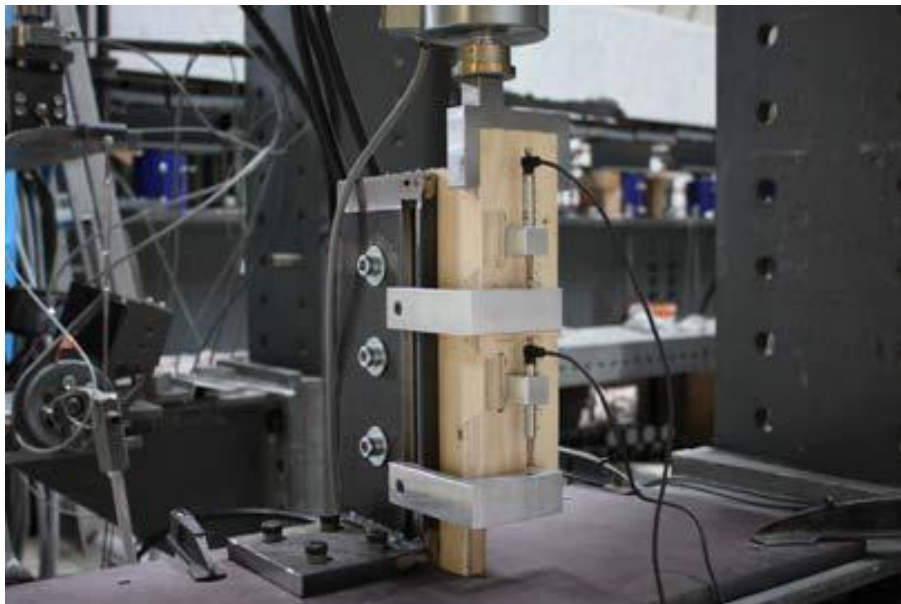




ENAC project: Numerical Model of a Timber Tab-and-Slot Joint

Supervisors: Prof. Yves Weinand, Anh Chi Nguyen, Gamarro Julien

Student : Vincent Lestang



Falls Semester 2018-2019

Content

| | | |
|-------|--|----|
| 1 | Context..... | 3 |
| 1.1 | State of the art..... | 3 |
| 1.1.1 | Integrally attached timber folded surface structures | 3 |
| 1.1.2 | Rotational stiffness of the Multiple Tab-and-Slot Joints (MTSJ)..... | 3 |
| 1.1.3 | Experimental study of the shear strength | 4 |
| 1.1.4 | Missing link in the structural design | 4 |
| 1.2 | Applications..... | 4 |
| 2 | Aims of the project | 5 |
| 3 | Limits of the project..... | 6 |
| 4 | Tasks | 7 |
| 4.1 | Understanding of the CreateJoint plugin..... | 7 |
| 4.2 | Removal of the composite structure..... | 8 |
| 4.3 | Mesh refinement | 11 |
| 4.3.1 | Mesh dimensions | 12 |
| 4.3.2 | Change of type of the elements..... | 13 |
| 4.4 | Boundary conditions | 15 |
| 5 | Results | 17 |
| 5.1 | Comparison with laboratory tests | 17 |
| 5.1.1 | Laboratory tests..... | 17 |
| 5.2 | Summary | 18 |
| 5.3 | Study over the mesh | 19 |
| 6 | Recommendations..... | 20 |
| 7 | Conclusions of the ENAC project | 21 |
| 8 | Acknowledgement..... | 22 |
| 9 | References..... | 22 |
| | Appendix: Comparative results between the experimental campaign and the plugin | 23 |

1 Context

1.1 State of the art

1.1.1 Integrally attached timber folded surface structures

Andrea Stitic investigated the potential of folded structures built with high performance timber panels [7]. The folding of structures permits to achieve gains in material and structural efficiency. A system composed of multiple discrete panels is required to achieve large span, under the constraint of the available fabrication techniques. The connections of the panels are integrated directly on the panel sides, using numerical prefabrication process for the newly rediscovered integrated mechanical attachments.

A large campaign of laboratory tests was completed to quantify the behavior of such structures, highly influenced by the stiffness of the multiple tab-and-slot joints. This process requires large financial needs to reach results that didn't permit to qualify the behavior of discrete joints.

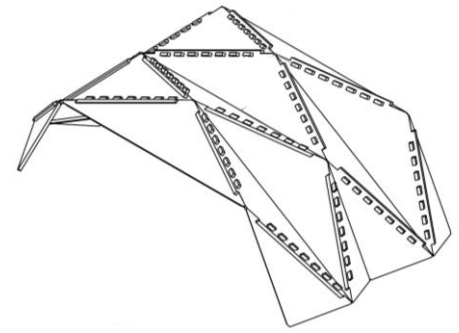


Figure 1 - Timber plate folded structure

1.1.2 Rotational stiffness of the Multiple Tab-and-Slot Joints (MTSJ)

The semi-rigid behavior of these multiple tab-and-slot joints were investigated by Roche & al. [5]. Indeed, it appeared that using this concept to define the joints in numerical models permits to achieve more accurate results regarding the stress and deflections. Semi-rigidity refers to the intermediate state between hinged joints that lets free the degrees-of-freedom related to rotations and rigid joint that block these degrees-of-freedom. The timber folded structures induced a solicitation in bending at the joint, so a more precise knowledge regarding their rotational stiffness is necessary to be able to predict the behavior of such structures. Roche investigated this parameter with experimental tests and the development of an FEM model with Mattoni [3] using the software Abaqus, in such a way that the rotational stiffness of a joint in specific geometric conditions can be easily estimated.

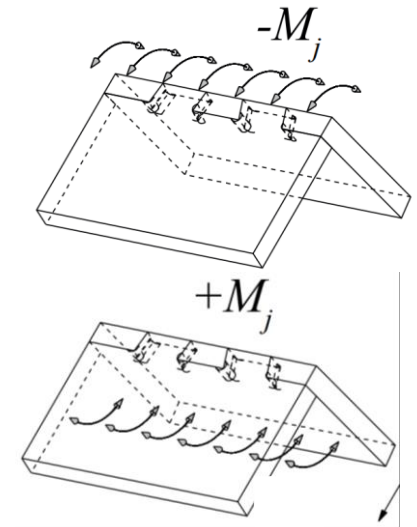


Figure 2 - Moments under study

1.1.3 Experimental study of the shear strength

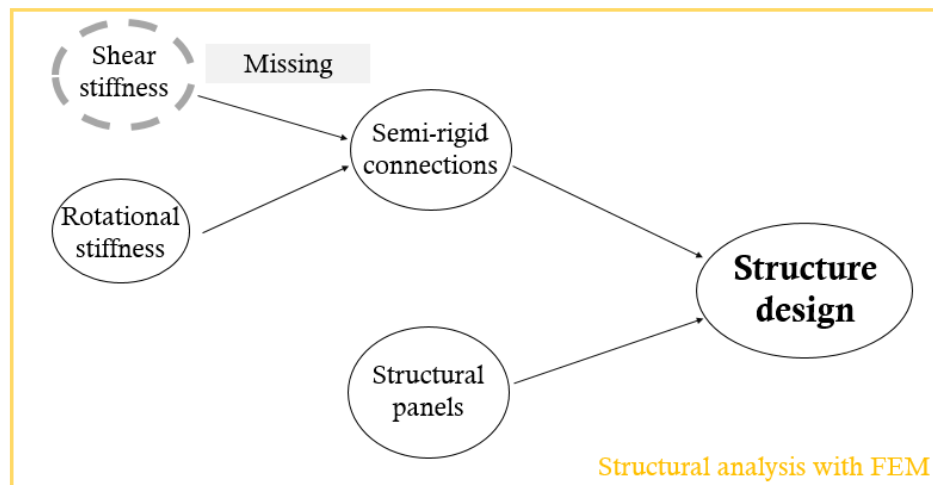
A first experimental campaign was completed to understand the mechanical behavior of the MTSJ in Kerto. Dedijer investigated nine different configurations with a variation of angles to observe the fracture mechanic, stiffness and strength of the joints [1]. The results of this experimental study will be used as a reference in this project.



Figure 3 - Testing device

1.1.4 Missing link in the structural design

The structures under study at IBOIS are mainly based on timber panels connected by MTSJ. Their design using FEM requires to know on the one hand, the behavior of the panels and on the other hand, the stiffness of their connection. In the models, the semi-rigidity of the joints is input as springs, meaning that a displacement in a certain direction is allowed regarding the force or the rotation applied in this direction. The model developed by Roche permits to determine the rotational stiffness, but the model determining the shear stiffness is missing. The development of such model could permit to achieve the entire design of a structure using mainly FEM.



1.2 Applications

Julien Gamarro investigated the slip modulus of joints subjected to shear the use the MTSJ in slab and frame walls applications [2]. A large testing campaign in laboratory was been necessary to determine this mechanical property. During this research, OSB was taken as building material, with different properties than the Kerto used in the research presented in the state of the art.



Figure 4 - Real scale structure

Anh Chi Nguyen studied this constructive solution based on a two layers of horizontal timber panels linked by vertical ones on their sides [4]. The whole system forms boxes that are used to form double-curved arch structures. The used type of joints is multiple tab-and-slot through-tenon, that were modeled as springs with a stiffness equal to the slip modulus which deduced from previous laboratory experiment. The geometry and conditions of the joints are changing along the structure, such that assumptions must be done to implement the properties of the joints. In the end, all these approximations must be validated by real scale experiments. So, a large testing procedure is necessary during the multiple steps of the design process to finally be able to predict the behavior of the structure.

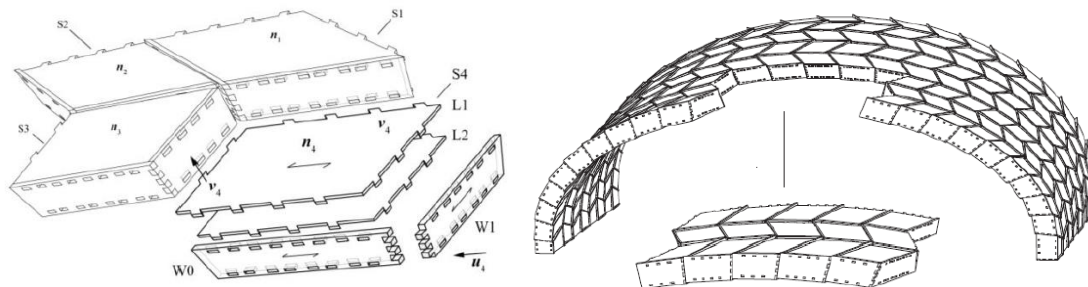


Figure 5 - Constructive principle

2 Aims of the project

As described above, S. Roche and G. Mattoni developed a FEM model on Abaqus of MTSJ using the constitutive relation given by Sandhaas. This model was first used for determining the rotational stiffness and strength for connection between laminated veneer timber (LVL) Kerto panels. The user must input the different timber layers and their orientation. The actual model achieve convergence in a limited number of geometries when used to determine the semi-rigidity of the joints in shear.

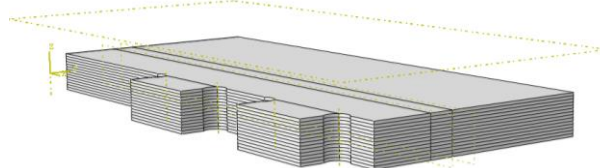


Figure 6 - Layer composite section

The field of application of the model is thus strictly limited in term of material and geometry.

The aims of this project are consequently to improve the existing Abaqus plugin to permit a generalization of material and geometry, such that the semi-rigidity in shear of the joint can be reasonably estimated for any MTSJ conditions.

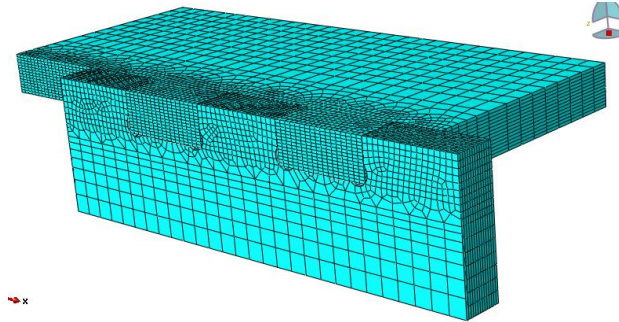


Figure 7 - Mesh of the assembled panels

3 Limits of the project

The model is not supposed to replace the experimental campaigns but to permit to reduce their number. Indeed, the experimental campaign are always necessary because FEM must not be taken as reality, because of some real effects that do not appear in their calculations due to user errors, or their formulation for instance. So, this model aims to be an additional tool that should be used in parallel with experiments to make the experimental investigations less heavy in the design process.

Moreover, this model aims to be representative for the linear elastic part the joints behavior related to the relative displacements of the panels. The stress states haven't been studied, same as the failure mechanism.

4 Tasks

4.1 Understanding of the CreateJoint plugin

The CreateJoint can be accessed on Abaqus as a normal plugin.

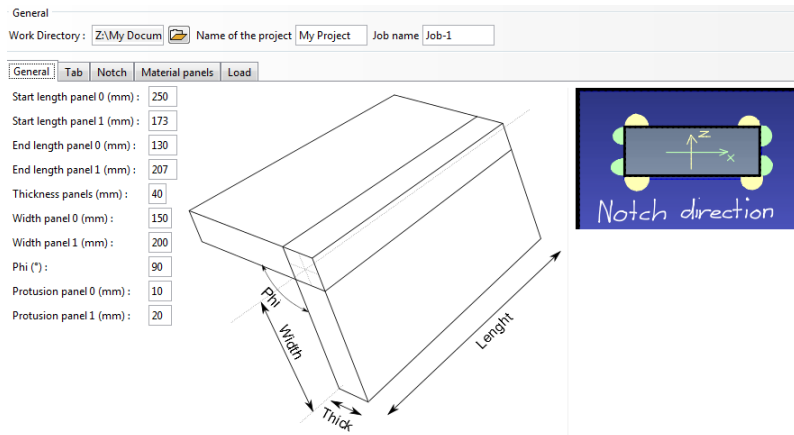


Figure 8 - Screenshot of the user interface

A first step is to understand the existing plug in and its different components.

The program is divided in 3 main scripts:

- **CreateJoint-plugin** is the code communicating with the Abaqus software, it is charged to organize and distribute the variables from the user interface and to run the main functions of the plugin.
- **CreateJointDB** oversees the display of the plugin user interface.
- **CreateJoint** is the main code running successively the different steps of an Abaqus project. It is linked with subprograms that generate the geometry, the material and the type of section, assembly the panels, defines the loading steps, apply the boundary conditions and the loads, and finally mesh the geometry.

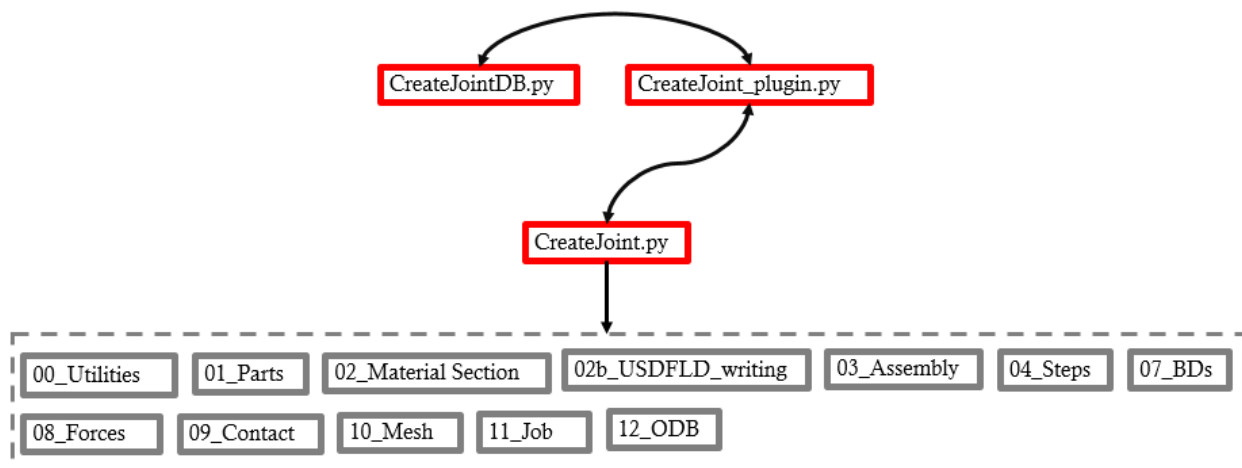


Figure 9 - Plugin codes structure

4.2 Removal of the composite structure

The first big step of this project is the removal of the composite structure of the panel.

The following explanation aims to show the main modifications into the code. Others may have happened, but they follow the same logic than the ones that will be presented in this report.

The user interface part related to the panel composition is deleted because the user doesn't have to input a layer composition anymore.

- **createJoint_DB.py :**

Suppression of the user interface:

To suppress the following command for the user:




Figure 10 - Previous user input in the plugin

The corresponding code is set as comment:

```
260 Layer=FXHorizontalFrame(p=MechaData,opts=FRAME_RAISED|FRAME_THICK|LAYOUT_FILL_X,
261 x=0, y=0, w=0, h=0, pl=DEFAULT_SPACING, pr=DEFAULT_SPACING,
262 pt=DEFAULT_SPACING, pb=DEFAULT_SPACING, hs=DEFAULT_SPACING, vs=DEFAULT_SPACING)
263 # AFXTextField(p=Layer, ncols=20, labelText='Composition:', tgt=form.COMPkw, sel=0)
264 # l = FXLabel(p=Layer, text='I for principal layer, - for crossed layer', opts=JUSTIFY_LEFT)
265 AFXTextField(p=MechaData, ncols=5, labelText='Friction coefficient :', tgt=form.CPKw, sel=0)
```

- **createJoint_plugin.py :**

```
87 self.MATKw = AFXStringKeyword(self.cmd, 'MAT', True, 'Beech LVL-Q 40mm')
88 self.MDKw = AFXFloatKeyword(self.cmd, 'MD', True, 480)
89 self.COMPKw = AFXStringKeyword(self.cmd, 'COMP', True, 'III-III-III-III')
90 self.alpha0Kw = AFXFloatKeyword(self.cmd, 'alpha0', True, 0)
```

Figure 11 - Definition of the variable 'COMP'

The CreateJoint method requires 44 inputs. As shown above, we suppressed some inputs given by the user, so we have to modify the method definition such that it won't ask for these.

```
110
111 def createJoint(workDirPath,jobName,modelName,theta1,theta2,theta3,phi,alpha0,alpha1,L0end,L1end,T0,w0,
112 w1,P0,P1,L0start,L1start,CL,C,Ltab,Gtab,Ntab,JT,JTend,N0type,N1type,N0diam,N1diam,N0dir,N1dir,
113 N0of,N1of,MAT,MD,MC,SL,GF,CP,COMP,LdT,Ld,LdL,BdL): #,troughTenon,border): #DG,jointType,
```

The previous variable permitted to define the number of layers and their thickness when combined with the total thickness. The variables describing the layers properties are turned to comments.


```

179 # N0layer=len(COMP)
180 # O0layer=[] # layer orientation
181 # for i in range(N0layer):
182 #     if COMP[i]=='I' :
183 #         O0layer.append(0)
184 #     elif COMP[i]=='-' :
185 #         O0layer.append(90)
186 # T0layer=T0/N0layer
187
188 # N1layer=len(COMP)
189 # O1layer=[] # layer orientation
190 # for i in range(N1layer):
191 #     if COMP[i]=='I' :
192 #         O1layer.append(0)
193 #     elif COMP[i]=='-' :
194 #         O1layer.append(90)
195 # T1layer=T1/N1layer

```

- **01_Parts**

The program was cutting up the panels in cells, depending on the position regarding the joint. These cells are used to make the mesh finer when closed to the joint and larger when far from the joint. These cells were sliced afterward making subparts of cells corresponding to the thickness of the layers.

The partition in cells but the second partition of the panel in layers is suppressed. In the end, two layer0 and layer1 that contain the cells geometric entities.

```

1171 allCells0=[]
1172 for k in range(len(panel0.cells)):
1173     allCells0.append(panel0.cells[k])
1174 allCells0=tuple(allCells0)
1175
1176 allCells1=[]
1177 for k in range(len(panel1.cells)):
1178     allCells1.append(panel1.cells[k])
1179 allCells1=tuple(allCells1)
1180
1181 pCell0=panel0.cells.pointsOn
1182 pCell1=panel1.cells.pointsOn
1183 sCell0=[]
1184 sCell1=[]
1185 for i in range(len(pCell0)): # create empty array of right dimension (nLayer)
1186     sCell0.append([])
1187 for i in range(len(pCell1)): # create empty array of right dimension (nLayer)
1188     sCell1.append([])
1189 for i in range(len(pCell0)):
1190     sCell0[i]=pCell0[i]
1191 for i in range(len(pCell1)):
1192     sCell1[i]=pCell1[i]
1193
1194 layer0=[]
1195 layer1=[]
1196 # # Put all the cells into one array per panel
1197 for i in range(len(sCell0)):
1198     layer0.append(panel0.cells.findAt(sCell0[i],))
1199 layer0=tuple(layer0)
1200
1201 for i in range(len(sCell1)):
1202     layer1.append(panel1.cells.findAt(sCell1[i],))
1203 layer1=tuple(layer1)

```

- **02_MaterialSection**

The material properties are now defined as for an isotropic material with the stiffness in the different direction. These were previously discretized for every layer.

```
10 (E1,E2,E3,v12,v13,v23,G12,G13,G23)=tuple(MC)
11 tableConst2=[E1,E2,E3,v12,v13,v23,G12,G13,G23]
12 tableConst=[]
13 tableConst.append(tableConst2)
24 material=jointModel.Material(name=MAT)
25 material.Density(table=((MD, ), ))
26 material.Elastic(type=ENGINEERING_CONSTANTS, dependencies=0, table=tuple(tableConst))
27 material.UserDefinedField()
28 material.Depvar(n=32)
```

The properties are defined in the global reference system, so the mesh can be defined without paying attention to the material definition.

For a layered definition of material, the section must a CompositeSolidSection object. Consequently, the type of section is changed to a HomogenousSolidSection and because the section of layers isn't required anymore.

Before:

```
29 sectionLayer = section.SectionLayer(material=MAT, thickness=1,
30 orientAngle=0.0, numIntPts=3, plyName='SingleVeneer')
31 jointModel.CompositeSolidSection(name='SectionComposite',
32 layupName='SingleLayer', symmetric=False, layup=(sectionLayer, ))
```

After:

```
35 jointModel.HomogeneousSolidSection(material=MAT,name='FullSection', thickness=None)
```

This type of section is then assigned to both panel models.

- **03_Assembly**

Due to the structure in layers of the panels, their assembly had to be done layer by layer, thanks to loops that navigate over the thickness of the panels. These loops are suppressed because the new definition of cells results in main single cells.

Before:

```
61 pi=V2P(P2V((Pref1[0]+L1,0,0))+(T1/2-T1layer/2)/sin(phiRad)*VZ)
62 shearFace.append(secondInstance.faces.findAt((pi,)))
63 print 'error here'
64 for j in range(N1layer-1):
65     shearFace.append(secondInstance.faces.findAt(((pi[0],pi[1],pi[2]-(j+1)*T1layer/sin(phiRad)),)))
```

After:

```
61 pi=V2P(P2V((Pref1[0]+L1,0,0))+(T1/2-T1layer/2)/sin(phiRad)*VZ)
62 shearFace.append(secondInstance.faces.findAt((pi,)))
63 print 'error here'
64 shearFace.append(secondInstance.faces.findAt(((pi[0],pi[1],pi[2]-(1)*T0/sin(phiRad)),)))
```

- **06_Assembly**

The procedure is the same than for the assembly of the panels. The loops navigating through the thickness of the panels are suppressed.

Before:

```

56 # Contact 2
57 contactFace1=[]
58 ▼ for i in range(1,len(intLine0)-2,2):
59     tempV=P2V(extLine0[i])-P2V(intLine0[i])
60     tempV=tempV/tempV[1]*T0layer
61     pi=V2P(0.5*(P2V(intLine0[i])+P2V(intLine0[i+1]))+0.5*tempV)
62     contactFace1.append(firstInstance.faces.findAt((pi,)))
63     for j in range(N0layer-1):
64         contactFace1.append(firstInstance.faces.findAt((V2P(P2V(pi)+(j+1)*tempV),)))
65

```

After:

```

44 contactFace1.append(firstInstance.faces.findAt((pi,)))
45 #for j in range(N0layer-1):
46 contactFace1.append(firstInstance.faces.findAt(((pi[0],pi[1]+(1)*T0layer,pi[2]+((1)*T0layer)/tan(phiRad)),)))

```

4.3 Mesh refinement

After running the code, the mesh appears being unsatisfying. The seed were defined regarding the thickness of the layers. Moreover, the samples have a complex geometry which is hard to fit, especially for the notch due to the CNC for the automation of the production of elements.

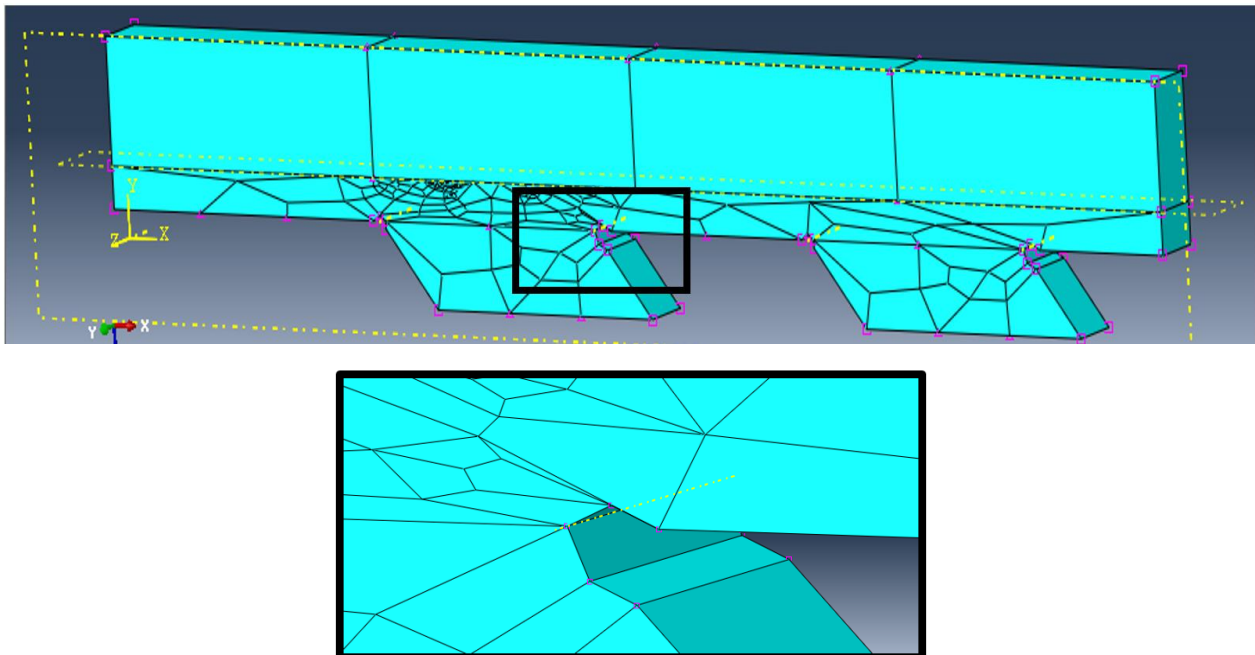


Figure 12 - Mesh approximation in fitting notches

Previously, the mesh seed was defined in function the layers thickness:

- **10_Mesh:**

```
minSeed0=T0layer*2
maxSeed0=T0layer*6
```

Consequently, a first approach is to define an input into the user interface such that the mesh size can be easily modified. Thus, the user is active in the choice of mesh seeds dimensions that has an influence on the convergence of the model and the accuracy of the results. One of the aim of this project is to make possible the automation of tests. Consequently, a seed dimension must be defined in the end regarding the thickness of the panels or another parameter.

4.3.1 Mesh dimensions

- User interface tab

In order to avoid to go into the code each time the user wants to try a new geometry, it appeared relevant to create a new tab in the graphical interface, in which the user can input the value of the variable maxSeed0, minSeed0, respectively maxSeed1, minSeed1, that correspond to the maximal and minimum seeds dimensions for the panel 0, respectively the panel 1.

A new tab is created in CreateJointDB and given as input for CreateJoint. The main tasks has been to input these variables into the code CreateJoint_plugin because of the indentations which have a chaotic organization.

Apart the suppression of the previous mesh dimensions into the code 10_Mesh, no other action is necessary into the code.

- **CreateJoint_plugin.py:**

```
83 self.N0ofKw = AFXStringKeyword(self.cmd, 'N0of', True, '0')
84 self.N1dirKw = AFXStringKeyword(self.cmd, 'N1dir', True, 'X-direction')
85 self.N0ofKw = AFXStringKeyword(self.cmd, 'N0of', True, '0')
86 self.N1ofKw = AFXStringKeyword(self.cmd, 'N1of', True, '0')
87 self.MATKw = AFXStringKeyword(self.cmd, 'MAT', True, 'Beech LVL-Q 40mm')
88 self.MDKw = AFXFloatKeyword(self.cmd, 'MD', True, 480)
89 #self.COMPKw = AFXStringKeyword(self.cmd, 'COMP', True, 'III-III-III-III')
90 self.alpha0Kw = AFXFloatKeyword(self.cmd, 'alpha0', True, 0)
91 self.alpha1Kw = AFXFloatKeyword(self.cmd, 'alpha1', True, 0)
92 self.minSeed0Kw = AFXFloatKeyword(self.cmd, 'minSeed0', True, 0.005)
93 self.maxSeed0Kw = AFXFloatKeyword(self.cmd, 'maxSeed0', True, 0.05)
94 self.minSeed1Kw = AFXFloatKeyword(self.cmd, 'minSeed1', True, 0.005)
95 self.maxSeed1Kw = AFXFloatKeyword(self.cmd, 'maxSeed1', True, 0.05)
96 self.LdTKw = AFXStringKeyword(self.cmd, 'LdT', True, 'Shear')
97 self.LdKw = AFXIntKeyword(self.cmd, 'Ld', True, 200000)
98 self.LdLKw = AFXFloatKeyword(self.cmd, 'LdL', True, 40)
```

This results into a tab in which the user input his desired dimensions in meters.

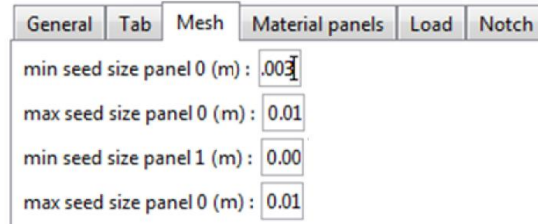


Figure 13 - Screenshot of the additional panel

- Automation

Based on the experience, it appears that the most effective minimal seed dimension is about $0.15t_{\text{panel}}$ and the maximal one is about $0.5t_{\text{panel}}$, with t_{panel} being the thickness of the panel. There relation one the other is important. Indeed, every panel is more-or-less divided in three main volumes (cells).

- One close to the panel connection in which the effect of the shear will be the more important and visible. So, the size of the mesh must be minimized in this cell.
- One far from the panel connection in which the effect of the shear will be the less important and visible. This part is necessary for constructive reasons during the test but are not the ones giving the most precious data. the size of the mesh must be maximized in this cell.
- An intermediate cell which is linked with the two others described above. So, the size of the mesh varies between the larger and shorter dimension. So, the quality of the meshing in this part is highly dependent on the mesh seed dimensions, that must not be consequently too different.

So, it seems that the optimal ratio between the shorter and larger dimensions is about 3/10.

• 10_Mesh:

```
12 minSeed0=0.15*T0
13 maxSeed0=0.5*T0
```

4.3.2 Change of type of the elements

The work on the mesh dimensions isn't necessary to achieve a mesh of good quality. Indeed, it appears that the complexity of the panels induce error when meshing the areas next to the notches.

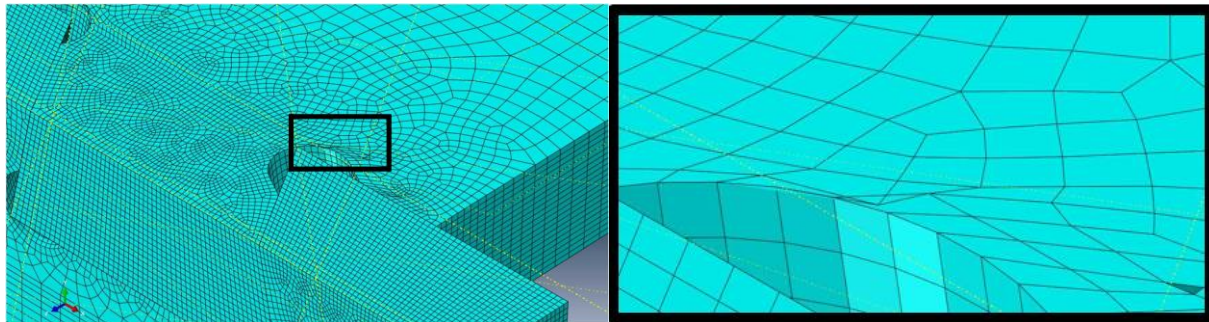


Figure 14 - Problems with the mesh

The actual elements used for meshing the panel are hexahedral elements. Their shape doesn't fit the required one to achieve a proper meshing. So, this made necessary to change the element shape to fit more complex geometries. The tetrahedral elements are efficient to achieve such results. Their weakness is that the time of calculation which can be increased.

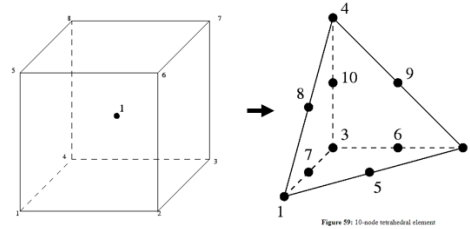


Figure 15 - New type of elements

Consequently, the elements were changed from HEX to TET. The meshing technique wasn't adapted anymore to these new elements. Indeed, the previous technique was SWEEP, meaning that a 2D mesh is created on one face and then extruded. This technique is efficient for planar elements but not for complex volumetric elements such that tetrahedral. The meshing technique is set as FREE, meaning that the software chose itself the most appropriate technique.

The element code elemCode can be found on the internet. Lots of tetrahedral elements can be found in libraries

- 10_Mesh:

```

5 import mesh
6
7 elemType = mesh.ElemType(elemCode=C3D10M, elemLibrary=STANDARD, secondOrderAccuracy=OFF)
8 #####
9 # PANEL 0
10 #####
11
12 # minSeed0=0.01
13 # maxSeed0=0.1
14
15 stackFace0=panel0.faces.findAt((L0/2,T0/2,Pext0[2]-(Q0-BdL)/2))
16 # stackFace0=panel0.faces.findAt((L0/2,0,(negZ0max+negZ0min)/2))
17 # print (L0/2,0,(negZ0max+negZ0min)/2)
18 panel0.setElementType(regions=(centrCell0, ), elemTypes=(elemType,))
19 panel0.setMeshControls(regions=centrCell0, elemShape=TET, technique=FREE, algorithm=ADVANCING_FRONT, allowMapped=True)
20
21 centrEdges0=[]
22 for i in range(len(centrCell0)):
23     for j in range(len(centrCell0[i].getEdges())):
24         centrEdges0.append(panel0.edges[centrCell0[i].getEdges()[j]])
25
26 panel0.seedEdgeBySize(edges=centrEdges0, size=minSeed0, deviationFactor=0.1, constraint=FINER)
27 panel0.assignStackDirection(cells=centrCell0, referenceRegion=stackFace0)
28
29 if P0 > 1.5*T1/sin(phiRad):
30     panel0.setElementType(regions=(protCell0, ), elemTypes=(elemType,))
31     panel0.setMeshControls(regions=protCell0, elemShape=TET)
32

```


4.4 Boundary conditions

It appeared that when the geometry of the MTSJ gets complicated, the software is unable to reach convergence. This problem was also present on the layered model.

The loading simulations on the model is divided into two steps:

- During the first step, the MTSJ model is initialized, meaning that the panels are assembled, the boundary conditions are applied...
- During the second step, the load is applied incrementally on one of the panel, activating the panel connection stiffness.

The monitoring of the calculations showed that the first step was getting well but the second step wasn't finding convergence after multiple tries for the first increment of load. When checking the deflection after the first step, it appeared that the loaded panel was sliding out of the connection, explaining why Abaqus was unable to compute. The red arrows show the higher displacements and the blue ones the smaller ones.

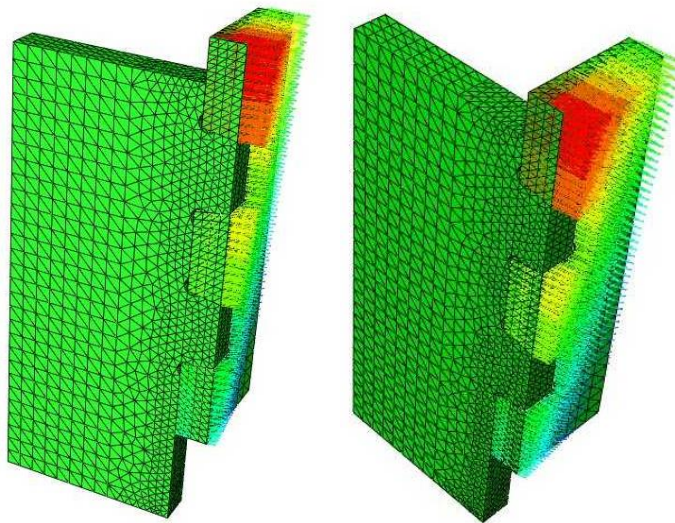


Figure 16 - Displacements after the first step

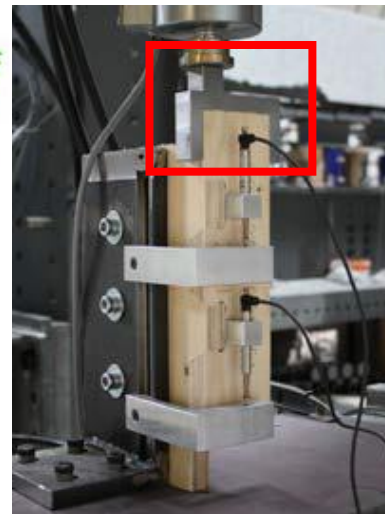


Figure 17 - Missing boundary condition

This shows that the boundary conditions in the plugin aren't correct. Our computations are based and compared with real samples that have been tested. These tests worked fine for the same loading type, so our model should be able to reach convergence too. Indeed, the comparison with reality shows that the application of load required a clamping system on the top of one of the panels, having support effects at the same time.

So, it is necessary to add supports to the model to achieve convergence. To do so, the existing curves and surface are used to control the displacements about the global y and z axis. The x-direction is the one of application of the shear load. A more precise definition may have been achieved regarding the real clamping system, but it didn't appear relevant to proceed now to its implementation because the testing procedure may change in the future. The definition of the boundary conditions must be consequently changed either.

It is first necessary to define the surfaces and edges on which these boundary conditions will be applied.

- **03_Assembly:**

```

60 ▾ elif LdT == 'Shear':
61     shearFaceFixed=[]
62     #Added surfaces and curves regarding the experimental test procedure/ December 2018
63     shearFaceSliding=[]
64     shearEdgeSliding1=[]
65     shearEdgeSliding2=[]
66     shearEdgeSliding3=[]
67     shearEdgeSliding4=[]
68     shearEdgeSliding5=[]
69     shearEdgeSliding6=[]
70
71     pi=V2P(P2V((Pref1[0]+L1,0,0)))+(T1/2)/sin(phiRad)*VZ)
72     shearFaceFixed.append(secondInstance.faces.findAt((pi,)))
73     shearFaceFixed.append(secondInstance.faces.findAt(((pi[0],pi[1],pi[2]),)))
74
75     BCsFaceControlled=secondInstance.faces.findAt(((P1min0[0]/2+L1,-P1min0[2],0),))
76     shearFaceSliding.append(secondInstance.faces.findAt(((P1min0[0]/2+L1,-P1min0[2],0),)))
77
78     BCsEdgeSliding1 = secondInstance.edges.findAt(((P1min0[0]+L1,P1min0[2]/2,P1min0[2]),))
79     shearEdgeSliding1.append(secondInstance.edges.findAt(((P1min0[0]+L1,P1min0[2]/2,P1min0[2]),)))

```

- **07_BDs:**

```

18 if LdT == 'Shear':
19     controlledRegion=regionToolset.Region(faces=BCsFaceControlled)
20     jointModel.DisplacementBC(name='Second Panel Controlled Face', createStepName='Initial',region=controlledRegion, u2=SET, u3=SET)
21     faces2=shearFaceSliding
22
23     #Added BC regarding the experimental test procedure/ December 2018
24     region = contact.Set(faces=faces2, name=LdT+' Face2')
25     controlled2Region=regionToolset.Region(edges=BCsEdgeSliding1)
26     jointModel.DisplacementBC(name='Second Panel Controlled Edge 1', createStepName='Initial',region=controlled2Region,u2=SET, u3=SET)
27     edges=shearEdgeSliding1
28     region = contact.Set(edges=edges, name=LdT+' Face3')

```

The following boundary conditions are consequently obtained:

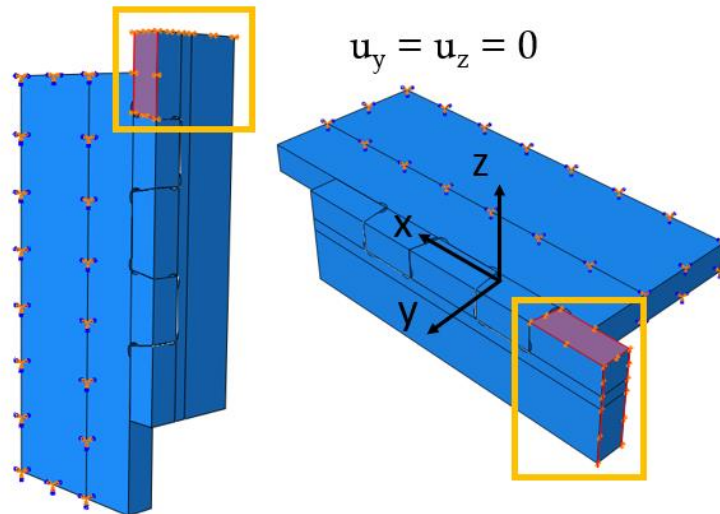


Figure 18 - Additional boundary conditions (into the yellow rectangle)

5 Results

5.1 Comparison with laboratory tests

5.1.1 Laboratory tests

In order to investigate the accuracy of the plugin for shear tests, results from FEM are compared with the results of the laboratory tests run by Dedijer [1]. The testing dispositive consists in applying a pressure load on the surface of one of the panel such that a shear force is induced at the joint. The displacements of the panel on which the load is applied are monitored such that the stiffness of the joints can be computed afterward.

In Abaqus, the load is only applied on a surface aligned with the joint. Indeed, if the load was applied on the whole face, the joint would also be subjected to a moment due to the eccentricity between the resulting load and the joint.

As presented before, the only investigated parameter is the elastic stiffness of the joint, consequently no fracture mechanic will be observed.

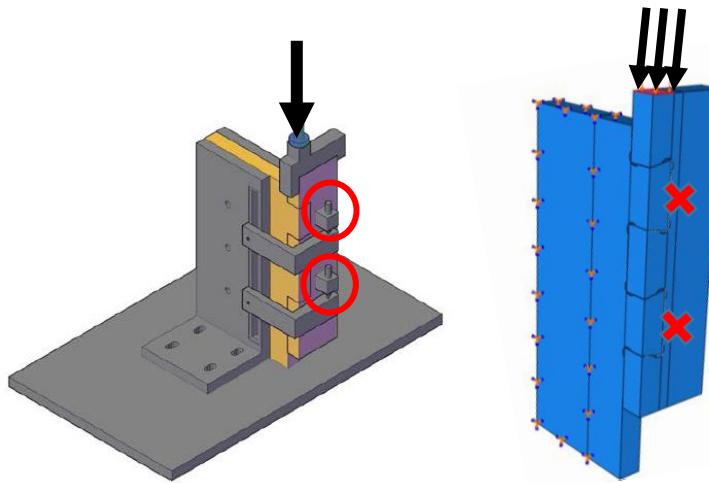
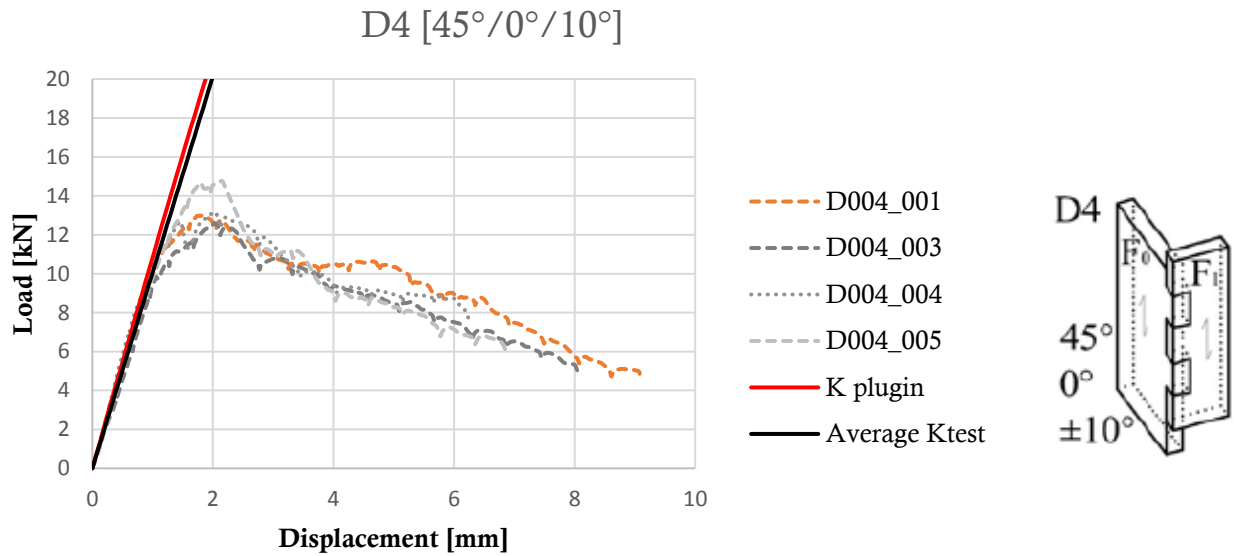


Figure 19 - Load application and displacement monitoring for the real experimental device (left) and the model (right)

The displacements are extracted from points that must be selected manually on the model (red cross of the above figure). This point could be a limitation in the automation of the plugin, indeed the mesh is changing with the variation of geometry. So, an element index doesn't always correspond to the same position on the sample.

The stiffness of the joint is investigated in 9 different configurations. The model D2 didn't reach convergence. This is due to the boundary conditions defined above that appear being a good approximation for most of the angles combination but not for this one. Suppressing of the newly defined boundary conditions solve the problem. The comparison between the load-displacement curve obtained from the software and from the tests can be seen in the appendix.

Joint type 4:

For each model, an average stiffness of the joints is computed for the lab tests by taking successive load-displacement relations in the elastic part. Consequently, a relative error between the model and the real samples can be computed:

$$\text{Relative error } \delta = \frac{K_{ser,plugin} - K_{ser,tests}}{K_{ser,tests}}$$

In this case:

- $K_{ser,tests} = 10'072 \text{ [N/mm]}$
- $K_{ser,plugin} = 10'817 \text{ [N/mm]}$
- $\delta = 7.4 \text{ [%]}$

For this model, the results are close to the real cases, but the relative error is superior to the conventional threshold of 5%.

5.2 Summary

The results from the tests and the model are summarized:

| Geometry | D1 | D3 | D4 | D5 | D6 | D7 | D8 | D9 |
|---------------------------------------|-------|-------|--------|-------|-------|-------|-------|-------|
| $K_{ser,experimental} \text{ [N/mm]}$ | 10143 | 7544 | 10072 | 6896 | 6635 | 11584 | 6933 | 6564 |
| $K_{ser,plugin} \text{ [N/mm]}$ | 10663 | 5601 | 100817 | 3868 | 3979 | 11163 | 5704 | 5907 |
| Relative error [%] | 5.1 | -25.8 | 7.4 | -43.9 | -40.0 | -3.6 | -17.7 | -10.0 |

The relative error varies a lot, it is not possible to determine a global tendency. As explained above the new boundary conditions have a large effect on the results. To reduce these errors, the implementation of the real boundary conditions could be a solution.

Another parameter that can have an influence on the results is the size of the mesh.

5.3 Study over the mesh

As explained before, a mesh size has been implemented based on the experience, after several tries. The following study aims to investigate the sensitivity of the stiffness of the joints from the model to this mesh size.

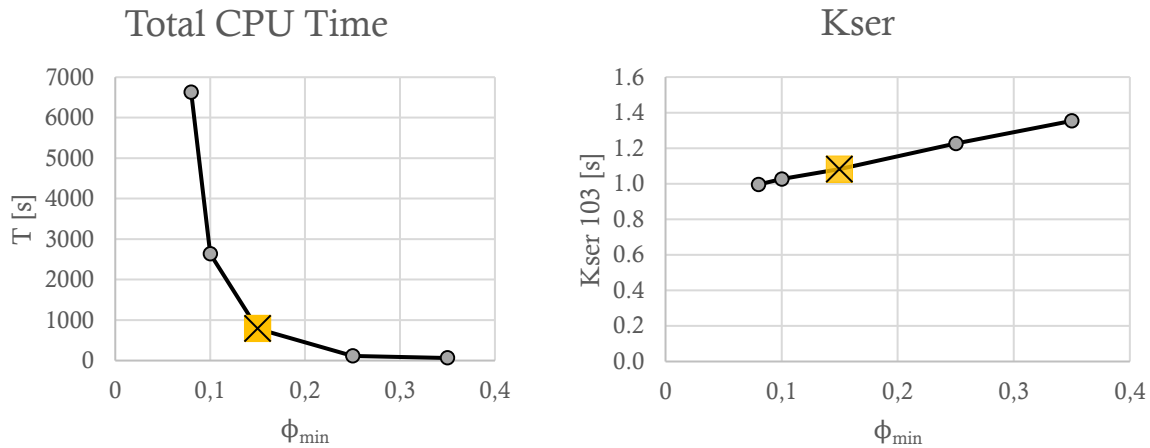
The size of the mesh is defined in the following way:

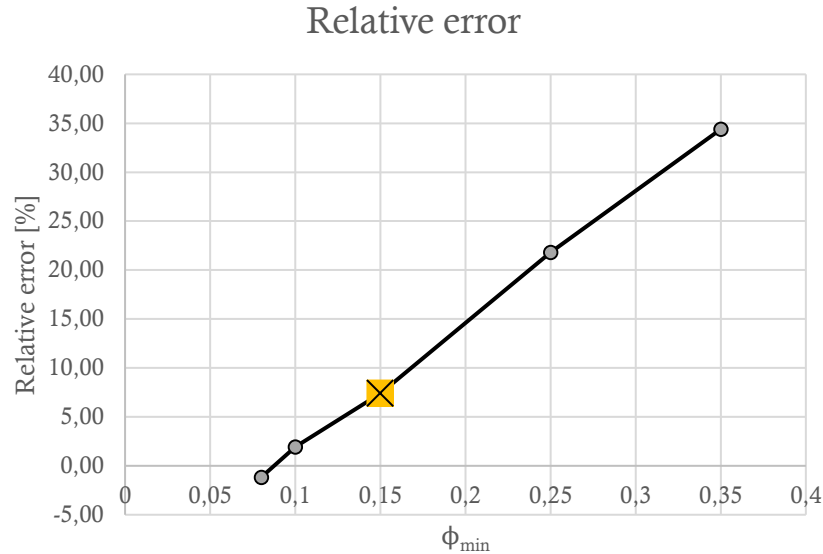
- Minimal mesh size = $\phi_{\min} t_{\text{bois}}$
- Maximal mesh size = $\phi_{\max} t_{\text{bois}} = 3/10 \cdot \phi_{\min} t_{\text{bois}}$

The mesh size corresponding to the one used to find the previous results is marked in yellow (see 4.3.1). The minimum mesh size coefficient is varying, and the maximum size is consequently adapted.

| | | | | | |
|---------------|------|------|------|------|------|
| ϕ_{\min} | 0.08 | 0.1 | 0.15 | 0.25 | 0.35 |
| ϕ_{\max} | 0.27 | 0.33 | 0.50 | 0.83 | 1.17 |

For the test D4, the following results are obtained:





The total CPU time corresponds to the time between to code starts and ends to run. When the minimum size factor decreases, the total CPU time increases exponentially. The variation doesn't appear changing a lot, but it decreases linearly with the size coefficient. Same as for the relative error that approaches precisely the real stiffness. So, this gain in precision has a large computational cost. Consequently, it is necessary to make a choice between having less accurate results with a reduced computational time or more precise results under the constraint related to the desired level of precision on the behavior of the structure

6 Recommendations

As shown before, lots of parameters have an influence on the results. During the process of improvement of the plugin, some parameters have been fixed: the type of elements, the type of section and the ratio between the extreme mesh sizes. It should not be necessary to investigate these parameters in the future. Nevertheless, the great influence of some parameters was also highlighted: the boundary conditions and the mesh size.

It is recommended to investigate a good combination between the mesh size and the boundary conditions comparing with some experiments in lab, such that the plugin is calibrated with the testing device. As explained These parameters can be changed at the code lines showed in figures in the paragraphs 4.3.1 and 4.4.

Moreover, the implementation of the precise boundary conditions hasn't been completed here due to its complexity that required to create news lines and surfaces. It is recommended to use simple and repetitive testing device, otherwise the plugin use would be inefficient. Indeed, the implementation of the boundary conditions is a long and tedious task, so the plugin will not be used for human reasons if the devices are always too complicated to implement. Moreover, it would be relevant to dissociate the boundary condition device from the load application device.

7 Conclusions of the ENAC project

This project has been a good way for me to investigate the relationship between FEM models and real case studies at scale of a connection, which is a topic that I didn't explore so much during my studies.

The first task was to understand how the plugin was built and the logic of the previous developers. Then I proceed to the generalization in term of material with suppressing the composite structure of the panel to a homogenous section. It appeared the layers permitted to ensure a certain quality of mesh, so I changed the type of element to fit more precisely the complex geometry of the panels with a good quality mesh. From a first group of results, I investigated why the convergence wasn't achieved for some complex connection geometries. This was solved with a task on the boundary conditions. Finally, I investigated the influence of the mesh size such that I could give recommendations for the future use of the plugin.

8 Acknowledgement

I would like to thank Anh Chi and Julien for the liberty they allow me to investigate solutions to the different issues I faced during this project and more generally for their patience and reactivity. I also would like to thank the IBOIS team for its friendly welcome when I used to come to use the lab facilities. Finally, I would like to thank Prof. Weinand to propose this ENAC project which was instructive in term of knowledge and reflection, and for his good questioning during our meetings.

9 References

- [1] M. Dedijer; S. N. Roche; Y. Weinand : ***Shear Resistance and Failure Modes of Edgewise Multiple Tab-and-Slot Joint (MTSJ) Connection with Dovetail Design for Thin LVL spruce plywood Kerto-Q Panels.*** 2016. *World Conference on Timber Engineering, Vienna, Austria, August 22-25, 2016.*
- [2] J. Gamero, I. Lemaître, Y. Weinand : ***Mechanical characterization of timber structural elements using integral mechanical attachments.*** 2018. *World Conference on Timber Engineering, Seoul, Republic of Korea, August 20-23, 2018.*
- [3] Geoffroy Mattoni: ***Folded plate structure, Design and analysis of woodworking joints for structural timber panels,*** Lausanne, EPFL, 2015
- [4] Anh Chi Nguyen/ Yves Weinand: ***Development of a spring model for the structural analysis of a double-layered timber plate structure with through-tenon joints,*** 2018. *World Conference on Timber Engineering, Seoul, Republic of Korea, August 20-23, 2018.*
- [5] S. N. Roche / Y. Weinand: ***Semi-Rigid Moment-Resisting Behavior of Multiple Tab-and-Slot Joint for Freeform Timber Plate Structures,*** Lausanne, EPFL, 2017., DOI : 10.5075/epfl-thesis-8236.
- [6] S. Roche; J. Gamero; Y. Weinand : ***Multiple Tab-and-Slot Joint : Improvement of the Rotational Stiffness for the Connection of Thin Structural Wood Panels.*** 2016. *World Conference on Timber Engineering, Vienna, Austria, August 22-25, 2016.*
- [7] A. Štitić / Y. Weinand (Dir.): ***Integrally attached timber folded surface structures,*** Lausanne, EPFL, 2017., DOI : 10.5075/epfl-thesis-8114.

Appendix: Comparative results between the experimental campaign and the plugin

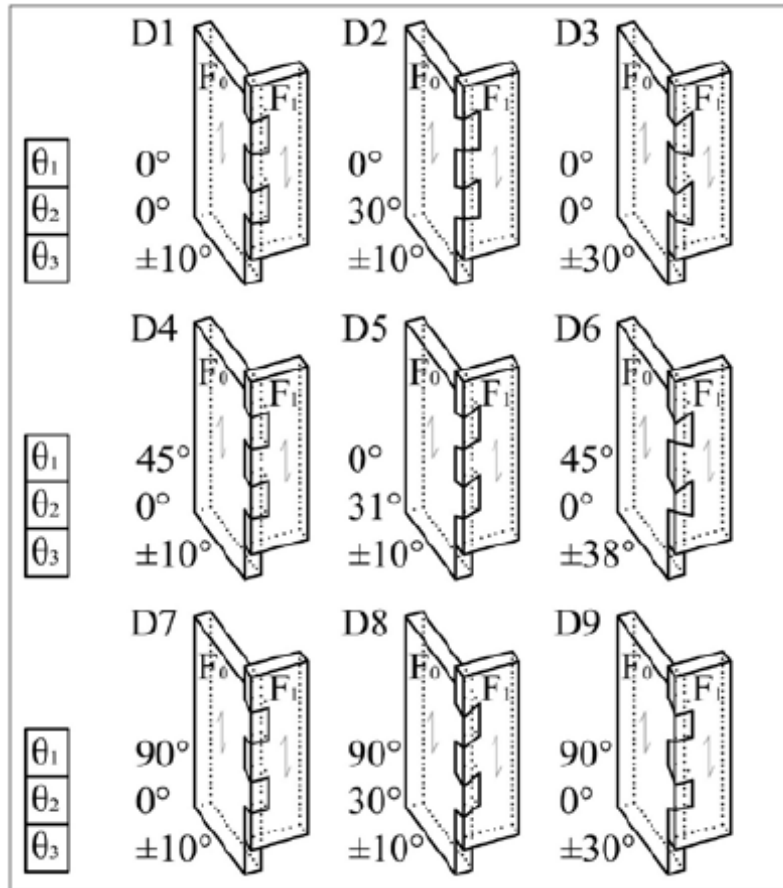


Figure 20 - Geometry of the connections under study

