DNV Software Products : Genie Illustrated Report, with functions and loops

This page last changed on nov 26, 2009 by atwe.

```
//
// This prototype script produces different plots for ALL sets and ALL loadcases in a model and
 adds them to a report.
//
// The script may be edited to fit your own purposes.
//
// The following functions are defined and used within the JS file.
//   box_beams      - This functions finds and returns the bounding box of all the straight beams
 in the named set being passed into the function
//   view_vectors   - This function determines the normal vector and local x-vector based on the
 geometry on the bounding box being passed into the function
//   plotConcepts   - This function plots concepts with labelling and/or color coding (possibly
 with conceptual loads for a given loadcase)
//   plotMesh       - This function plots FEM mesh with labelling and/or color coding (possibly
 with FEM loads for a given loadcase)
//   plotResult     - This function plots FEM vonMises stress results for plate elements for a
 given loadcase
//   plotCapacity   - This function plots capacity modles with labelling and/or color coding (for a
 given loadcase or for worst loadcase)
//
// The following issues should be noted:
// - The script may become inefficient for large models.
// - The bounding box and normal vector is being calculated based on the straight beams in a set.
 If a set contains no straight beams, the view will default to isometric.
// - In the capacity model, labelling of UfTot does not work, unless sepcified by all capacity
 model names.
// - The vonMises Stress result plots only show stresses in plates. If a set only contains beams,
 nothing is showed. Beam force reporting is not avaiable.
//
//                    Atle Westvang / Petter Blindheim, DNV Software  2. Feb 2009
//
function bbox_beams(set_to_find_bbox)
{
    //This functions finds and returns the bounding box of all the straight beams in the named set
 being passed into the function

    var bboxXmax = 0;
    var bboxYmax = 0;
    var bboxZmax = 0;
    var bboxXmin = 0;
    var bboxYmin = 0;
    var bboxZmin = 0;

    if( set_to_find_bbox.supportsType(typeSet)) {

        //using one end of the first beam in the set to find the initital bounding box
        for (var objectInSet in set_to_find_bbox){
           if(objectInSet.supportsType(typeStraightBeam)){
                bboxXmax = parseFloat(objectInSet.end1.x());
                bboxYmax = parseFloat(objectInSet.end1.y());
                bboxZmax = parseFloat(objectInSet.end1.z());
                bboxXmin = parseFloat(objectInSet.end1.x());
                bboxYmin = parseFloat(objectInSet.end1.y());
                bboxZmin = parseFloat(objectInSet.end1.z());
                break;
            }
        }
        //Now looping through all beams in set
        for (var objectInSet in set_to_find_bbox){
            if(objectInSet.supportsType(typeStraightBeam)){
```

```
                bboxXmin
 = Math.min(parseFloat(objectInSet.end1.x()),parseFloat(objectInSet.end2.x()),bboxXmin);
                bboxYmin
 = Math.min(parseFloat(objectInSet.end1.y()),parseFloat(objectInSet.end2.y()),bboxYmin);
                bboxZmin
 = Math.min(parseFloat(objectInSet.end1.z()),parseFloat(objectInSet.end2.z()),bboxZmin);

                bboxXmax
 = Math.max(parseFloat(objectInSet.end1.x()),parseFloat(objectInSet.end2.x()),bboxXmax);
                bboxYmax
 = Math.max(parseFloat(objectInSet.end1.y()),parseFloat(objectInSet.end2.y()),bboxYmax);
                bboxZmax
 = Math.max(parseFloat(objectInSet.end1.z()),parseFloat(objectInSet.end2.z()),bboxZmax);
            }
        }
    }
    var returnArray = new Array(bboxXmin,bboxYmin,bboxZmin,bboxXmax,bboxYmax,bboxZmax);
    return returnArray;
} // function bbox_beams

function view_vectors(bbox_array)
{
    //This function determines the normal vector and local x-vector based on the geometry on the
 bounding box being passed into the function

    var bboxXmin = bbox_array[0];
    var bboxYmin = bbox_array[1];
    var bboxZmin = bbox_array[2];
    var bboxXmax = bbox_array[3];
    var bboxYmax = bbox_array[4];
    var bboxZmax = bbox_array[5];

    //Creating a normal for the set and displaying it
    var dx = bboxXmax - bboxXmin;
    var dy = bboxYmax - bboxYmin;
    var dz = bboxZmax - bboxZmin;

    var fac=0.1;
    var locX = Vector3d(-1,1,-1);
    var normVec = Vector3d(0,0,1);

    //in xy-plane, view towards x-axis:
    if ( (dz<fac*dx) && (dz<fac*dy) ){
        locX = Vector3d(0,0,-1);
        normVec = Vector3d(0,1,0);
    }
    //in xz-plane, view along y-axis
    else if ( (dy<fac*dx) && (dy<fac*dz) ){
        locX = Vector3d(0,1,0);
        normVec = Vector3d(0,0,1);
    }
    //in yz-plane, view towards x-axis
    else if ( (dx<fac*dy) && (dx<fac*dz) ){
        locX = Vector3d(-1,0,0);
        normVec = Vector3d(0,0,1);
    }
    var returnArray = new Array(locX,normVec);
    return returnArray;
} // function view_vectors

function plotConcepts(areport,plotset,view,prop,lcname,label_choice,colorcode_choice)
{
//   plotConcepts   - This function plots concepts with labelling and/or color coding (possibly
 with conceptual loads for a given loadcase)

    var bbox_array = bbox_beams(plotset);
```

```
    var viewvector_array = view_vectors(bbox_array);

    var locX    = viewvector_array[0];
    var normVec = viewvector_array[1];

    ModelView_temp = ModelView(Point(0 m,0 m,0 m), locX, normVec);
    ModelView_temp.addElement(DisplayConfiguration(view, moPaper));
    ModelView_temp.addElement(VisibleModel());
    if( plotset.supportsType(typeSet)) {
        ModelView_temp.visibleModel.include(plotset);
        plotset_name = plotset.name;
    }
    else {
        ModelView_temp.visibleModel.showAll = true;
        plotset_name = "All";
    }

    if( colorcode_choice ) {
        ModelView_temp.addElement(ColorCode("Properties", prop));
    }
    else {
        ModelView_temp.addElement(ColorCode());
    }

    ModelView_temp.addElement(Labels());
    if( label_choice && plotset.supportsType(typeSet) ) {
       ModelView_temp.labels.addLabel(plotset,prop);
       ModelView_temp.labels.smartPosition = true;
       ModelView_temp.labels.orientedText = true;
    }

    ModelView_temp.activate();

    var newname =  plotset_name + "_" + prop + "_" + lcname;

    var filename = newname + ".png";
    Graphics.fitModel();
    Graphics.saveImage(filename,hsize,vsize);

    var title = plotset_name + " : " + prop;
    areport.back.back.add( Chapter(title) );
    areport.back.back.back.add( Figure(newname,filename) );

    Delete(ModelView_temp);

} //plotConcepts

function plotCapacity(areport,plotset,view,res,lcname,label_choice,colorcode_choice)
{
//   plotCapacity   - This function plots capacity modles with labelling and/or color coding (for a
 given loadcase or for worst loadcase)

    var bbox_array = bbox_beams(plotset);
    var viewvector_array = view_vectors(bbox_array);

    var locX    = viewvector_array[0];
    var normVec = viewvector_array[1];

    setWorstCodeCheckCase();
    ModelView_temp = ModelView(Point(0 m,0 m,0 m), locX, normVec);

    ModelView_temp.addElement(DisplayConfiguration(view, moPaper));
    ModelView_temp.addElement(VisibleModel());
    if( plotset.supportsType(typeSet)) {
        ModelView_temp.visibleModel.include(plotset);
        plotset_name = plotset.name;
```

```
    }
    else {
        ModelView_temp.visibleModel.showAll = true;
        plotset_name = "All";
    }
    ModelView_temp.addElement(ResultPresentation());


    ModelView_temp.addElement(Labels());
    if( colorcode_choice ) {
        ModelView_temp.addElement(ColorCode("Results", res));
    }
    else {
        ModelView_temp.addElement(ColorCode());
    }

    ModelView_temp.activate();

    var newname =  plotset.name + "_Results_" + res + "_" + lcname;

    var filename = newname + ".png";
    Graphics.fitModel();
    Graphics.saveImage(filename,hsize,vsize);

    var title = plotset.name + " : " + res;
    areport.back.back.add( Chapter(title) );
    areport.back.back.back.add( Figure(newname,filename) );

    Delete(ModelView_temp);

} //plotCapacity


function plotMesh(areport,plotset,view,prop,lcname,label_choice,colorcode_choice)
{
//   plotMesh       - This function plots FEM mesh with labelling and/or color coding (possibly
 with FEM loads for a given loadcase)

    var bbox_array = bbox_beams(plotset);
    var viewvector_array = view_vectors(bbox_array);

    var locX    = viewvector_array[0];
    var normVec = viewvector_array[1];

    ModelView_temp = ModelView(Point(0 m,0 m,0 m), locX, normVec);
    ModelView_temp.addElement(DisplayConfiguration(view, moPaper));
    ModelView_temp.addElement(VisibleModel());
    if( plotset.supportsType(typeSet)) {
        ModelView_temp.visibleModel.include(plotset);
        plotset_name = plotset.name;
    }
    else {
        ModelView_temp.visibleModel.showAll = true;
        plotset_name = "All";
    }

    if( colorcode_choice ) {
        ModelView_temp.addElement(ColorCode("Mesh", prop));
    }
    else {
        ModelView_temp.addElement(ColorCode());
    }

    ModelView_temp.addElement(Labels());
    if( label_choice ) {
       ModelView_temp.labels.addLabel(plotset,prop);
```

```
        ModelView_temp.labels.smartPosition = true;
        ModelView_temp.labels.orientedText = true;
    }

    ModelView_temp.activate();

    var newname =  plotset_name + "_Mesh_" + lcname + "_" +  prop;

    var filename = newname + ".png";
    Graphics.fitModel();
    Graphics.saveImage(filename,hsize,vsize);

    var title = plotset_name + " : " + prop;
    areport.back.back.add( Chapter(title) );
    areport.back.back.back.add( Figure(newname,filename) );

    Delete(ModelView_temp);

} //plotMesh

function plotResult(areport,plotset,view,lcname,label_choice,colorcode_choice)
{
//   plotResult     - This function plots FEM vonMises stress results for plate elements for a
 given loadcase

    var bbox_array = bbox_beams(plotset);
    var viewvector_array = view_vectors(bbox_array);

    var locX    = viewvector_array[0];
    var normVec = viewvector_array[1];

    ModelView_temp = ModelView(Point(0 m,0 m,0 m), locX, normVec);
    ModelView_temp.addElement(DisplayConfiguration(view, moPaper));
    ModelView_temp.addElement(VisibleModel());
    if( plotset.supportsType(typeSet)) {
        ModelView_temp.visibleModel.include(plotset);
        plotset_name = plotset.name;
    }
    else {
        ModelView_temp.visibleModel.showAll = true;
        plotset_name = "All";
    }

    ModelView_temp.addElement(ColorCode());
    ModelView_temp.addElement(Labels());

    ModelView_temp.addElement(ResultPresentation());
    ModelView_temp.resultPresentation.resultComponent = rsStress;
    ModelView_temp.resultPresentation.calculationType = rsVonMises;
    ModelView_temp.resultPresentation.surfaceId = rsSurfaceMiddle;
    ModelView_temp.resultPresentation.drawAlgorithm = rsContour;
    ModelView_temp.resultPresentation.optionMinmax = false;

    ModelView_temp.resultPresentation.optionValues = true;
    ModelView_temp.resultPresentation.option3dContours = true;
    ModelView_temp.activate();

    var newname =  plotset_name + "_Results_" + lcname + "_VonMises";

    var filename = newname + ".png";
    Graphics.fitModel();
    Graphics.saveImage(filename,hsize,vsize);

    var title = plotset_name + " : " + lcname;
    areport.back.back.add( Chapter(title) );
    areport.back.back.back.add( Figure(newname,filename) );
```

```
    Delete(ModelView_temp);

} //plotResult



//Size of graphics window, aspect ratio must be the same as the Genie graphics window - or the
 aspect ratio will become wrong on the plots
var hsize = 800;
var vsize = 600;


// THIS WILL DESTROY ALL MODEL VIEWS !!!
// Delete all existing model views
for(var object in ModelObjects) {

   // Limit the search to named model views
   if(object.supportsType(typeModelView)) {
      Delete(object);
   }
}

//Creating a report to use for displaying our resulting plots

areport = Report("Demo Report");
var main_title = "Named sets";
areport.add( Chapter(main_title) );

SetNoLoadcase();

areport.back.add( Chapter("Default view of whole model") );

plotConcepts(areport,areport,"Default display","None","None",false,false);

areport.back.add( Chapter("Default view of named sets") );


for(var object in ModelObjects) {

    //  Limit the search to named sets
    if(object.supportsType(typeSet)) {
        plotConcepts(areport,object,"Default display","None","None",false,false);
    }
}

areport.back.add( Chapter("Material view of named sets") );

for(var object in ModelObjects) {

    //  Limit the search to named sets
    if(object.supportsType(typeSet)) {
        plotConcepts(areport,object,"Default display","Material","None",true,true);
    }
}

areport.back.add( Chapter("Section view of named sets") );

for(var object in ModelObjects) {

    //  Limit the search to named sets
    if(object.supportsType(typeSet)) {
        plotConcepts(areport,object,"Default display","Section","None",true,true);
    }
}
```

```
areport.back.add( Chapter("Thickness view of named sets") );

for(var object in ModelObjects) {

    //  Limit the search to named sets
    if(object.supportsType(typeSet)) {
        plotConcepts(areport,object,"Default display","Thickness","None",true,true);
    }
}


areport.back.add( Chapter("Mesh view with thickness of named sets") );
for(var object in ModelObjects) {

    //  Limit the search to named sets
    if(object.supportsType(typeSet)) {
        plotMesh(areport,object,"Mesh - All","Thickness","None",true,true);
    }
}



//Looping over all loadcases to display loadcase specific info
for(var lcobject in ModelObjects) {
   //  Limit the search to loadcases
   if(lcobject.supportsType(typeBasicState)) {

       areport.back.add( Chapter("Loads and results for loadcase : " + lcobject.name) );

       lcobject.setCurrent();

       plotConcepts(areport,areport,"Default display","None",lcobject.name,false,false);

       plotMesh(areport,areport,"Mesh - All","None",lcobject.name,false,false);

       plotResult(areport,areport,"Results - All",lcobject.name,true,true);

       plotCapacity(areport,areport,"Capacity Models","UfTot",lcobject.name,true,true);

    }
}

//Worst loadcase only

setWorstCodeCheckCase();

areport.back.add( Chapter("Capacity check results of named sets for worst loadcase") );
for(var object in ModelObjects) {

    //  Limit the search to named sets
    if(object.supportsType(typeSet)) {
        plotCapacity(areport,object,"Capacity Models","UfTot","Worst",true,true);
    }
}


areport.saveAs("Illustrated_report.doc",mrWordXML);
areport.saveAs("Illustrated_report.htm",mrHtml);
```