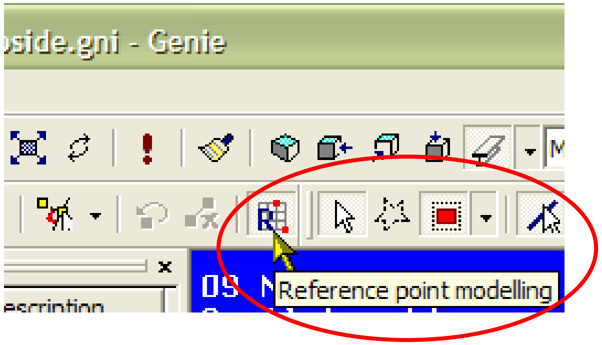
Applications of JScript in GeniE

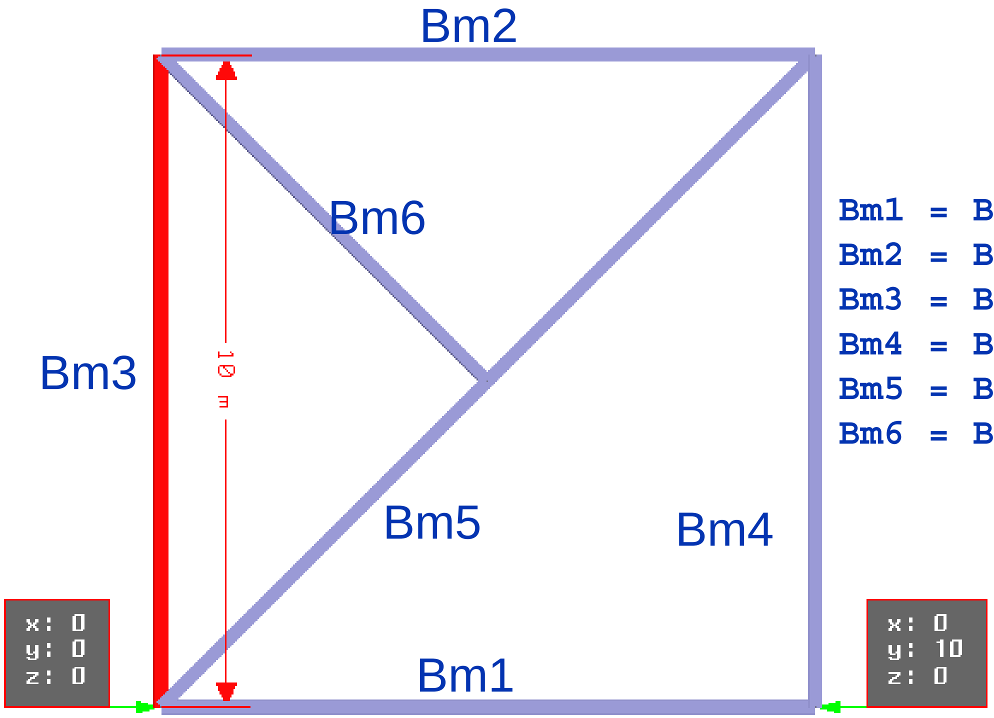
MANAGING RISK DNV

- Develop custom engineering applications
  - Script that controls GeniE and other programs
- Perform standard operations
  - Setting up a new workspace
  - Set code check parameters
  - Reporting
- Create models
  - Well organized model
  - Steady progress
  - Robust against changes
  - Parametric modeling
- Extensive model updates
  - Efficient
  - Quality control
- Distributed loads
  - Load intensity as JScript

Parametric models in GeniE

- Can be combination of variables and reference point modelling





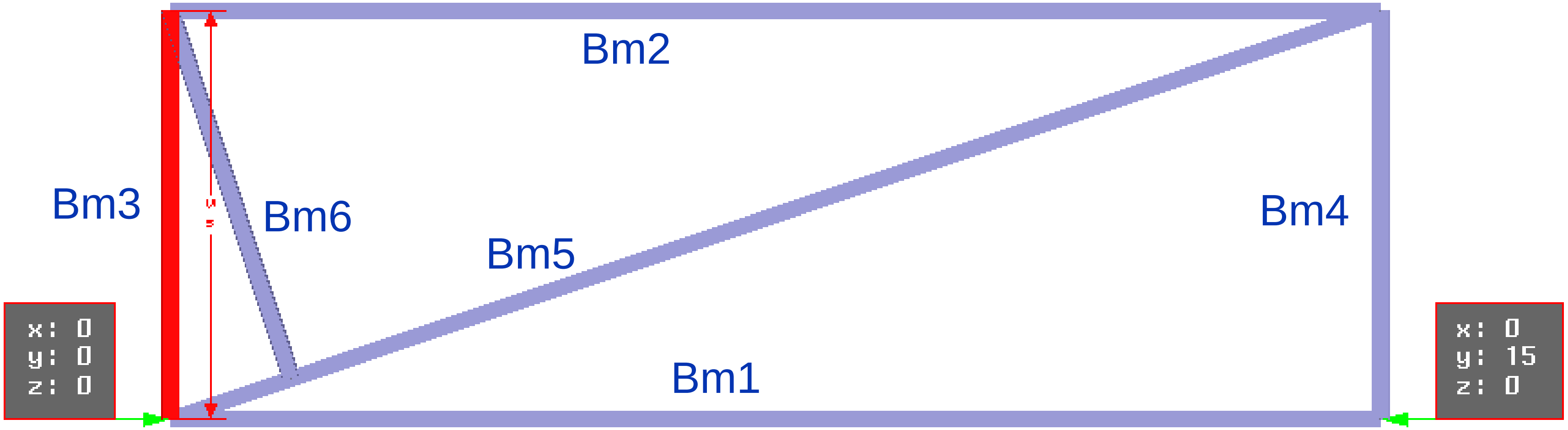
```
Bm1  = Beam(Point(0m,0m,0m),Point(0m,10m,0m));
Bm2  = Bm1.copyTranslate(Vector3d(0m,0m,10m));
Bm3  = Beam(Bm1.end1,Bm2.end1);
Bm4  = Beam(Bm1.end2,Bm2.end2);
Bm5  = Beam(Bm3.end1,Bm4.end2);
Bm6  = Beam(Bm3.end2,Bm5.project(Bm3.end2));
```

The journal file refers to beam ends rather than explicit coordinate values

Parametric models in GeniE

- Modify input parameters and re-run journal file
  - New length of Bm1: 15 m
  - New copy vector: 5 m

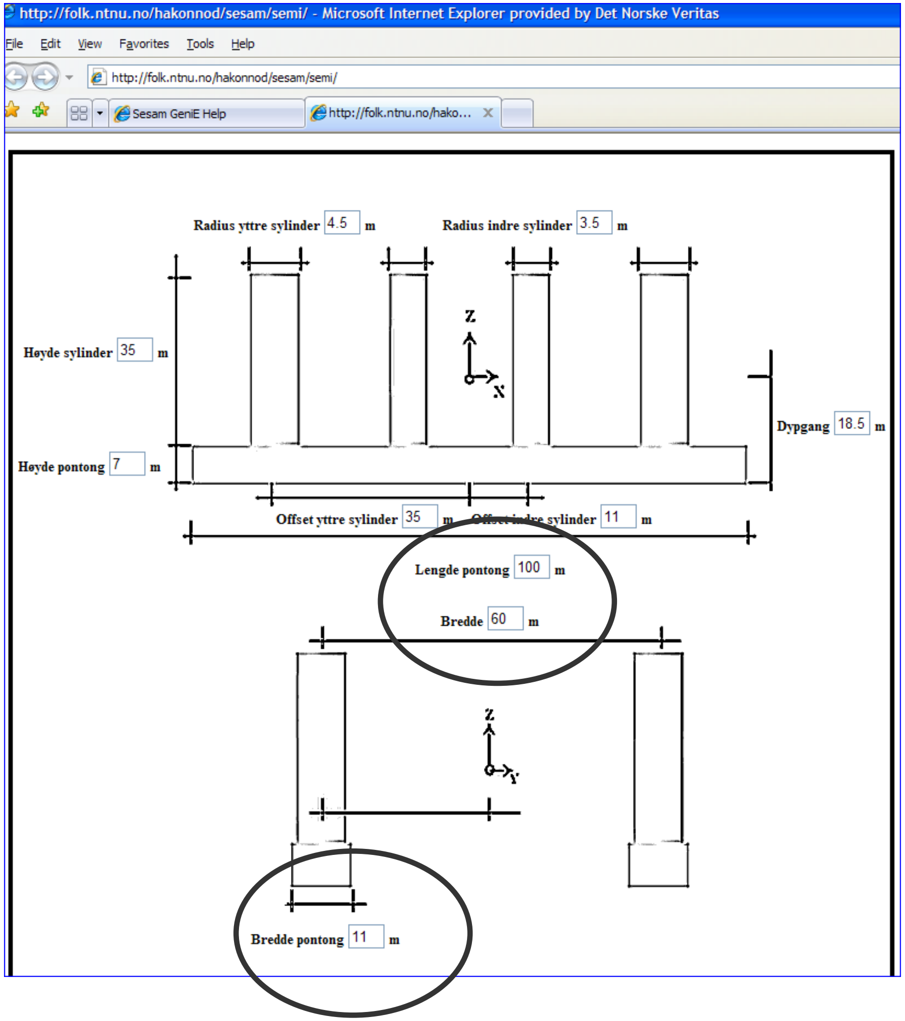
```
Bm1 = Beam(Point(0m,0m,0m),Point(0m,15m,0m));
Bm2 = Bm1.copyTranslate(Vector3d(0m,0m,5m));
Bm3 = Beam(Bm1.end1,Bm2.end1);
Bm4 = Beam(Bm1.end2,Bm2.end2);
Bm5 = Beam(Bm3.end1,Bm4.end2);
Bm6 = Beam(Bm3.end2,Bm5.project(Bm3.end2));
```



- New model automatically created
- Much more advanced models may be created
- GeniE comes with wizards using this feature
- ‘Clean’ js-file (File > Export > Genie journal file) loses reference point

Custom Engineering Application

- Example on student use at a university



http://folk.ntnu.no/hakonnod/sesam/semi/ - Microsoft Internet Explorer provided by Det Norske Veritas

http://folk.ntnu.no/hakonnod/sesam/semi/

Radius yttre sylinder 4.5 m

Radius indre sylinder 3.5 m

Høyde sylinder 35 m

Høyde pontong 7 m

Dypgang 18.5 m

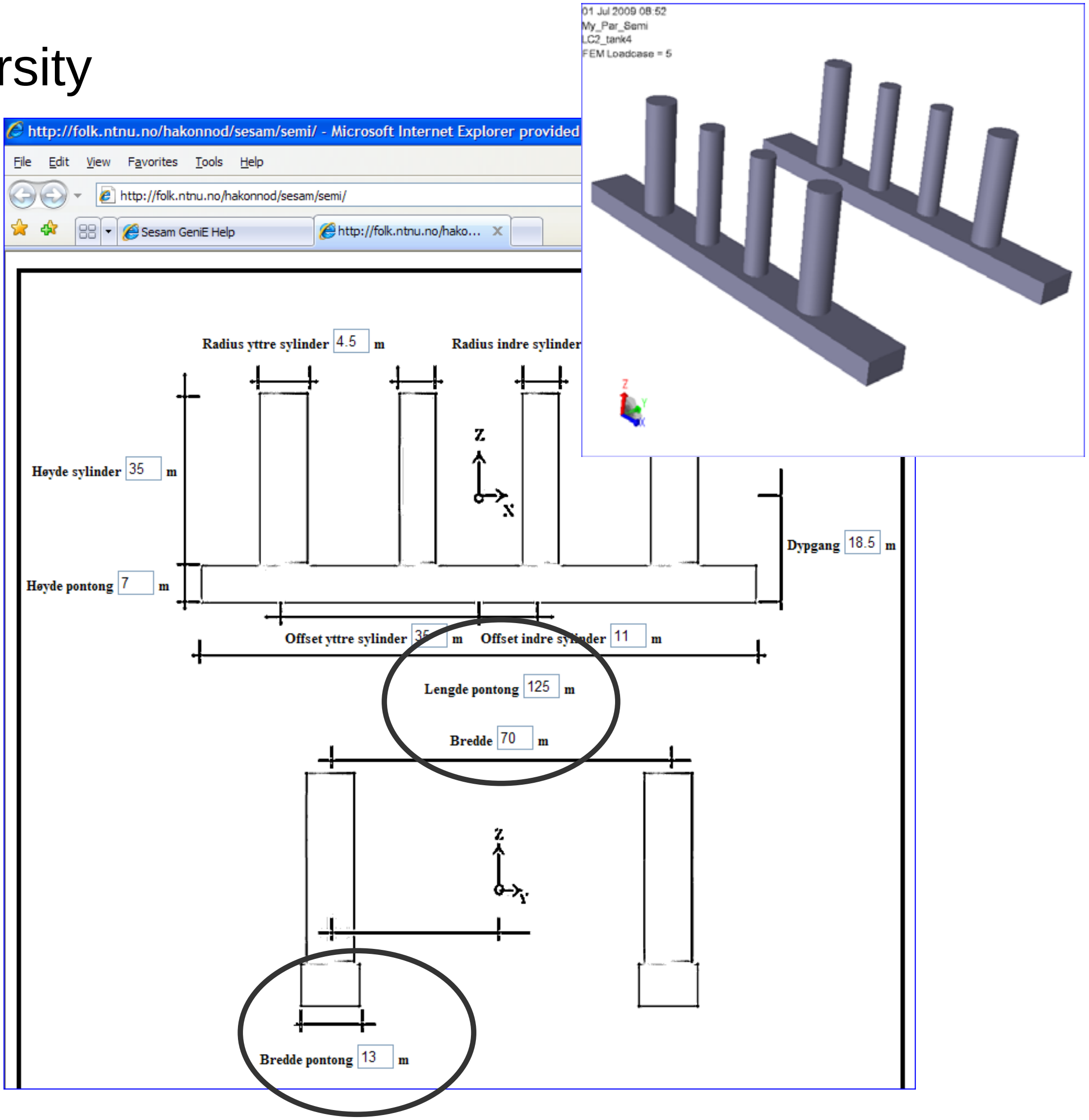
Offset yttre sylinder 35 m

Offset indre sylinder 11 m

Lengde pontong 100 m

Bredde 60 m

Bredde pontong 11 m



http://folk.ntnu.no/hakonnod/sesam/semi/ - Microsoft Internet Explorer provided

http://folk.ntnu.no/hakonnod/sesam/semi/

Radius yttre sylinder 4.5 m

Radius indre sylinder

Høyde sylinder 35 m

Høyde pontong 7 m

Dypgang 18.5 m

Offset yttre sylinder 35 m

Offset indre sylinder 11 m

Lengde pontong 125 m

Bredde 70 m

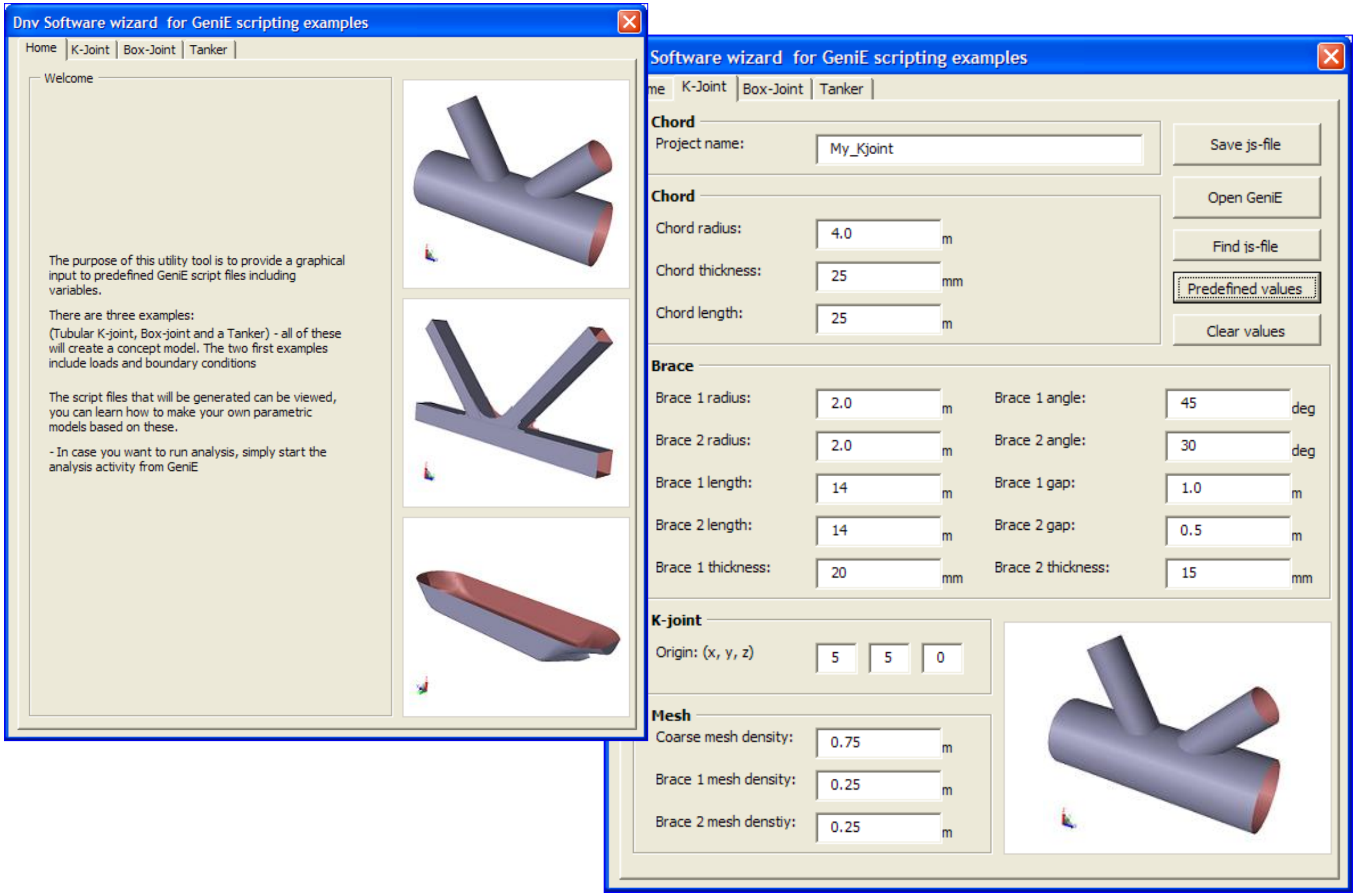
Bredde pontong 13 m

01 Jul 2009 08:52
My_Par_Semi
LC2_tank4
FEM Loadcase = 5

Custom Engineering Application

MANAGING RISK DNV

- Parametric models available from Excel VBA



Dnv Software wizard for GeniE scripting examples

The purpose of this utility tool is to provide a graphical input to predefined GeniE script files including variables.

There are three examples:

(Tubular K-joint, Box-joint and a Tanker) - all of these will create a concept model. The two first examples include loads and boundary conditions

The script files that will be generated can be viewed, you can learn how to make your own parametric models based on these.

- In case you want to run analysis, simply start the analysis activity from GeniE

Software wizard for GeniE scripting examples

Chord

Project name: My_Kjoint

Chord

Chord radius: 4.0 m

Chord thickness: 25 mm

Chord length: 25 m

Brace

Brace 1 radius:	2.0 m	Brace 1 angle:	45 deg
Brace 2 radius:	2.0 m	Brace 2 angle:	30 deg
Brace 1 length:	14 m	Brace 1 gap:	1.0 m
Brace 2 length:	14 m	Brace 2 gap:	0.5 m
Brace 1 thickness:	20 mm	Brace 2 thickness:	15 mm

K-joint

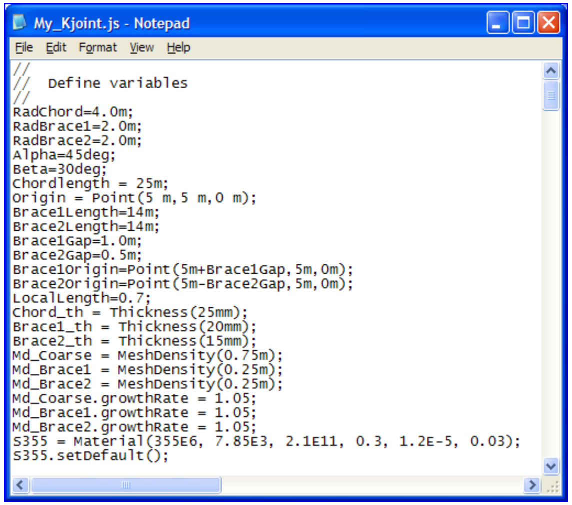
Origin: (x, y, z) 5 5 0

Mesh

Coarse mesh density: 0.75 m

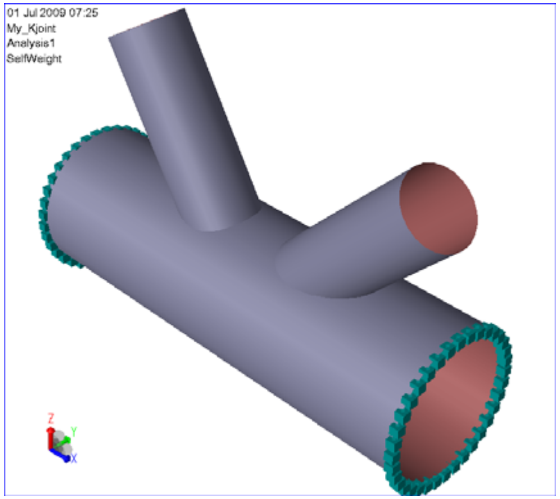
Brace 1 mesh density: 0.25 m

Brace 2 mesh denstiy: 0.25 m



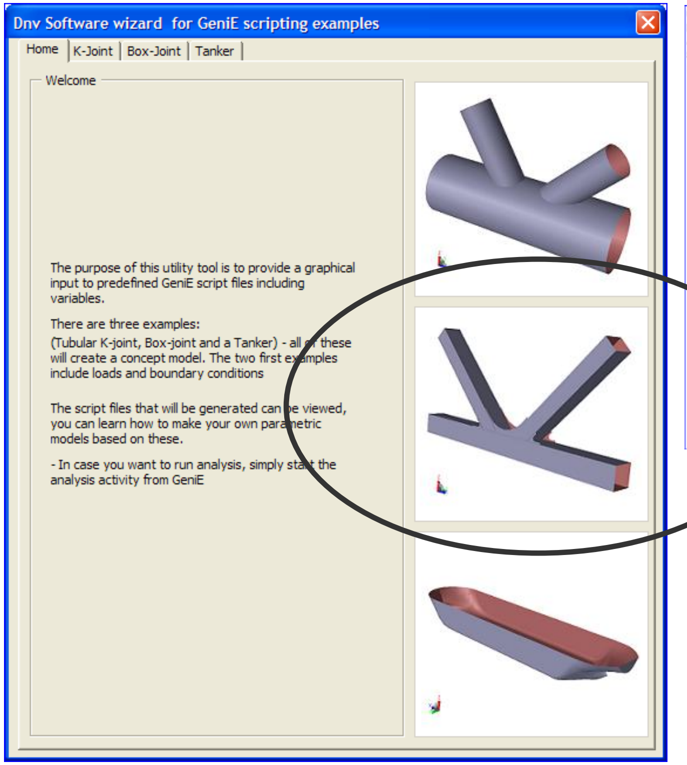
My_Kjoint.js - Notepad

```
//
//  Define variables
//
RadChord=4.0m;
RadBrace1=2.0m;
RadBrace2=2.0m;
Alpha=45deg;
Beta=30deg;
Chordlength = 25m;
Origin = Point(5 m,5 m,0 m);
Brace1Length=14m;
Brace2Length=14m;
Brace1Gap=1.0m;
Brace2Gap=0.5m;
Brace1Origin=Point(5m+Brace1Gap,5m,0m);
Brace2Origin=Point(5m-Brace2Gap,5m,0m);
LocalLength=0.7;
Chord_th = Thickness(25mm);
Brace1_th = Thickness(20mm);
Brace2_th = Thickness(15mm);
Md_Coarse = MeshDensity(0.75m);
Md_Brace1 = MeshDensity(0.25m);
Md_Brace2 = MeshDensity(0.25m);
Md_Coarse.growthRate = 1.05;
Md_Brace1.growthRate = 1.05;
Md_Brace2.growthRate = 1.05;
S355 = Material(355E6, 7.85E3, 2.1E11, 0.3, 1.2E-5, 0.03);
S355.setDefault();
```



Custom Engineering Application

- Parametric models available from Excel VBA



Dnv Software wizard for GeniE scripting examples

Welcome

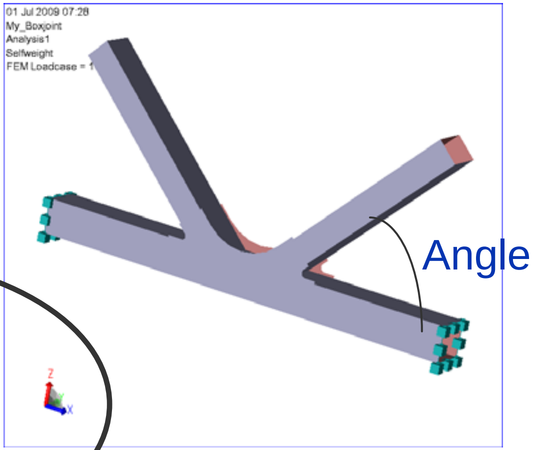
The purpose of this utility tool is to provide a graphical input to predefined GeniE script files including variables.

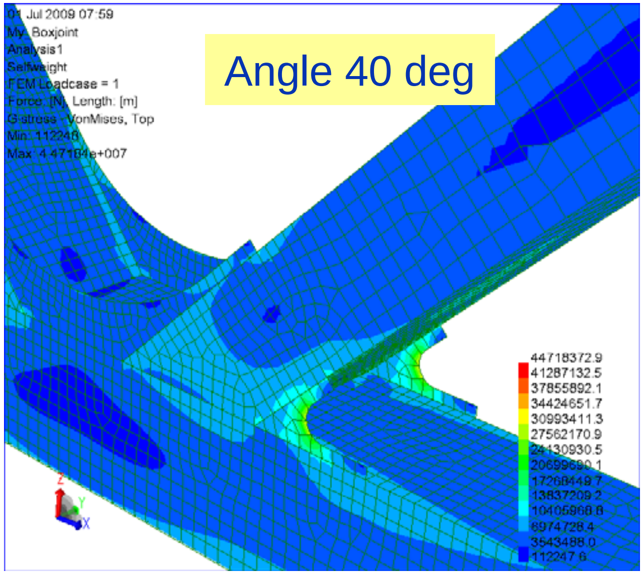
There are three examples:

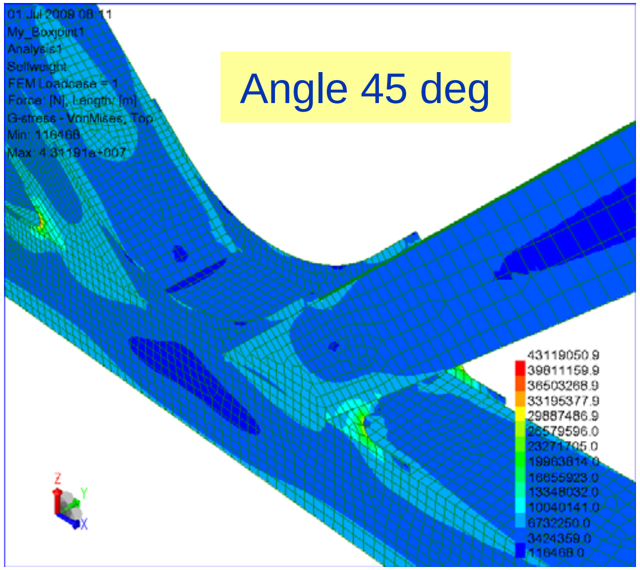
(Tubular K-joint, Box-joint and a Tanker) - all of these will create a concept model. The two first examples include loads and boundary conditions

The script files that will be generated can be viewed, you can learn how to make your own parametric models based on these.

- In case you want to run analysis, simply start the analysis activity from GeniE

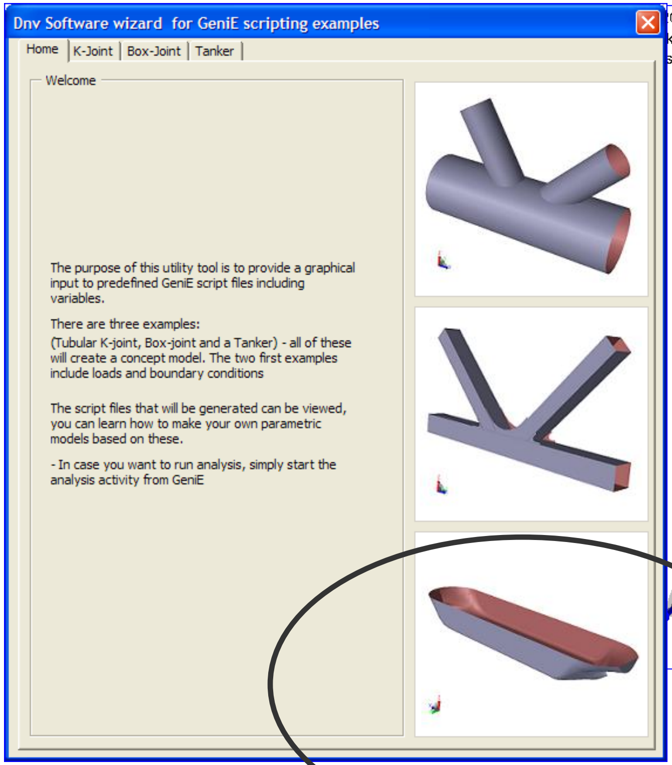






Custom Engineering Application

- Parametric models available from Excel VBA



Dnv Software wizard for GenIE scripting examples

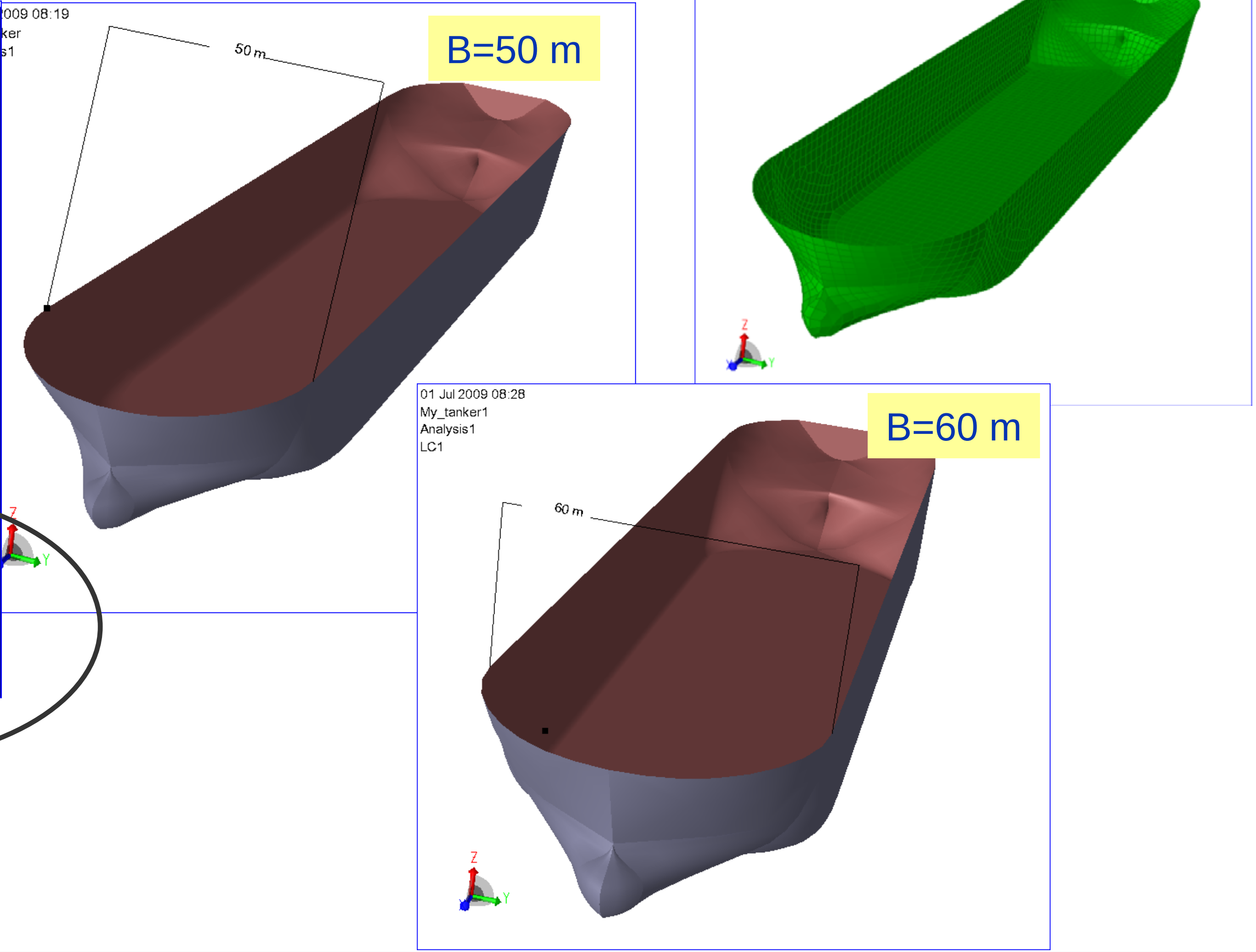
The purpose of this utility tool is to provide a graphical input to predefined GeniE script files including variables.

There are three examples:

(Tubular K-joint, Box-joint and a Tanker) - all of these will create a concept model. The two first examples include loads and boundary conditions

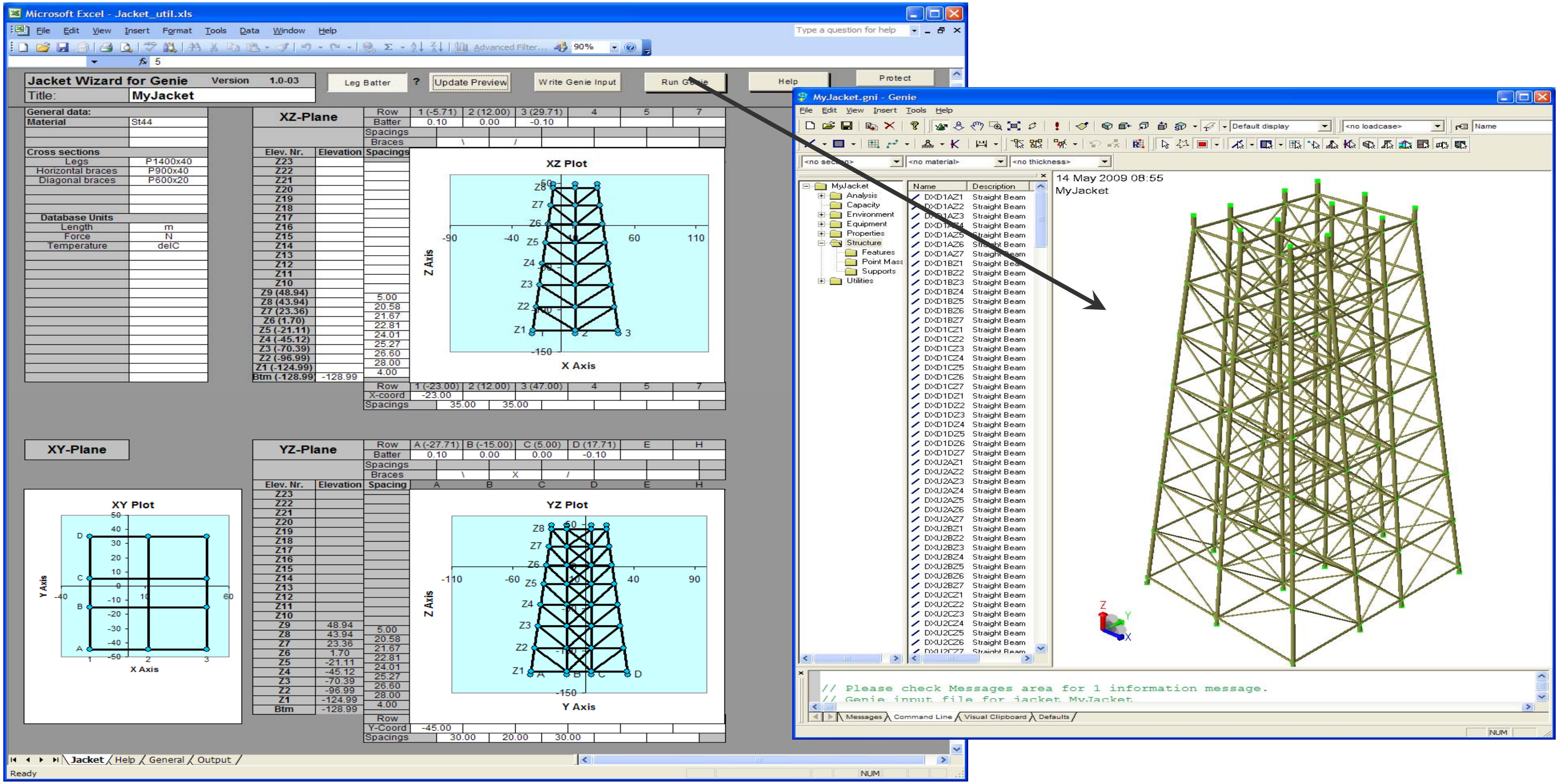
The script files that will be generated can be viewed, you can learn how to make your own parametric models based on these.

- In case you want to run analysis, simply start the analysis activity from GeniE



Custom Engineering Application

- Parametric models as part of the GeniE installation – jacket
  - Help -> Help Topics > Wizard templates



Jacket Wizard for Genie Version 1.0-03

Title: MyJacket

fx 5

General data:	
Material	St44
Cross sections	
Legs	P1400x40
Horizontal braces	P900x40
Diagonal braces	P600x20
Database Units	
Length	m
Force	N
Temperature	delC

XZ-Plane

Row	1 (-5.71)	2 (12.00)	3 (29.71)	4	5	7
Batter	0.10	0.00	-0.10			
Spacings						
Braces	\	/				

Elev. Nr.	Elevation	Spacings
Z23		
Z22		
Z21		
Z20		
Z19		
Z18		
Z17		
Z16		
Z15		
Z14		
Z13		
Z12		
Z11		
Z10		
Z9 (48.94)		5.00
Z8 (43.94)		20.58
Z7 (23.36)		21.67
Z6 (1.70)		22.81
Z5 (-21.11)		24.01
Z4 (-45.12)		25.27
Z3 (-70.39)		26.60
Z2 (-96.99)		28.00
Z1 (-124.99)		4.00
Btm (-128.99)	-128.99	

Row	1 (-23.00)	2 (12.00)	3 (47.00)	4	5	7
X-coord	-23.00					
Spacings	35.00	35.00				

YZ-Plane

Row	A (-27.71)	B (-15.00)	C (5.00)	D (17.71)	E	H
Batter	0.10	0.00	0.00	-0.10		
Spacings						
Braces	\	X	/			

Elev. Nr.	Elevation	Spacing
Z23		
Z22		
Z21		
Z20		
Z19		
Z18		
Z17		
Z16		
Z15		
Z14		
Z13		
Z12		
Z11		
Z10		
Z9	48.94	5.00
Z8	43.94	20.58
Z7	23.36	21.67
Z6	1.70	22.81
Z5	-21.11	24.01
Z4	-45.12	25.27
Z3	-70.39	26.60
Z2	-96.99	28.00
Z1	-124.99	4.00
Btm	-128.99	

Row						
Y-Coord	-45.00					
Spacings	30.00	20.00	30.00			

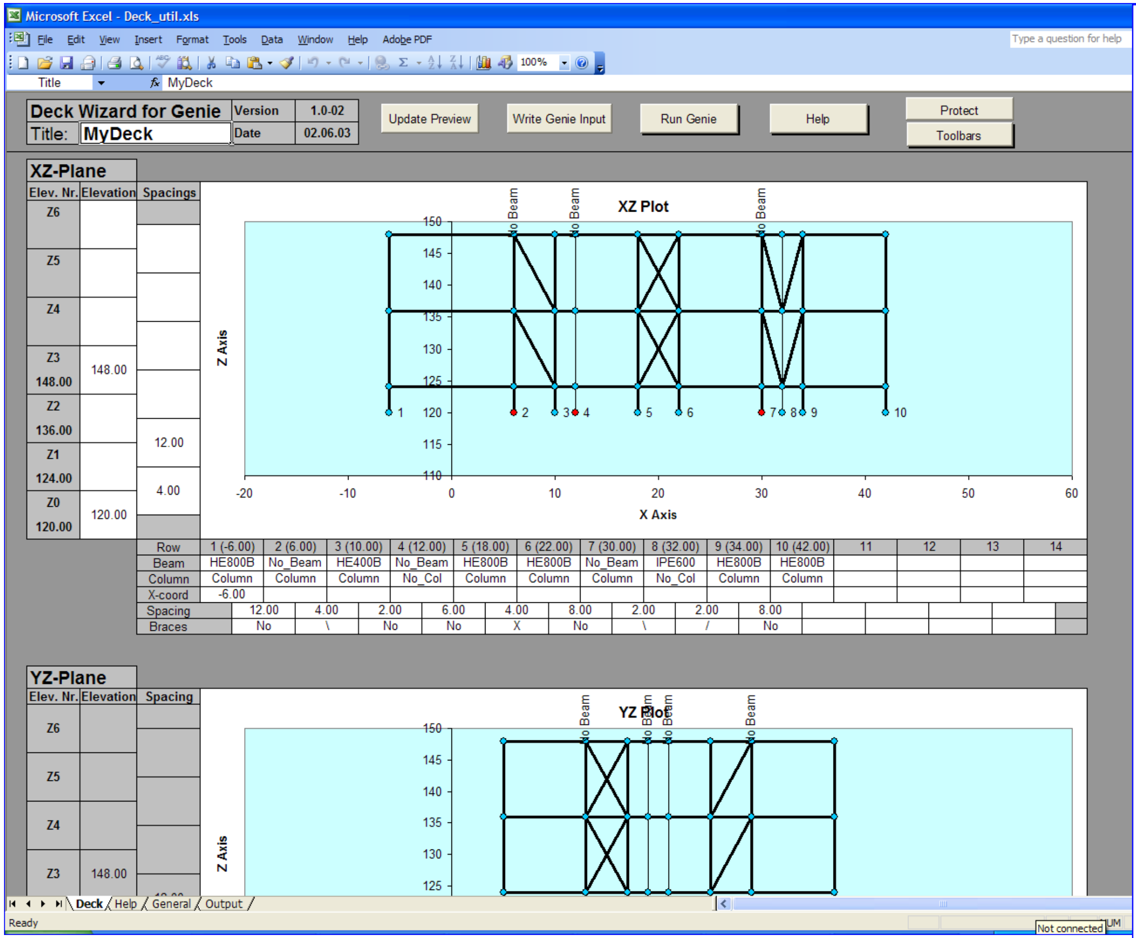
14 May 2009 08:55
MyJacket

Name	Description
DXD1AZ1	Straight Beam
DXD1AZ2	Straight Beam
DXD1AZ3	Straight Beam
DXD1AZ4	Straight Beam
DXD1AZ5	Straight Beam
DXD1AZ6	Straight Beam
DXD1AZ7	Straight Beam
DXD1BZ1	Straight Beam
DXD1BZ2	Straight Beam
DXD1BZ3	Straight Beam
DXD1BZ4	Straight Beam
DXD1BZ5	Straight Beam
DXD1BZ6	Straight Beam
DXD1BZ7	Straight Beam
DXD1CZ1	Straight Beam
DXD1CZ2	Straight Beam
DXD1CZ3	Straight Beam
DXD1CZ4	Straight Beam
DXD1CZ5	Straight Beam
DXD1CZ6	Straight Beam
DXD1CZ7	Straight Beam
DXD1DZ1	Straight Beam
DXD1DZ2	Straight Beam
DXD1DZ3	Straight Beam
DXD1DZ4	Straight Beam
DXD1DZ5	Straight Beam
DXD1DZ6	Straight Beam
DXD1DZ7	Straight Beam
DXU2AZ1	Straight Beam
DXU2AZ2	Straight Beam
DXU2AZ3	Straight Beam
DXU2AZ4	Straight Beam
DXU2AZ5	Straight Beam
DXU2AZ6	Straight Beam
DXU2AZ7	Straight Beam
DXU2BZ1	Straight Beam
DXU2BZ2	Straight Beam
DXU2BZ3	Straight Beam
DXU2BZ4	Straight Beam
DXU2BZ5	Straight Beam
DXU2BZ6	Straight Beam
DXU2BZ7	Straight Beam
DXU2CZ1	Straight Beam
DXU2CZ2	Straight Beam
DXU2CZ3	Straight Beam
DXU2CZ4	Straight Beam
DXU2CZ5	Straight Beam
DXU2CZ6	Straight Beam
DXU2CZ7	Straight Beam

```
// Please check Messages area for 1 information message.
// Genie input file for jacket MyJacket
```

Custom Engineering Application

- Parametric models as part of the GeniE installation – topside
  - Help -> Help Topics > Wizard templates



Microsoft Excel - Deck_util.xls

Deck Wizard for Genie	Version	1.0-02
Title: MyDeck	Date	02.06.03

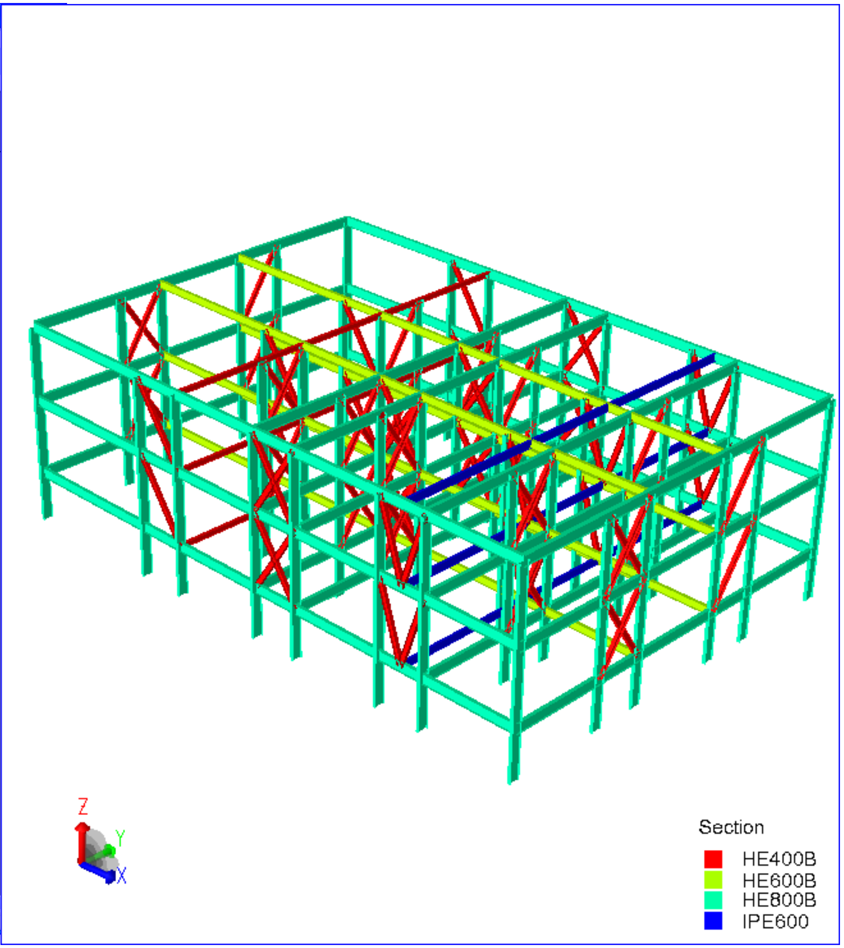
XZ-Plane

Elev. Nr.	Elevation	Spacings
Z6		
Z5		
Z4		
Z3 148.00	148.00	
Z2 136.00		12.00
Z1 124.00		4.00
Z0 120.00	120.00	

Row	1 (-6.00)	2 (6.00)	3 (10.00)	4 (12.00)	5 (18.00)	6 (22.00)	7 (30.00)	8 (32.00)	9 (34.00)	10 (42.00)	11	12	13	14
Beam	HE800B	No_Beam	HE400B	No_Beam	HE800B	HE800B	No_Beam	IPE600	HE800B	HE800B				
Column	Column	Column	Column	No_Col	Column	Column	Column	No_Col	Column	Column				
X-coord	-6.00													
Spacing	12.00	4.00	2.00	6.00	4.00	8.00	2.00	2.00	8.00					
Braces	No	\	No	No	X	No	\	/	No					

YZ-Plane

Elev. Nr.	Elevation	Spacing
Z6		
Z5		
Z4		
Z3	148.00	

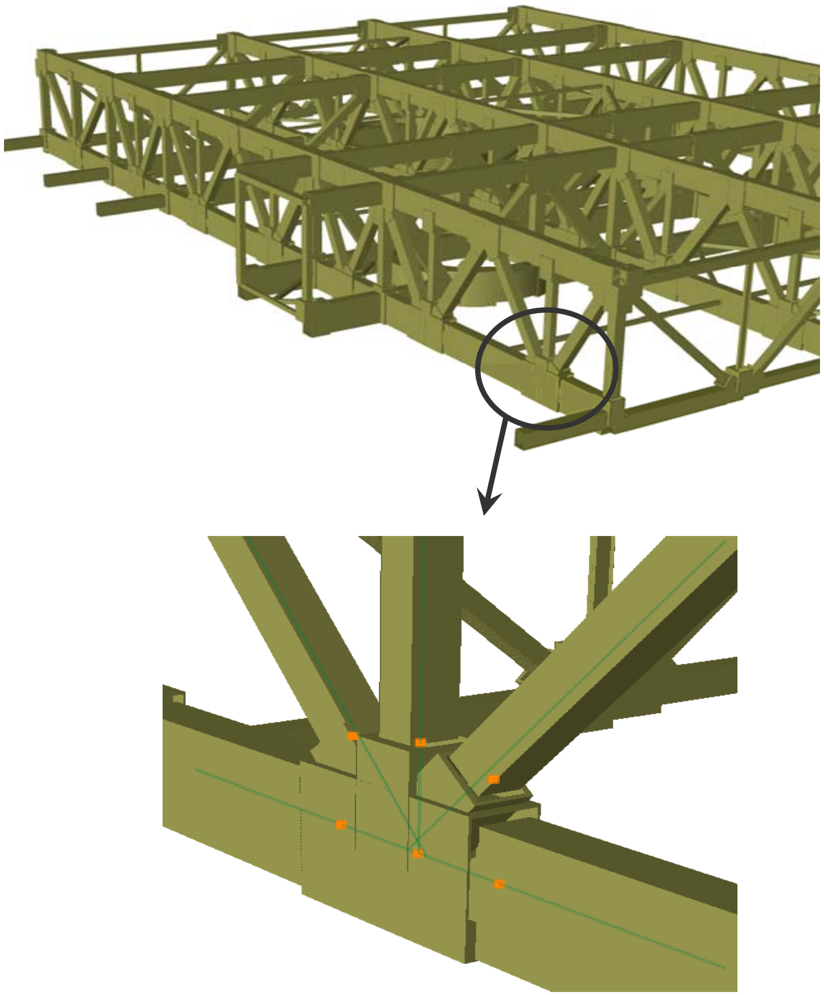


Section

🟥	HE400B
🟨	HE600B
🟩	HE800B
🟦	IPE600

Parametric models in GeniE

- Example: Extensive model update
  - Apply reinforcement and offset for gusset plates at all beam ends
- Make a function to
  - divide each beam into segments
  - set segment cross sections
  - set beam end offsets
- Call the function for each beam using tabulated data from drawings
- Faster, easier to check, more fun than manually editing hundreds of beams



Using a function to update the beam ends

```
//  First list the beams to be updated
//  in an array
var Bea = new Array();
Bea[1] = Dx1A1;
Bea[2] = Dx1C1;
Bea[3] = Dx1E1;
Bea[4] = Dx1G1;
//  and many more
//
//  Define eccentricities and
//  offsets for these beams
//
End1_X_center = 0.15 m;          =>          These data were listed in tables on the drawings
End2_X_center = 0.16 m;
End1_Offset = 1.837 m;
End2_Offset = 2.102 m;
Main_Section = D39;
End1_Section = Node_Diagonal_X;
End2_Section = Node_Diagonal_X;
```

```
//  Then call a user function to do the update
//
DiagonalProp (Bea,End1_Offset,End2_Offset,
Main_Section,End1_X_center,End2_X_center,End
1_Section,End2_Section);
//
```

DiagonalProp – function to update beams

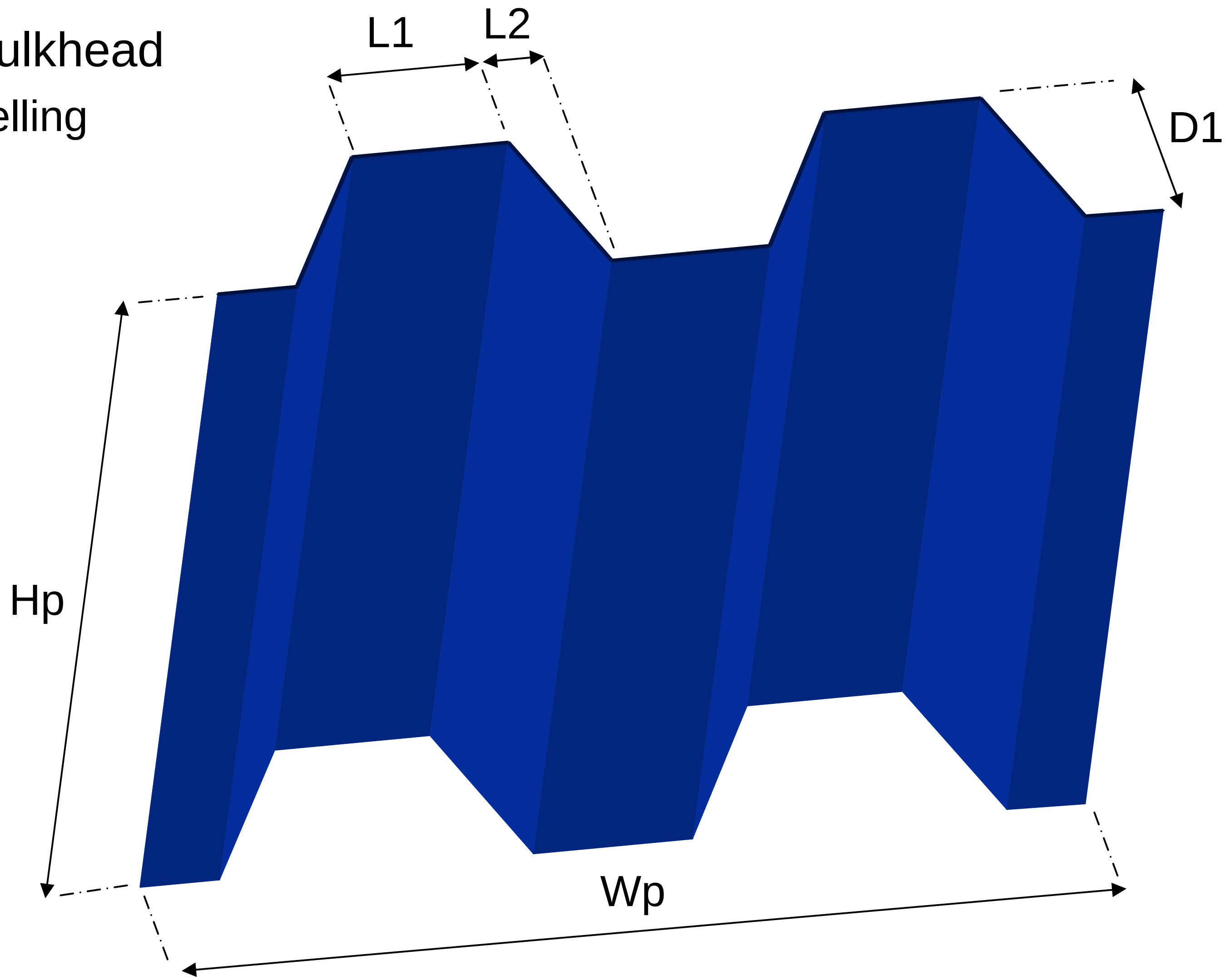
```
function DiagonalProp (Bea,End1_Offset,End2_Offset,Main_Section,
  End1_X_center,End2_X_center, End1_Section,End2_Section)    {
  var NumBeam = Bea.length;
   for (i = 1; i < NumBeam ; i++) {
          Bea[i].section = Main_Section;
          //
          // Eccentricities modelled by EndOffset
          //
          if ( End1_X_center != 0. m )
                    { Bea[i].setEndOffset(1,
                    Vector3d(End1_X_center, 0 m,0 m));
          }
          if ( End2_X_center != 0. m )
                    { Bea[i].setEndOffset(2,
                    Vector3d(End2_X_center, 0 m,0 m));
          }
  // Continue on next page…
```

DiagonalProp – function continued…

```
// Set End1 Offset and/or End2 offset
// The beam is divided at relative position $L_{segment}$ / $L_{beam}$
if ( End1_Offset != 0. m) {
         Bea[i].divideSegmentAt (1,(End1_Offset / Bea[i].length()));
         Bea[i].SetSegmentSection(1, End1_Section);
     if ( End2_Offset != 0. m) {
         Bea[i].divideSegmentAt
         (2,((Bea[i].getSegmentLength(2)-End2_Offset)/Bea[i].getSegmentLength(2)));
         Bea[i].SetSegmentSection(3, End2_Section);
         }
   }
.... and similar for End2...
```

Parametric modelling - example

- Making a corrugated bulkhead
  - Reference point modelling and variables
  - Pure scripting



Parametric Modelling by Reference Point

- Reference point modelling will journal
  - relative to other structure - Point1 = Curve1.curvePoint(1);
  - not actual coordinates - Point2 = Point(1 m,0.4 m,0 m);
- Reference point modelling can be used to make parametric models

// Read the measurements from a journal file or define yourself

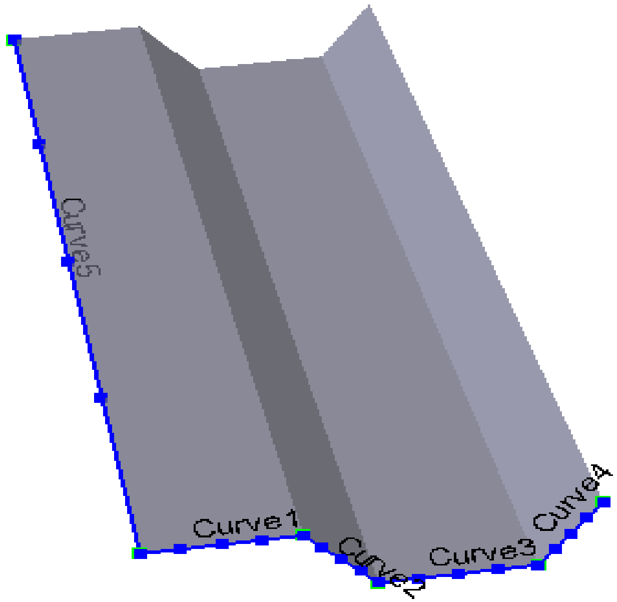
Hp = 6.23 m; // Height of plate

Wp = 12.32 m; // Width of plate

L1 = 700 mm; // Length of sub panel

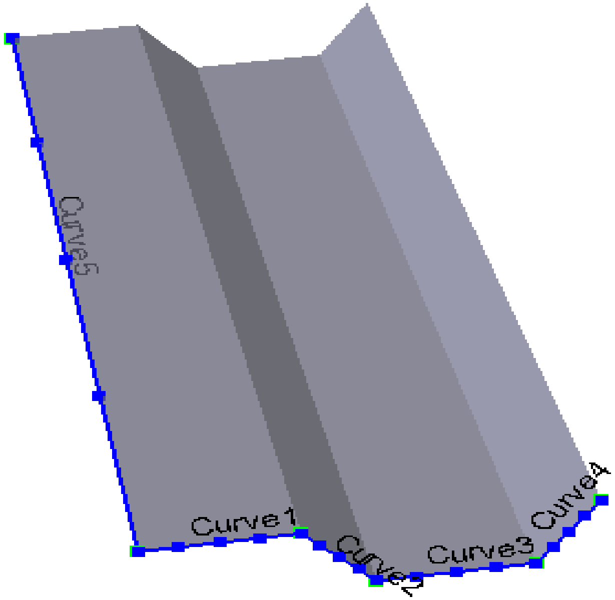
L2 = 300 mm; // Length of corrugation

D1 = 400 mm; // Depth of corrugation



Parametric Modelling by Reference Point

```
// Then model interactively using reference point modelling
Point1 = Point(0,0,0);
Point2 = Point1.copyTranslate(Vector3d(L1,0,0));
Point3 = Point2.copyTranslate(Vector3d(L2,D1,0));
Point4 = Point3.copyTranslate(Vector3d(L1,0,0));
Point5 = Point4.copyTranslate(Vector3d(L2,-D1,0));
Curve1 = GuideLine(Point1, Point2, 3);
Curve2 = GuideLine(Point2, Point3, 3);
Curve3 = GuideLine(Point3, Point4, 3);
Curve4 = GuideLine(Point4, Point5, 3);
Point6 = Point1.copyTranslate(Vector3d(0,0,Hp));
Curve5 = GuideLine(Point1, Point6, 3);
Pl1 = SweepCurve(Curve1, Curve5);
Pl2 = SweepCurve(Curve2, Curve5);
Pl3 = SweepCurve(Curve3, Curve5);
```



Parametric modelling by scripting

- Start with a section defining input parameter values:

```
Hp = 6.23 m;  // Height of plate
Wp = 12.32 m; // Width of plate
L1 = 700 mm; // Length of sub panel
L2 = 300 mm; // Length of corrugation
D1 = 400 mm; // Depth of corrugation
```

Parametric modelling by scripting

- Then use the parameter names in GeniE commands to create your model

```
var i = 0;  // Counter for sub panels
var j = 0;  // Counter for corrugations = 4 sub panels
var Curves = new Array();  // Guide curves at the base of the plate
// Create sub panels
do {
  var StartP = (L1+L2)*2*j;      // Start point for the next corrugation
  if ( PartNr == 1) { Curves[i] = GuideLine(Point(StartP,0,0), Point((StartP+L1),0,0), 3);
        rename(Curves[i], "Line_"+i);
        LengthPlate = LengthPlate + L1;
        PartNr = 2;}
```

- To update the model you only need to enter new parameter values

GeniE & JScript – what you need to know

- Basic JScript
  - Operators, data types, control flow, functions and objects
- Basic GeniE commands
  - GeniE objects, their functions and properties
  - The commands you get when you work interactively
- GeniE utility commands
  - ds1=DynamicSet(LimitInPlane(ZPlane3d(0)));
  - for(var sect in ModelObjects)
  - if(a.supportsType(typeSection))
- Practical applications
  - Creating objects
  - Setting and getting object properties
  - Addressing subsets of your model
  - Addressing objects by their name
  - Creating your own functions

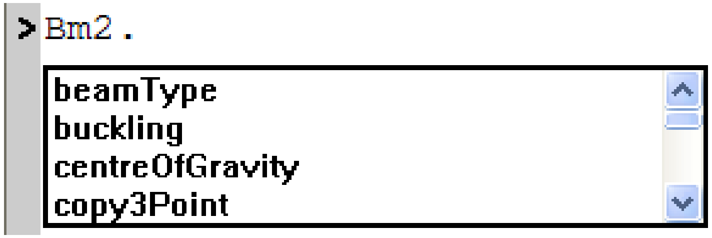
Documentation of GeniE JScript

- JScript – Microsoft Developers network
  - http://msdn.microsoft.com/en-us/library/hbxc2t98(VS.85).aspx
- GeniE Help – Volume 6 and JScript Commands
  - Overview
    - The GeniE commands
    - Example:
      - Beam (three methods to create a beam)
  - Class Hierarchy
    - The GeniE objects and properties
    - Example:

```
NamedObject
   - Transformable
      - BasicConcept
         - BasicBeam
            - StraigthBeam
```

Documentation of GeniE JScript

- Run GeniE from GUI – look at the journal
- Use the Command window – tab completion



```
> Bm2.
beamType
buckling
centreOfGravity
copy3Point
```

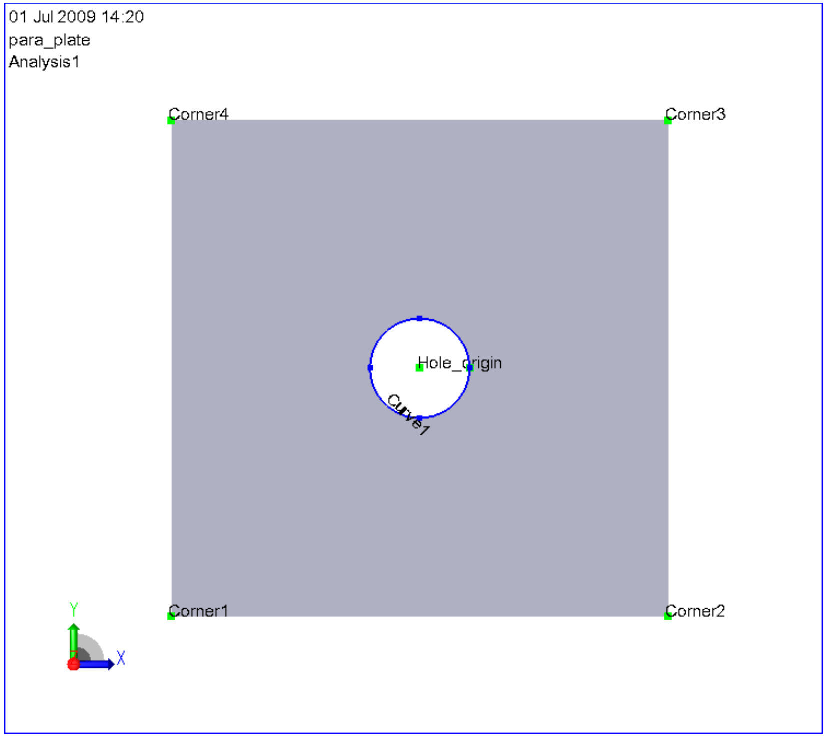
- Open the folder with GeniE Help files;
  C:\Program Files\DNVS\GeniE\Help\jscript
  in file explorer and search for word in file

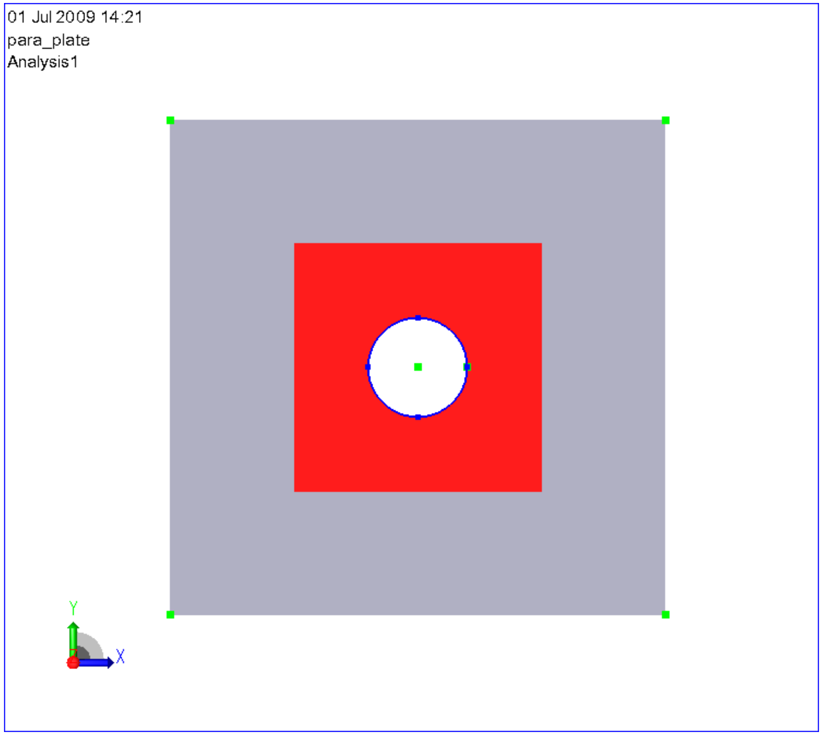
Developing Scripts

- Use an editor to type the script file (Input.js)
- Then read the script into GeniE by
  - File > Read Command file
  - File > Recent Command files
  - Copy/Paste from editor to the GeniE command window
- Once you have tried a chunk of commands all variables, objects and functions will reside in the workspace until you close the workspace and create a new. To be sure use a new workspace each time you test your script.
- While developing your scriptfile you may
  - do things interactively in GeniE to get the command syntax
  - use the Command window > Tab completion to see available functions for your objects

A practical example

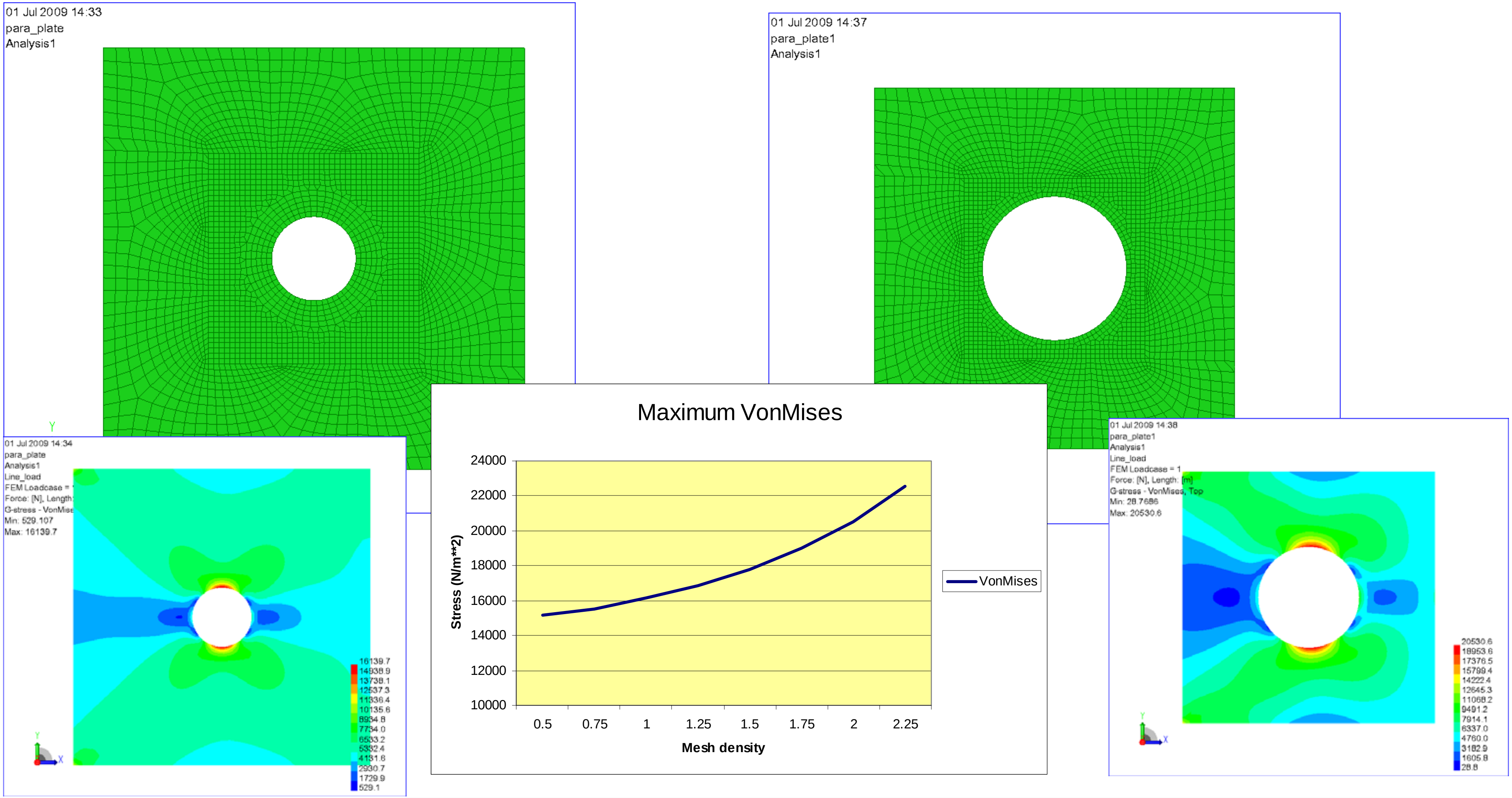
- Make a plate with a variable hole – check the VonMises stresses for R=0.5m, 1.0m, 1.5m and 2.0 m
- The inner part has a refined mesh and is based on paver meshing
- This example includes the parametric input file “Param_plate_in.js”


01 Jul 2009 14:20
para_plate
Analysis1


01 Jul 2009 14:21
para_plate
Analysis1

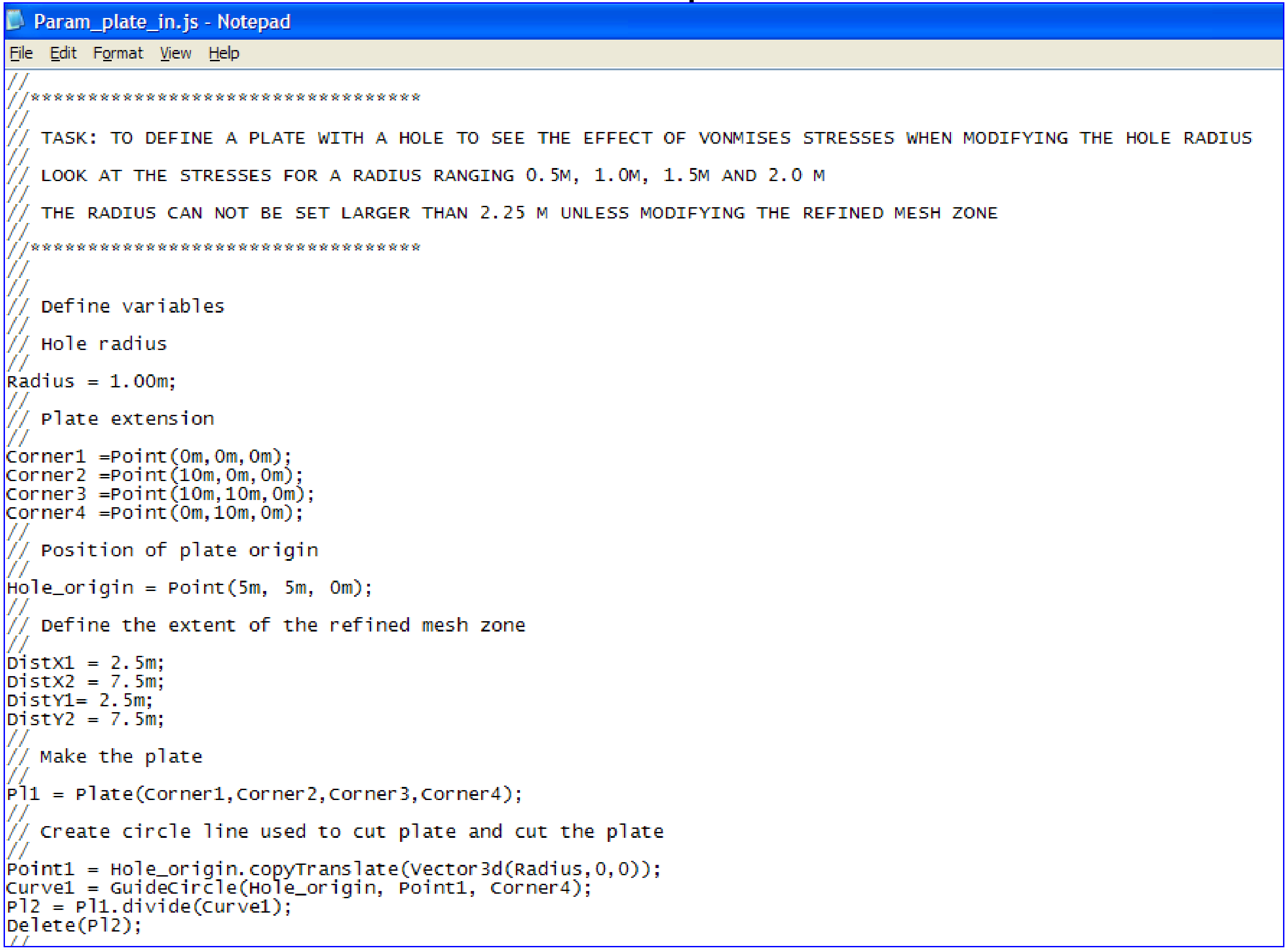
A practical example

- Radius 1.0 m -> VonMises 16139 N/m^2
- Radius 2.0 m -> VonMises 20531 N/m^2



A practical example

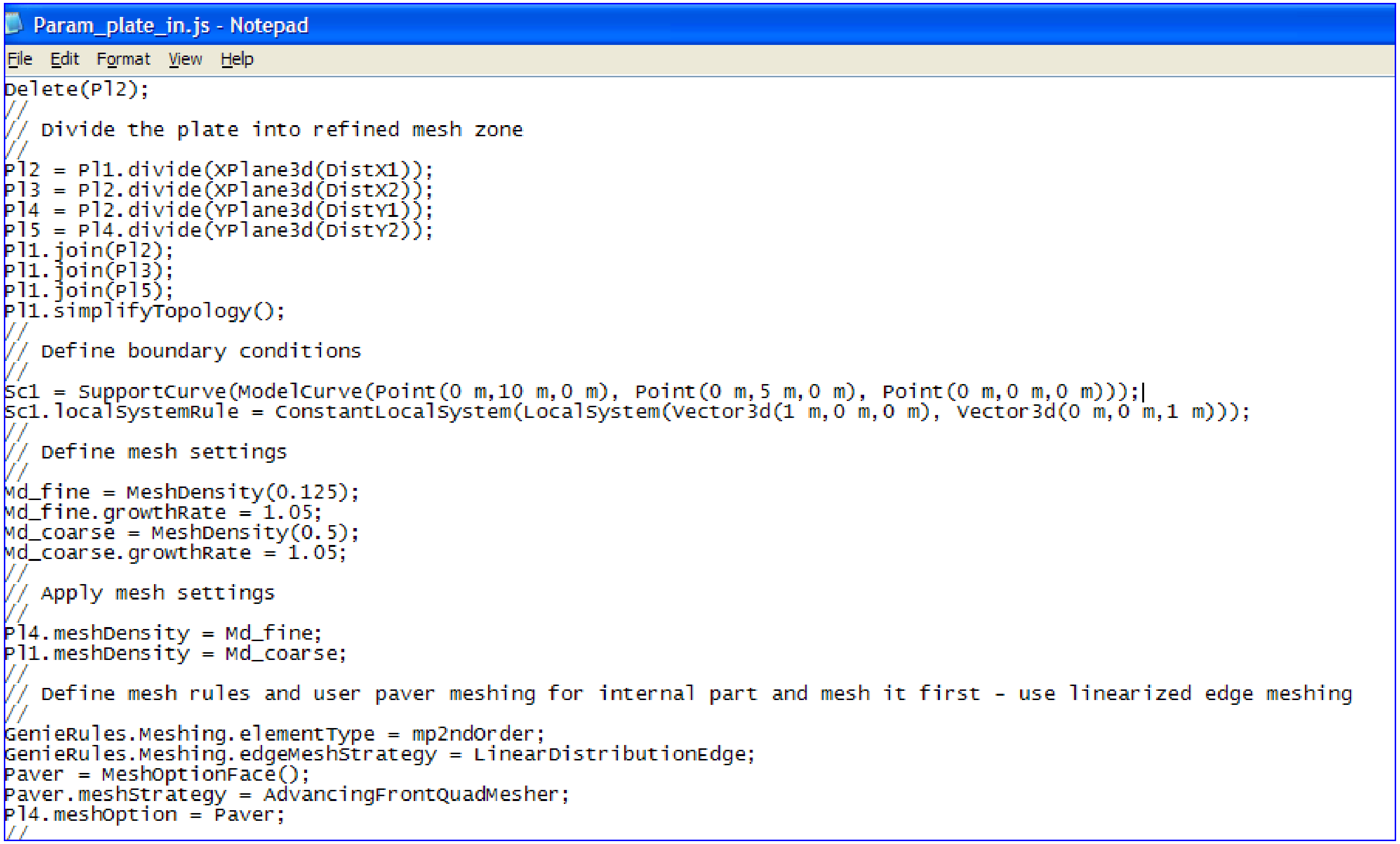
- Follow the hints in the shown js-file - 1



```
//
//**********************************
//
// TASK: TO DEFINE A PLATE WITH A HOLE TO SEE THE EFFECT OF VONMISES STRESSES WHEN MODIFYING THE HOLE RADIUS
//
// LOOK AT THE STRESSES FOR A RADIUS RANGING 0.5M, 1.0M, 1.5M AND 2.0 M
//
// THE RADIUS CAN NOT BE SET LARGER THAN 2.25 M UNLESS MODIFYING THE REFINED MESH ZONE
//
//**********************************
//
//
// Define variables
//
// Hole radius
//
Radius = 1.00m;
//
// Plate extension
//
Corner1 =Point(0m,0m,0m);
Corner2 =Point(10m,0m,0m);
Corner3 =Point(10m,10m,0m);
Corner4 =Point(0m,10m,0m);
//
// Position of plate origin
//
Hole_origin = Point(5m, 5m, 0m);
//
// Define the extent of the refined mesh zone
//
DistX1 = 2.5m;
DistX2 = 7.5m;
DistY1= 2.5m;
DistY2 = 7.5m;
//
// Make the plate
//
Pl1 = Plate(Corner1,Corner2,Corner3,Corner4);
//
// Create circle line used to cut plate and cut the plate
//
Point1 = Hole_origin.copyTranslate(Vector3d(Radius,0,0));
Curve1 = GuideCircle(Hole_origin, Point1, Corner4);
Pl2 = Pl1.divide(Curve1);
Delete(Pl2);
//
```

A practical example

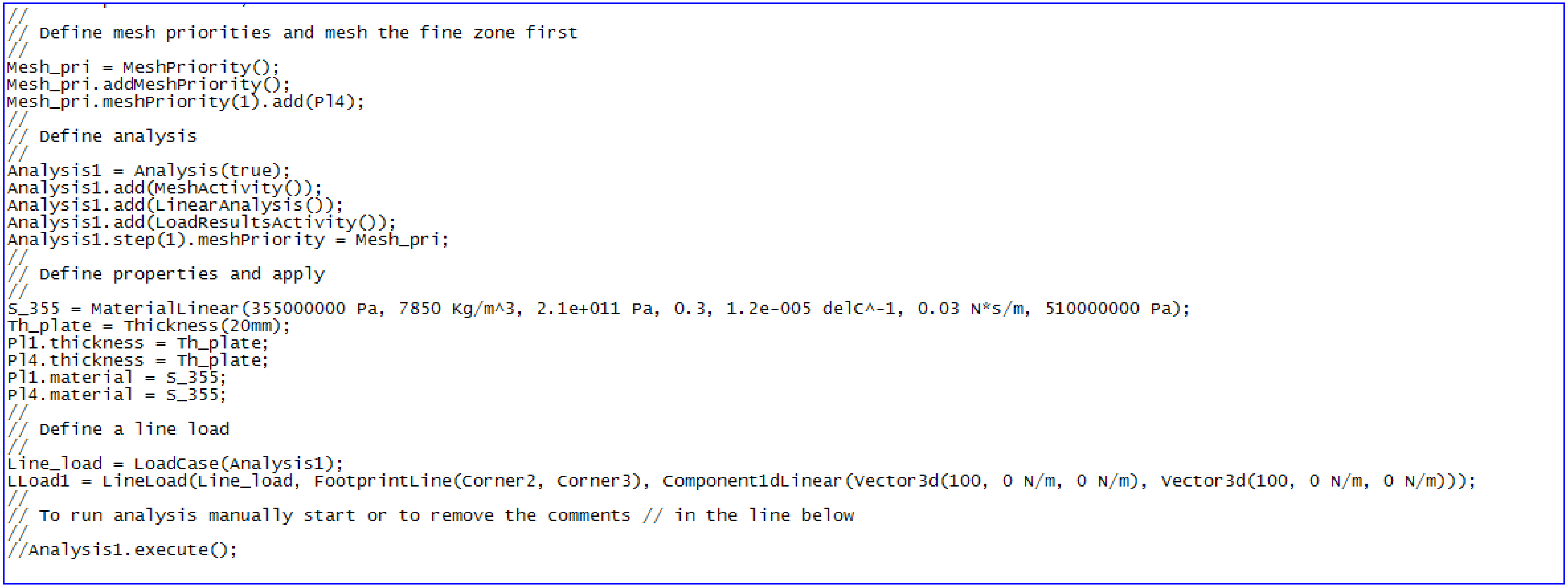
- Follow the hints in the shown js-file - 2



```
Delete(Pl2);
//
// Divide the plate into refined mesh zone
//
Pl2 = Pl1.divide(XPlane3d(DistX1));
Pl3 = Pl2.divide(XPlane3d(DistX2));
Pl4 = Pl2.divide(YPlane3d(DistY1));
Pl5 = Pl4.divide(YPlane3d(DistY2));
Pl1.join(Pl2);
Pl1.join(Pl3);
Pl1.join(Pl5);
Pl1.simplifyTopology();
//
// Define boundary conditions
//
Sc1 = SupportCurve(ModelCurve(Point(0 m,10 m,0 m), Point(0 m,5 m,0 m), Point(0 m,0 m,0 m)));
Sc1.localSystemRule = ConstantLocalSystem(LocalSystem(Vector3d(1 m,0 m,0 m), Vector3d(0 m,0 m,1 m)));
//
// Define mesh settings
//
Md_fine = MeshDensity(0.125);
Md_fine.growthRate = 1.05;
Md_coarse = MeshDensity(0.5);
Md_coarse.growthRate = 1.05;
//
// Apply mesh settings
//
Pl4.meshDensity = Md_fine;
Pl1.meshDensity = Md_coarse;
//
// Define mesh rules and user paver meshing for internal part and mesh it first - use linearized edge meshing
//
GenieRules.Meshing.elementType = mp2ndOrder;
GenieRules.Meshing.edgeMeshStrategy = LinearDistributionEdge;
Paver = MeshOptionFace();
Paver.meshStrategy = AdvancingFrontQuadMesher;
Pl4.meshOption = Paver;
//
```

A practical example

- Follow the hints in the shown js-file – 2



```
//
// Define mesh priorities and mesh the fine zone first
//
Mesh_pri = MeshPriority();
Mesh_pri.addMeshPriority();
Mesh_pri.meshPriority(1).add(Pl4);
//
// Define analysis
//
Analysis1 = Analysis(true);
Analysis1.add(MeshActivity());
Analysis1.add(LinearAnalysis());
Analysis1.add(LoadResultsActivity());
Analysis1.step(1).meshPriority = Mesh_pri;
//
// Define properties and apply
//
S_355 = MaterialLinear(355000000 Pa, 7850 Kg/m^3, 2.1e+011 Pa, 0.3, 1.2e-005 delC^-1, 0.03 N*s/m, 510000000 Pa);
Th_plate = Thickness(20mm);
Pl1.thickness = Th_plate;
Pl4.thickness = Th_plate;
Pl1.material = S_355;
Pl4.material = S_355;
//
// Define a line load
//
Line_load = LoadCase(Analysis1);
LLoad1 = LineLoad(Line_load, FootprintLine(Corner2, Corner3), Component1dLinear(Vector3d(100, 0 N/m, 0 N/m), Vector3d(100, 0 N/m, 0 N/m)));
//
// To run analysis manually start or to remove the comments // in the line below
//
//Analysis1.execute();
```

- You can “cheat” by using the file “Param_plate_in.js”

For the advanced users

- The rest of this handout includes documentation for the advanced user who wants to learn how to make parametric models by pure scripting.

Basic JScript

- All commands end with a semicolon ;
- Comments
  - /*	Here is a block of comments
    - Another comment line
  - */
  - // Here is a single comment
- Print
  - MyVariable = “Hello there”;
  - Print(“MyVariable = “ + MyVariable);

- Operators

+	addition,	-	subtraction,
*	multiplication,	/	division
++	increment,	--	decrement

Logical:

<	Less,	>	Greater,
==	Equal,	!=	Inequality,
&&	And,	||	Or
!	Not,		

Variables and data types

MANAGING RISK DNV

- var MyBeam

  // Declaration of a new variable

- var MyBeam, MyPlate

  // Declaration of two new variables

- var MyBeam = Beam(Point(1,0,24), Point(13,1,18));

  // declaration and initialization

- JScript is a loosely typed language

  Variables have no predetermined type

    JScript variables get a type that corresponds to the type of value they contain

Basic JScript data types

- Primary Data Types
  - String
  - Number
  - Boolean
- Composite
  - Object
  - Array
- Special
  - Null
  - Undefined

GeniE data types

- All engineering objects have types
  - StraightBeam, Plate, SupportPoint, PointLoad + many more
  - Print(Bm1.supportsType(typeStraightBeam));

    -> 1   (1 true, 0 is false)

- General GeniE types
  - Length, Mass, Vector3D, LocalSystem, Double and some more

  - Type Length & Mass support toDouble and toString
    - Radius = 2.34 m
    - Math.pow(Radius.toDouble(),2)

GeniE data types - some examples:

- Print(Bm2.end1.x());

  -> 5 m // that is, x is of type length

- Number versus Length
  - Point(3,4,5); // three numbers converted to lengths
  - Point(3+5.8, 4, 5); // adding two numbers is OK
  - Point(3m+5.8mm, 4, 5); // adding two lengths is OK
  - Point(3m+5.8, 4, 5); // adding a length + number gives error

    type + floating point
  - Point(3m*1.2, 4, 5); // Length * number is OK
  - Point(3m+Length(5.8), 4, 5);

    // The function Length will convert (5.8) to length

- GeniE can interprete several input formats
  - Point(3,4,5);
  - Point(Length(3)+5.8mm,4,5);

- Optional functions to change type - parseFloat, parseInt

Understanding the structure of GeniE commands

MANAGING RISK DNV

- Native GeniE commands - Object notation
- Create:

```
Bm1 = Beam(Point(4 m,20 m,0 m), Point(4 m,0 m,0 m));
```

(Beam is a constructor function to create an instance, Bm1, of the straigthBeam class)

- Set property:

```
Bm1.section = W200X100;
```

- Get property:

```
Print(Bm1.section.flangeThickness());
```

- Use copy method:

```
Bm11 = Bm1.copyTranslate(Vector3d(0 m,0 m,1 m));
```

- Setting and looking up properties

```
Bm1.setEndOffset(2,Bm2.localSystem.xVector.normalise()*(-1.3));
```

Basic JScript within GeniE commands

- The GeniE command interpreter will recognize
  - basic GeniE commands
  - basic JScript commands
  - a mix of GeniE and JScript commands
- Examples

```
Bm114 = Bm94.copyTranslate
(Vector3d( 2.3 m , 0.3 m ,10 m));

Bm114 = Bm94.copyTranslate
(Vector3d( 2.3 ,L1 + 0.3 m , Bm2.end1.x()/4));

Bm115 = Beam(Point1,Point2);

Bm115 = Beam(Point(Math.abs(Math.pow (Bm62.end1.y.toDouble(), 2)),H1,1200
mm),Bm62.end2());
```

Units

When the interpreter finds a number not followed by an operator it will interprete the next word as a unit.
The number will get type from the unit.

GeniE objects names

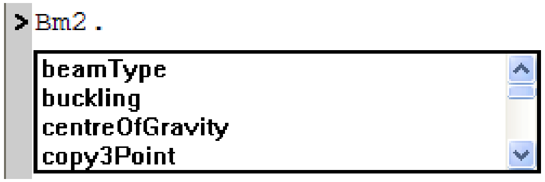
- Object names versus variable names

```
// Creating beams in a loop…
for (i = 1;i<(Floors+1); i++) {
// Get a point by it’s name
End1 = GetNamedObject(“Point"+i);
   End2 = GetNamedObject(“Point"+(i+1));
   aBeam = Beam(End1,End2);
   // The beam name is now the same as the variable “aBeam”
   aBeam.section = I600;
   aBeam.material = St44;
// Give the beam a unique name "Leg"+i
   Rename(aBeam, "Leg"+i);
   Print (GetNamedObject("Leg"+i).section.name());
Print (Leg1.section.name()); // but not Print(“Leg”+i...  }
```

Note, we can not type

“Leg”+i = Beam(End1,End2);

Exercise - objects and properties



```
> Bm2.
```

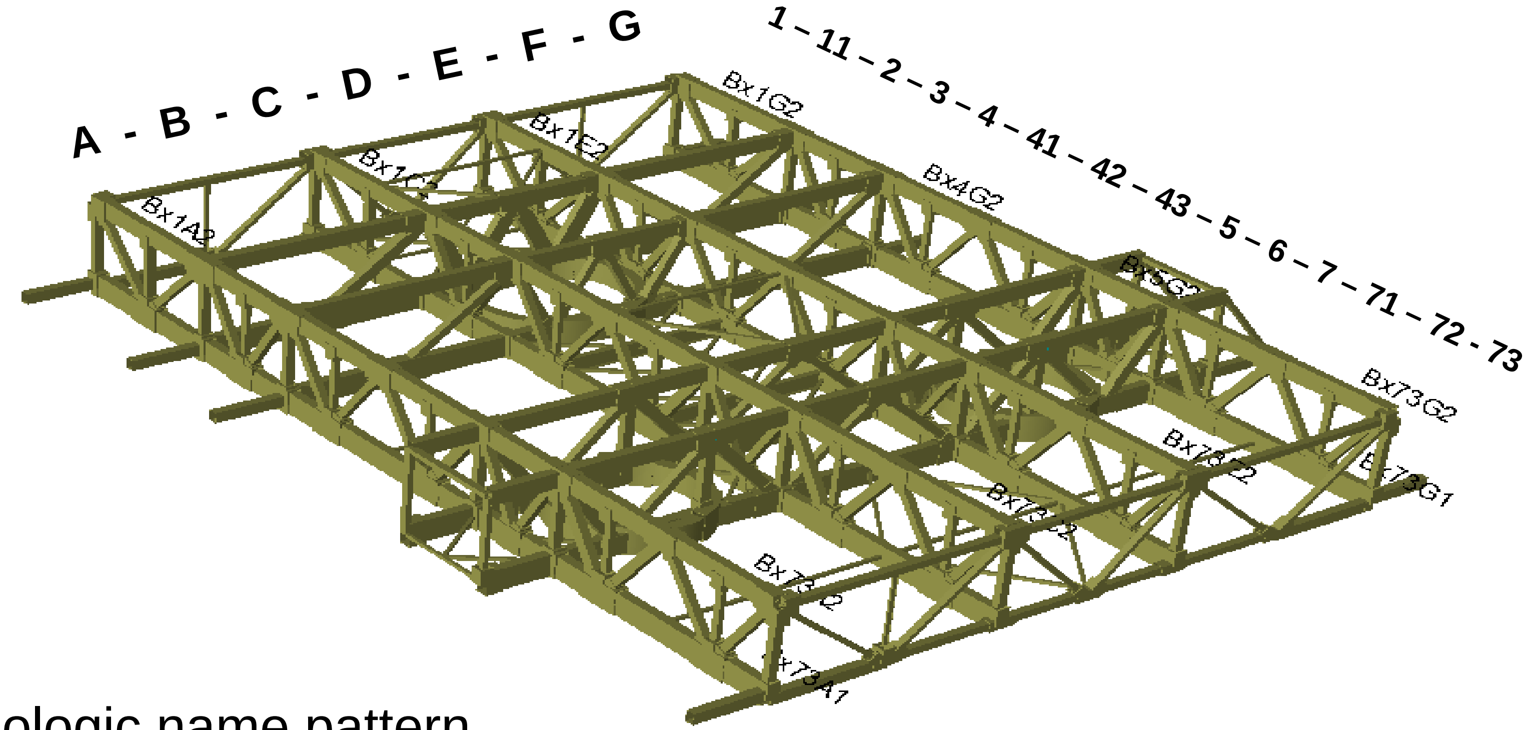
- Read in the frame.js
- Type in the command window: Bm2. - and press the Tab key
- Study the properties and functions of Bm2
- Look up the StraigthBeam in Help - Class hierarchy:
- ModelObject
  - NamedObject
    - Transformable
      - BasicConcept
        - BasicBeam
          - StraightBeam
- Print the cross section of Bm2
  - Print(Bm2.section.diameter());
- Set end1 x-offset of Bm2 equal half the diameter of Bm3

  Bm2.setEndOffset(1,Vector3d(Bm3.section.diameter()/2.0,0.0,0.0));

Accessing the objects in a model

- You can access objects by
  - by name
    - Genie has no name pattern suitable for programming
    - You can define your own
  - by name sets
    - Static sets
    - Dynamic sets
    - In scripted input it is easy to add objects to sets as they are created
  - by some of it’s properties
    - Limit… functions
    - SupportType function

A naming pattern for structures



- Topologic name pattern
- Name increments in X,Y and Z direction
  - Beam Bx4G2
  - Beam in X direction at X-axis 4, Y-axis G, Z-axis 2
  - Type + X-name + Y-name + Z-name

Topologic name pattern

- Make a name from four sub strings Nam1, Nam2, Nam3 & Nam4 denoting:

Object type	X-direction	Y-direction	Z-direction
Nam1	Nam2	Nam3	Nam4

- Decide on a notation for type and make arrays of names in X,Y and Z direction

Nam1 = "P" // Point	**Nam2[0] = "1";**	**Nam3[0] = "A";**	**Nam4[0] = "1";**
Nam1 = "Bx" // Beam in Xdir	**Nam2[1] = ”11";**	**Nam3[1] = "B";**	**Nam4[1] = "2";**
Nam1 = "By" // Beam in Ydir	**Nam2[2] = ”2";**	**Nam3[2] = "C";**	**Nam4[2] = "3";**
Nam1 = "Bz" // Beam in Zdir	**Nam2[3] = ”3";**	**Nam3[3] = "D";**	**Nam4[3] = "4";**
Nam1 = "Dx" // Beam diagonal in XZdir	**Nam2[4] = ”4";**	**Nam3[4] = "E";**	**Nam4[4] = "5";**
Nam1 = "Dy" // Beam diagonal in YZdir	**Nam2[5] = ”41";**	**Nam3[5] = "F";**	**Nam4[5] = "6";**
....	**Nam2[6] = ”42";**	**Nam3[6] = "G";**	**Nam4[6] = "7";**
	Nam2[7] = ”43";	**Nam3[7] = "H";**	**Nam4[7] = "8";**
	Nam2[8] = ”5";	**Nam3[8] = "I";**	**Nam4[8] = "9";**
	Nam2[9] = ”6";	**Nam3[9] = "J";**	

- Now you can build a name; ”By” + Nam2[5] + Nam3[2] + Nam4[1] => ”By41C2”

Topologic name pattern

- Make a function to build a name from the sub strings
  - function GetName (N1,N2,N3,N4) { return(N1 +Nam2[N2] +Nam3[N3] +Nam4[N4]); }
- And a function to get an object by its name:
  - function GetObjectByName (N1,N2,N3,N4) { return(GetNamedObject(GetName(N1,N2,N3,N4))); }
- Now you have a naming pattern suitable for programming:

```
for (i = 0;i< NumX ; i++)  {
   for (j = 0;j< NumY ; j++) {
          for (k = 0; k < NumZ ; k++)  {
                    var Bms1 = new Array();
                    Bms1[i] = Beam(GetObjectByName("P",i,j,k),
                    GetObjectByName("P",i+1,j,k));
                    Bms1[i].name = GetName("BmX",i,j,k);
                    ........
```

Dynamic sets

- A Dynamic set uses a query to identify its members.
- Dynamic sets are continously updated when new objects are created
- Example: All objects in a Plane:

  ```
  dset1=DynamicSet(LimitInPlane(ZPlane3d(0)));
  ```

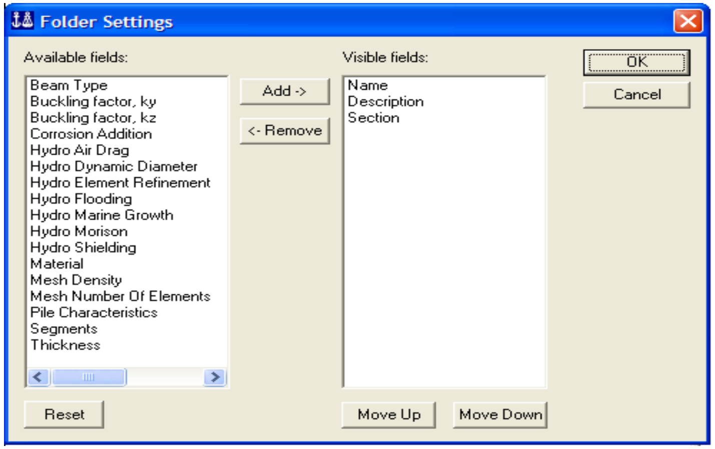
- Example:
  Set comprising beams in plane Z=0.0
  with flange Width < 0.2 m

  ```
  dset1=DynamicSet(LimitInPlane(ZPlane3d(0)) && LimitLower("Width",0.2m));
  ```

- You can query Genie objects about their properties

  LimitLower("Width",0.2m) is a query for Beams with section flange Width < 0.2 m

- The properties you can query are listed in the Fields dialogue for the object type in the browser. E-g. for Structure:



Available fields:

Beam Type
Buckling factor, ky
Buckling factor, kz
Corrosion Addition
Hydro Air Drag
Hydro Dynamic Diameter
Hydro Element Refinement
Hydro Flooding
Hydro Marine Growth
Hydro Morison
Hydro Shielding
Material
Mesh Density
Mesh Number Of Elements
Pile Characteristics
Segments
Thickness

Visible fields:

Name
Description
Section

Dynamic sets - more examples

- Set comprising objects along a line that also belongs to a named set:
  dset2 = DynamicSet(LimitAnd(LimitInSet(Frame_A),LimitLine(Point4,Point7)));
- Set comprising beams in plane Z=0.0 with Section I200:
  dset1=DynamicSet(LimitInPlane(ZPlane3d(0)) && LimitString("Section","I200",true));
- dset1 = DynamicSet(LimitLower("Diameter",0.7m));
- dset1 = DynamicSet(LimitString("Name","Bm.*",true));
  When the boolean is true LimitString supports regular expression e.g:

  .   => (dot) Matches any single character
  *   => Repeats the previous item zero or more times
  | => Match the part on either side of the |
  ?   => Makes the preceding item optional
- Examples:
  - Bm.*   => Matches all objects named Bm<something>
  - Bm7|Bm7.* or Bm7.*?   => Matches Bm7, Bm72, Bm78, Bm786 etc.

Then use the set you created...

dset1.setBeamOffset(Vector3d(0 m,0 m,.3 m));

- Objects in dset1 that do not support setBeamOffset will be ignored.

Exercise - Limit-functions

- Read the frame.js
- Use Tab in the command window to see all variants of the Limit… functions
- Or in GeniE help…

  Class hierarchy:

  ModelObject

  + NamedObject

    + AbstractLimit

  Direct Known Subclasses:

  LimitAnd ,LimitBox ,LimitInSet ,LimitLine , LimitNot ,LimitNumber , LimitOr ,LimitPlane ,LimitString +++

- Use the Limit... functions to extract various subsets of the frame model

Sort beams in sets based on section

- Put beams in named sets based on the section name:

```
for(var sect in ModelObjects) {     if(sect.supportsType(typeSection)) {
              newSet=Set();
              for(var bm in ModelObjects) {
                    if (bm.supportsType(TypeStraightBeam) ||
                    bm.supportsType(TypeCurvedBeam)) {
                             if(bm.section.name==sect.name) {
                                      newSet.add(bm);
                            }   } }
           newSet.name="Set_"+sect.name;
           }   }
```

- or accessing the beams directly...

```
  for(var a in ModelObjects)
        { if(a.supportsType(typeStraightBeam))
              Print(a.Length); }
```

Exercise - GeniE beam objects

- Read the frame.js journal
- Print the section name of beams in the command window by
  - Name – GetNamedObject
    - When to use GetNamedObject and when to use name directly?
  - Elevation – LimitInPlane
  - Objects that have cross section – if(thisObject.supportsType(typeSection))
  - Cross section = OD610X8
- Hint:

```
for(var a in ModelObjects)
  { if(a.supportsType(typeStraightBeam) && a.section.name=="OD610X8")
  Print(a.name()+"  "+a.section.name());}
```

- Use the Vector3D to find the distance between two beam ends

```
Print(Vector3D(Point1,Point2).length());
```

Controlling program flow

- If …else

Example; Creating braces in the jacket wizard

```
for (i = 0;  i < (NrxBrace);   i++)  {   Thisbrace = xBrace[i];
          if (Thisbrace == "braceup")  { BraceUpX(i); }
         else if (Thisbrace == "bracedown") { BraceDownX(i); }
         else if (Thisbrace == "xbrace") { BraceUpX(i); BraceDownX(i); }
         else if (Thisbrace == "kbrace") { Print("K-Brace not implemented"); }
 }
// xBrace[i] is an array of the braces to be created
// BraceUpX(i), BraceDownX(i) etc. are functions that create braces
```

do…while

- do …while

```
var i = 1;
do  {
   Print( GetNamedObject("Bm"+i).end1.z);
   i++;
} while (i < 10);
```

for…

```
// Create an array of points on a circle
var myPoints = new Array(NrPoints);
var NrPoints = 36;     // Number of points
var Rad = 10.0;        // Radius
var X,Y;               // X,Y coordinate
//
for(var icount = 0; icount < NrPoints; icount++)
{ X=Rad*Math.cos(icount*10*Math.PI/180);
  Y=Rad*Math.sin(icount*10*Math.PI/180);
  myPoints[icount]=Point(X,Y,0);
  myPoints[icount].name="Point"+icount;
 }
```

for…

```
// ( GetObjectByName and GetName are user defined functions)
// Creating beams
// Stepping in X, Y and Z direction
     for (i = 0;i< NumX-1 ; i++) {
       for (j = 0;j< NumY-1 ; j++)    {
          for (k = 0; k < NumZ-1 ; k++) {
                    var aBeam ;
            aBeam = Beam(GetObjectByName(0,i,j,k), GetObjectByName(0,i+1,j,k));
             aBeam.section = Box1400;
            aBeam.material = S275;
            Rename(aBeam,GetName(1,i,j,k));
       }
    }
}
```

for…in

- Set X-coordinate of some points

```
var MyPoints = new Array();
MyPoints[0] = Point(0, 0, 1.0);
MyPoints[1] = Point(0, 0, 2.0);
MyPoints[2] = Point(0, 0, 3.0);
//
for (var MyPoint in MyPoints)  { MyPoint.x = 2m; }
```

- Print length of all straight beams

```
for(var a in ModelObjects)
    { if(a.supportsType(typeStraightBeam))
        Print(a.Length); }
```

Functions

- JScript built-in functions

  GetObject

  ```
  MyBook = GetObject(“C:\\TestDir\\TestBook.xls");
  Elevation1 = Length(MyBook.Worksheets ("sheet1").Range("C8").Value);
  ```

  -Works for COM objects e.g. Microsoft Office documents

- GeniE Functions

  ```
  End1 = GetNamedObject("MainLegP"+i);
  ReadCommandFile(“Input.js");
  ```

  Plus many more…

  Most GeniE commands are functions that the various objects support

  e.g. the copyTranslate function for a plate

  ```
  Pl5 = Pl1.copyTranslate(Vector3d(2 m,0 m,6.23 m));
  ```

User defined functions

- Example:

```
// Function to transform a vector in local coordinate system to global coordinates
function VecTra(LocX, LocY, LocZ, LocalCoordSys)
    // Input is local vector X,Y,Z and localsystem
    // Returns global X,Y,Z vector
    {
    return GlobalVector = LengthVector3D(
        LocalCoordSys.xVector.normalise()*LocX +
        LocalCoordSys.yVector.normalise()*LocY+
        LocalCoordSys.zVector.normalise()*LocZ);
    }

// Example - move beam end in beams local coordinates
Bm1 = Beam(Point(2.5 m,0 m,0 m), Point(7.5 m,7.5 m,0 m));
MyCord = Bm1.localSystem();
Bm1.setEndOffset(2,VecTra(1 m ,1 m ,0 m , MyCord));
```

More user defined functions

```
// Function to make a name by adding four strings from
// name arrays Nam2[ ], Nam3[ ], Nam4[ ],
function GetName (N1,N2,N3,N4) { return(N1+Nam2[N2] +Nam3[N3] +Nam4[N4]); }

// then use GetName to name some points
function MakePoints (NumX, NumY, NumZ, Step)
{       var i, j, k, aPoint;   // Stepping in X, Y and Z direction
         for (i = 0;i< NumX ; i++)
         {  for (j = 0;j< NumY; j++)
             { for (k = 0; k < NumZ; k++)
                   {    X1 = i * Step;
                        Y1 = j * Step;
                        Z1 = k * 0.6 * Step;
                        aPoint = Point(X1, Y1, Z1);
                        aPoint.name = GetName(“Point”,i,j,k);
   } } } }
```

JScript objects – the Math object

- JScript Math object - methods

  abs , acos , asin , atan , atan2 , ceil , cos , exp , floor , log , max , min , pow , random , round , sin , sqrt , tan

- Example:

  Xcoord = (-1)*Math.abs(Math.sqrt(Math.abs(Math.pow(Radius1.toDouble()),2) - Math.pow(Length(YCord-Elev2).toDouble() ,2)))) + Length(Radius1 + WidePlat).toDouble;

  if your variable is of type “Length” you must use toDouble when you use it as input to the Math functions.

  Length(myMeasure) - converts to length

  myMeasure.toDouble – converts to number

Working with Office documents from GeniE

```
MyBook = GetObject("C:\\WorkSpace\\MyWorkbook.xls");
// Note regexp escape sequence \\ to get \ in path
MyBook.Worksheets("sheet1").Activate;
MyBook.Windows(1).Visible="True";
//      Writing an Array to EXCEL
var MyData = new Array();
MyData[1] = 100;
MyData[2] = 200;
MyData[3] = 300;
MyData[4] = 400;
NumberOfData  = MyData.length;
```

Working with Office documents from GeniE

```
// Write data from GeniE to a Excel sheet
Row = 5;
Col = 3;
//
for (i = 1;i< NumberOfData; i++)
    {
    MyBook.ActiveSheet.Cells((Row-1),Col).Offset(i,0).Value = MyData[i];
    }
//
// Read some data back to GeniE
aText = MyBook.Worksheets("sheet1").Cells(7,5).value;
```

User defined JScript objects

MANAGING RISK DNV

- Defining a Frame class with a function to calculate area

```
class Frame {
        constructor function Frame(w,h) {width=w;height=h;}
        function frame_area() {return width*height;}
        var width;
        var height;
            }


frame1 = new Frame(4.0, 6.0);

Print(frame1.frame_area());

-> 24.000000
```

User defined objects - Point and Line class

- Example:
  A class for a line object in two dimensions with a function to find the point of intersection with another line

```
// Define a class for a point in the X,Y plane
class Point2d {
  constructor function Point2d(x,y) {m_x=Length(x);m_y=Length(y);}
  function X() {return m_x;}
  function Y() {return m_y;}
  var m_x,m_y;
       }
```

and the Line class…

```
class Line2d {
   var m_start,m_end;
   constructor function Line2d(start,end) {m_start=start;m_end=end;}
   function Start() {return m_start;}
   function End() {return m_end;}
   function intersect(other) {
    ua1 = (other.End().X() - other.Start().X()) * (Start().Y() - other.Start().Y()) - (other.End().Y() -
     other.Start().Y()) * (Start().X() - other.Start().X());
    ua2 = (other.End().Y() - other.Start().Y()) * (End().X() - Start().X()) - (other.End().X() -
     other.Start().X()) * (End().Y() - Start().Y());
    ua = ua1 / ua2;
    x = Start().X() + ua * (End().X() - Start().X());
    y = Start().Y() + ua * (End().Y() - Start().Y());
    return new Point2d(x,y);
   }
}
```

Creating two line objects...

```
// Creating two line objects and a point at the intersection

l1=new Line2d(new Point2d(0,10m),new Point2d(5m,0));

l2=new Line2d(new Point2d(10m,0),new Point2d(0,5m));

p=l1.intersect(l2);
```

Generating reports

- To get a template for report journal do
  - File > Create Report
  - and make sure “Journal Report Generation” is checked
  - This will give you something like:

```
Case_11.add(ChapterStructure());
Case_11.element(1).add(TableBeamCoordinate());
Case_11.element(1).add(TableBeamProperty());
Case_11.element(1).add(TableBeamHydroProperty());
Case_11.element(1).add(TableBeamEccentricity());
```

Adding figures to a report

```
//  Make a function plotResult to plot VonMises stresses for "loadcasein“ on a set "setin"
// returns a plot Figure object
       function plotResult(loadcasein,setin) {
           // Delete "Demo_ModelView" if it exists
           for(var object in ModelObjects) {
              if(object.supportsType(typeModelView) && object.name()=="Demo_ModelView")
               { Delete(object); }
            }
    ModelView_temp = ModelView();
    ModelView_temp.addElement(DisplayConfiguration    ("Results - with Mesh", moPaper));
    ModelView_temp.addElement(ResultPresentation());
    ModelView_temp.resultPresentation.resultComponent = rsStress;
    ModelView_temp.resultPresentation.calculationType = rsVonMises;
// Continue...
```

Reports - continued…

```
ModelView_temp.resultPresentation.optionMinmax = false;
ModelView_temp.resultPresentation.optionValues = false;
ModelView_temp.name =  "Demo_ModelView";
ModelView_temp.addElement(VisibleModel());
myloadcase = GetNamedObject(loadcasein.name);
myloadcase.setCurrent();
if( setin.supportsType(typeSet)) {
        Demo_ModelView.visibleModel.include(setin);
        Demo_ModelView.activate();   }
Graphics.fitModel();
var Plottitle = "Loadcase " + loadcasein.name() + " - Part " + setin.name() + " - Von Mises Stresses";
var Plotfile = loadcasein.name() + setin.name() + ".jpg";
Graphics.saveImage(Plotfile);
var NewPlot = Figure(Plottitle , Plotfile);
return NewPlot;
 }
```

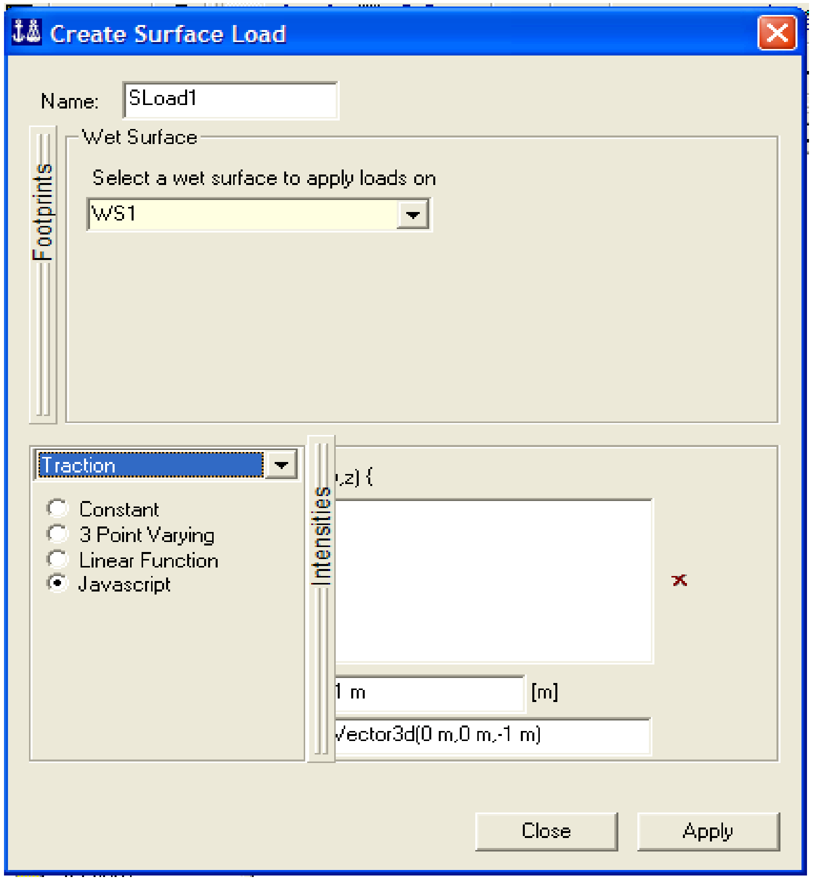
Using the function plotResult

```
//   Using the function plotResult to make a report
//
Case_1 = Report("Case_1");
Case_1.add(ChapterStructure());
Case_1.element(1).add(TablePlateCoordinate());
Case_1.element(1).add(TablePlateProperty());
Case_1.element(1).add(TableSupportBoundary());
Case_1.add(plotResult(Operation,UpperDeck));
Case_1.add(plotResult(Storm,UpperDeck));
Case_1.saveAs("Case_1.doc", mrWordXML);
```

- Once you have the predefined plot functions adding plots to the report is easy

Load Intensity by JScript

- The intensity of line and surface loads may be given by a JScript function
- E.g. GeniE menu; Insert > Explicit Load > Surface Load...



Create Surface Load

Name: SLoad1

Wet Surface

Select a wet surface to apply loads on: WS1

Traction

○ Constant
○ 3 Point Varying
○ Linear Function
● Javascript

,z) {

1 m [m]

Vector3d(0 m,0 m,-1 m)

- The dialogue takes a JScript function as input
- The function for surface loads may use the current x, y & z coordnate as input
- pressureFunction(x y z)
  { return *your_function*...}

- The function for line loads may use the relative length along the line as input
- component1Dfunction(param)
  { return *your_function*...}

Load Intensity by JScript - Examples

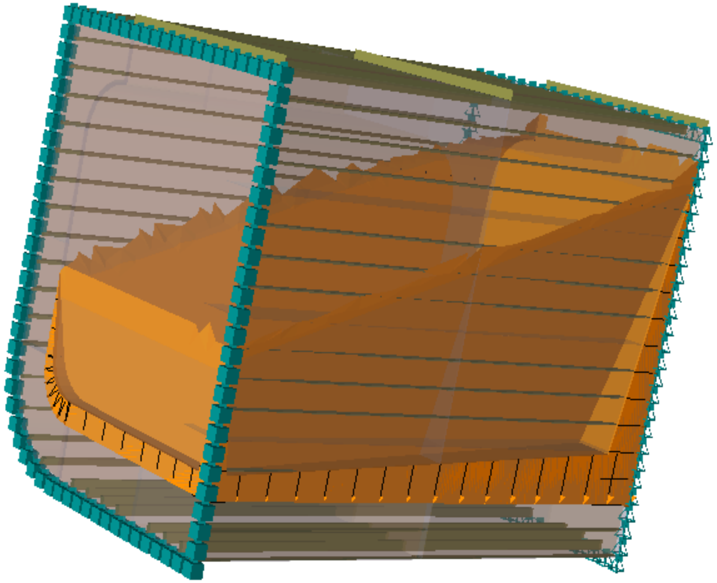
- Line Component Load:
  return Vector3d(1N/m,x*2N/m2,0);
  - //input x,y,z, all of type Length
  - //output type Vector3d(ForcePerLength,ForcePerLength,ForcePerLength)
- Line Component Load - Parametric:
  return Vector3d(1N/m,param*2N/m,0);
  - //input t, of type Number (unitless)
  - //output type Vector3d(ForcePerLength,ForcePerLength,ForcePerLength)
- Line Temperature Load:
  return 1delC*y.toDouble();
  - //input x,y,z, all of type Length
  - //output TempDiff
- Line Temperature Load - Parametric:
  return param*10delC;
  - //input t, of type Number (unitless)
  - //output type TempDiff

Load Intensity by JScript - Examples

- Surface Pressure:
  return z*1 G * 1025 kg/m3;
  - //input x,y,z, all of type Length
  - //output type ForcePerArea
- Surface Component Load:
  return Vector3d(1N/m2,x*2N/m3,0);
  - //input x,y,z, all of type Length
  - //output type Vector3d(ForcePerArea,ForcePerArea,ForcePerArea)
- Surface Traction - Example function:
  - return 15N/m2*Math.cos(x.toDouble());
  - //input x,y,z, all of type Length
  - //output type ForcePerArea

Load Intensity by JScript - Example

```
/* Function for generating a hydrostatic pressure in a compartment for
    20 deg. heel about x-axis and 4.5 m draught at origin
 - Define a wet surface.
 - Insert a pressure load on the wet surface
 - Set intensity to Javascript and paste the script below into the script window in the
    pressure dialogue
*/
HeelAngle = 20 ;
Draught = 4.5 m;
//
HeelAngleRad =(HeelAngle*Math.PI)/180;
Zlocal=(z*Math.cos(HeelAngleRad)) -
    (y*Math.sin(HeelAngleRad));
if ( Zlocal < Draught)
{
return 1 G * 1025 kg/m3 * (Draught - Zlocal);
}
else return 0;
```



Some limitations

- GeniE has limited JScript functions for object name handling
  (from Bm3 to Bm34 step 3)
  Automatic naming of new objects (next beam-name)
- JScript does not support file operations
  - but you can read and write to Excel etc.
- Custom dialogues and buttons
  - May be available in future versions
  - New .net based scripting is under evaluation
- Read input data to user functions from GeniE by cursor selection

www.dnv.com