POLITECNICO DI TORINO

Aerospace Engineering
Master´s Degree

Master's Degree Thesis Attachment

Detailed explanation of the API for Femap about failure criteria

Supervisor

prof. Erasmo Carrera

Candidate

Amedeo Grasso

March 2018

1 - Table of contents

2 - Api Code Explanation

The API allows three main activities:

1) Evaluation of Hoffman´s failure index(2D) and margin of safety;
2) Evaluation of failure indexes basing on LaRC05 criteria;
3) Evaluation of different margin of safety about buckling condition of panels (a company´s request).

In the following paragraphs the code of each will be described, adding brief comment to collective parts also.

2.1 - Collective parts and algorithms

In the first part, all public variable, that will be used by all procedures, were declared as in following rows [5-86]:

```
Public Class frmMain
6 ' variabili globali Femap
7 Public APP As femap.model
8 Public fmpOS As femap.OutputSet
9  Public fmpOut As femap.Output
...
85 Public Consist23(nOC - 1) As Double
86 Public Consist13(nOC - 1) As Double
```

Then was imposed the saving of all parameters included in the graphical user interface (GUI) when the program will be closed after a session of utilization [88-90]:

```
88 Private Sub frmMain_FormClosing(sender As Object, e As FormClosingEventArgs) Handles
Me.FormClosing
89 My.Settings.Save()
90 End Sub
```

Finally, the api was linked to the current active session of Femap with rows below [92-94]:

```
92 Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load APP =
GetObject(,"femap.model")
93 rc = APP.feAppRegisterAddInPane(True, Me.Handle, Me.Handle, False, False,2, 0)
94 End Sub
```

To reduce the risk of user´s error in the selection and to give a more user-friendly GUI, a lot of its elements were controlled by following commands [1717-1974], for example:

```
Private Sub RadioButton_LaRC05_CheckedChanged_1(sender As Object, e As
EventArgs) Handles RadioButton_LaRC05.CheckedChanged
If Me.RadioButton_LaRC05.Checked = True Then
Me.GroupBox_OpMod.Visible = True
Me.GroupBox_OpMod.Enabled = True
Me.RadioButton_Hoffman.Checked = False
Me.EntireMod.Enabled = True
Me.GroupMod.Enabled = True
Me.txtalpha0.Enabled = True
End If
End Sub
```

To allow user to do many analysis in the same session a reset of variables was necessary:

```
'RESET VARIABLE---------------------------------------------------- ----------------------------------------------------------
fmpMat = Nothing
fmpPr = Nothing
fmpPly = Nothing
fmpEl = Nothing
fmpOS = Nothing
fmpOut = Nothing
fmpElSet = Nothing
fmpMatSet = Nothing
fmpOV = Nothing
fmpCASEset = Nothing
nOC = Nothing
vOCids = Nothing
s = Nothing
ElIds = Nothing
'-------------------------------------------------------------------------------
```

2.2 – Hoffman and LaRC05 button

To evaluate failure using Hoffman or LaRC05 theory the program would run at the beginning some identical operations to read analysed model data.
Moreover, the API check that all needed inputs are put by user:

1°) Select stress data source [99-103]

```
If ComboBox_Stress.Text = "Stress Data - Choose Source" Then
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, " ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "ERROR: To choose a stress option!!!!")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, " ")
Return
```

In fact, it is possible to import stresses from an external file.

2°) Choice to analyse the entire model or only a group of it [106-112]

```
If Me.GroupMod.Checked = False Then 'Error control
If Me.EntireMod.Checked = False Then
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, " ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "ERROR: Choose an operative mode (group or
entire)!!")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, " ")
Return
End If
```

3°) If "*group mod"* is selected as operative mode, the user must insert the group ID or he must choose *"actived group option"*. [114-123]:

```
Else
If Me.CheckBox_activegroup.Checked = True Then
grID = APP.Info_ActiveID(24)
Else
grID = Me.txtgrID.Text
End If
If grID = 0 And Me.CheckBox_activegroup.Checked = False Then
APP.feAppMessage(4, "No group selected!")
Return
End If
```

4°) The last inputs are the load cases. They are selected from Femap window in *Fig.1 [129-130]:*

```
APP.feSelectOutputSets("Select LOAD CASEs to reading",fmpCASEset)
rc = fmpCASEset.GetArray(nOC, vOCids)
```

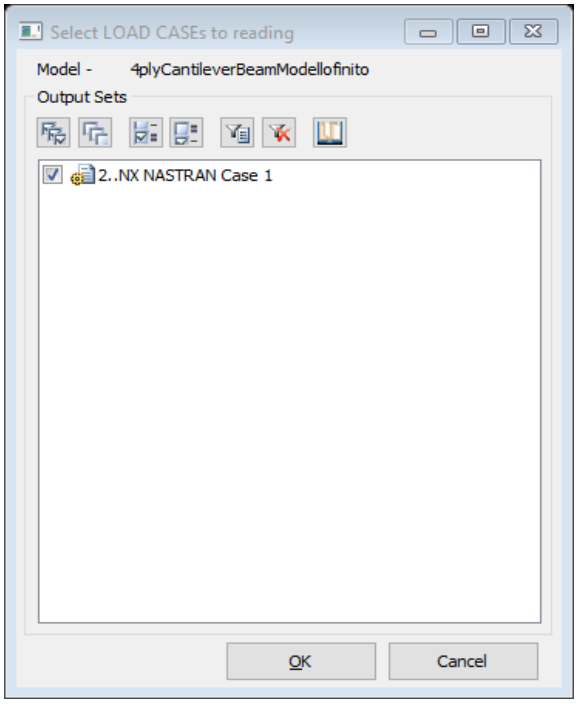

Figure 1 - Load case selection window

After these controls, specifics operations for each theory are run.

2.2.1 – Hoffman code [164-869]

2.2.1.1 – Reading of the model

At the beginning new variables were declared [178-237], then the program begins to read material information needed in failure indexes calculation. This operation is repeated for the number of materials created in the model("*nMat"*) [240 - :270]:

```
'Reading Material data----------------------------------------------------------------------------------------------------
fmpOut.setID = vOCids(0)
OutID = 1000020
rc = -1
nPlies = 0
Do While rc = -1
rc = fmpOut.Exist(OutID)
If rc = -1 Then
OutID += 200
nPlies += 1
End If
Loop
rc = fmpMatSet.AddAll(zDataType.FT_MATL)
rc = fmpMatSet.GetArray(nMat, matIDs)
ReDim matS1c(nMat - 1)
ReDim matS2c(nMat - 1)
ReDim matS1t(nMat - 1)
ReDim matS2t(nMat - 1)
ReDim matT12(nMat - 1)
ReDim matG12(nMat - 1)
For i = 0 To nMat - 1
rc = fmpMat.Get(matIDs(i))
matS1c(i) = fmpMat.CompressionLimit1
matS2c(i) = fmpMat.CompressionLimit2
matS1t(i) = fmpMat.TensionLimit1
matS2t(i) = fmpMat.TensionLimit2
matT12(i) = fmpMat.ShearLimit
matG12(i) = fmpMat.Gx
Next i
```

After that, for the entire model or a group of it some information about “pcomp elements” were imported and saved in different “vectors”: number of elements, IDs of each element, property ID of the element, type, shape of it…[274-342]

```
'FULL MODEL----------------------------------------------------------------------------------------------------------------
If Me.EntireMod.Checked = True Then
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "FULL MODEL MOD")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, " ")
rc = fmpElSet.AddRule(21,zGroupDefinitionType.FGD_ELEM_BYTYPE) 'aggiungo elementi pcomp ordine
1mo
rc = fmpElSet.AddRule(22, zGroupDefinitionType.FGD_ELEM_BYTYPE) 'aggiungo elementi pcomp ordine
2Do
'Reading and Saving MODEL DATA---------------------------------------------------------------------------------------
rc = fmpEl.GetAllArray(fmpElSet.ID, nEl, ElIds, propID, elType, topology, Lyr, color, formulation, orient,
offset, release, orientSET, orientID, nodes, conTYPE, conSEG)
'-------------------------------------------------------------------------------------------------------------------
ElseIf Me.GroupMod.Checked = True Then
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, " ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "GROUP MOD")
fmpGroup.Get(grID)
s = fmpGroup.List(8)
'Reading and Saving MODEL DATA---------------------------------------------------------------------------------------
rc = fmpEl.GetAllArray(s.ID, nEl, ElIds, propID, elType, topology, Lyr, color, formulation, orient, offset,
```

```
release, orientSET, orientID, nodes, conTYPE, conSEG)
'-------------------------------------------------------------------------------------------------------------------------
End If ' Group or full mod
```

Now, matrices with material data and Hoffman´s coefficients of each elements were defined and filled in [344-399]:

```
ReDim vMaxPly(nEl - 1)
Dim mS1c(nEl - 1, nPlies - 1) As Double
Dim mS2c(nEl - 1, nPlies - 1) As Double
Dim mS1t(nEl - 1, nPlies - 1) As Double
…
For i = 0 To nEl – 1
rc = fmpPr.Get(propID(i))
tmpLayID = fmpPr.layupID
tmpLayOpt = fmpPr.flagI(1)
rc = fmpPly.Get(tmpLayID)
tmpLen = fmpPly.NumberOfPlys
….
For k = 0 To tmpLen – 1
tmpIdx = Array.IndexOf(matIDs, vMatIds(k))
mS1c(i, k) = matS1c(tmpIdx)
mS1t(i, k) = matS1t(tmpIdx)
mS2c(i, k) = matS2c(tmpIdx)
mS2t(i, k) = matS2t(tmpIdx)
mT12(i, k) = matT12(tmpIdx)
mG12(i, k) = matG12(tmpIdx)

F1(i, k) = (1 / mS1t(i, k)) - (1 / mS1c(i, k))
F2(i, k) = (1 / mS2t(i, k)) - (1 / mS2c(i, k))
F11(i, k) = 1 / (mS1t(i, k) * mS1c(i, k))
F22(i, k) = 1 / (mS2t(i, k) * mS2c(i, k))
F66(i, k) = 1 / (mT12(i, k) ^ 2)
Next k
Next i
```

Afterwards, a message on monitor warning the user about the correct reading of the model and the starting of failure index evaluation.

```
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, " ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, " FINISHED READING DATA OF THE ANALYSIS MODEL")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, " ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "STARTING FAILURE INDEX EVALUATION HOFFMAN 2D")
```

1.2.1.2 - Hoffman FI and MOS calculation

In the first rows of this section of the code, the variables, which will be used only for evaluation and plotting of Hoffman failure index, were declared [407-429]

```
Dim s1(nEl - 1) As Double
Dim s2(nEl - 1) As Double
Dim t12(nEl - 1) As Double
…
```

```
Dim t12cor(nEl - 1) As Double
Dim PosElIds(nOC - 1, maxPly - 1) As Integer
```

Then, a section follows about the selection of one or more different Femap analysis studies [432-453]. For each analysis study the API executes the following operations for each *case load* selected before and each plies of the model analysed [454- 801]:

1st) Reading Stresses [465 – 526];
2nd) Failure index of each element evaluation and the iterative calculation of the maximum absolute FI for every element respect all plies. [528-536];
To run the next operation the checkbox about margin of safety must be checked,
3rd) Margin of safety of each element of is calculated also as output [538-554]
4th) Research of the minimum value of MOS in each ply and creation of minimum absolute MOS contour plot [556-573];
5th) Creation of Femap output vectors to plot and to display main results in Famap message window;[575-695];
6th) Subdivision of the minimum margins of safety basing on the materials of the plies [698-800].

```
For i = 0 To nOC - 1 ' Loop for each CASE study
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, " ")
vTitle = "----------------------------------------------------CASE STUDY " & vOCids(i) & "-----------------------------------“
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, vTitle)
…
For v = 0 To maxPly – 1
…
```

Reading stress

```
If Me.EntireMod.Checked = True Then
If ComboBox_Stress.SelectedIndex = 0 Then
'Reading Stress
s1 = Nothing
s2 = Nothing
t12 = Nothing
rc = fmpOut.Get(1000020 + 200 * v)
rc = fmpOut.GetScalarAtElemSet (fmpElSet.ID, s1)

rc = fmpOut.Get(1000021 + 200 * v)
rc = fmpOut.GetScalarAtElemSet(fmpElSet.ID, s2)

rc = fmpOut.Get(1000023 + 200 * v)
rc = fmpOut.GetScalarAtElemSet(fmpElSet.ID, t12)
ElseIf ComboBox_stressfile.SelectedIndex = 0 Then 'Import Stress
If v = 0 Then ' Selection of the stress file
rc = APP.feFileGetName("Select Stressfile", " ", "*.xlsx", True, StressFile) 'Selection of the file
excel = CreateObject("Excel.Application")
```

```
'excel.Visible = 1
End If
cartella = excel.Workbooks.Open(StressFile)
foglio = excel.ActiveWorkbook.SheetsCurSheet = foglio.Item(1)
For row = 2 To nEl + 1
s1(row - 2) = CurSheet.Cells(row, 2 +8 * v).Value
s2(row - 2) = CurSheet.Cells(row, 3 +8 * v).Value
t12(row - 2) = CurSheet.Cells(row, 5+ 8 * v).Value
Next row
End If 'Stress If importation
ElseIf Me.GroupMod.Checked = True Then
...
End If 'group or full mod
```

In both cases of full model or a group the program can read stresses directly from *Femap* output vectors of the analysis with Nastran, or, they can be imported from an external file selected by user. The final result is the creation of s1, s2 and t12 vectors corresponding to σ_1, σ_2 *and* τ_{12} stresses.

Failure index evaluation

```
For k = 0 To nEl - 1
FI_Hoffman(k) = F1(k, v) * s1(k) + F2(k, v) *s2(k) + F11(k, v) * s1(k) ^ 2 + F22(k, v) * s2(k) ^ 2 +F66(k, v) *
t12(k) ^ 2 - F11(k, v) * s1(k) * s2(k)

If v = 0 Then
FImax(k) = FI_Hoffman(k)
FImax_abs(k) = Abs(FImax(k))
ElseIf Abs(FImax(k)) < Abs(FI_Hoffman(k)) Then
FImax(k) = FI_Hoffman(k)
FImax_abs(k) = Abs(FImax(k))
End If
```

In this part the failure index of element 'k' of the ply 'v' is evaluated using Eq. 2.5-1 *Ref.1*. After, the application updates the failure index of each element in vector '*FImax'* when it is major then the own previous one saved.

Margin of safety calculation

```
'MOS EVALUATION
If Me.CheckBox_MOS.Checked = True Then
'Reading Parameters
ASF = Val(Me.ASF.Text)
DSF1 = Val(Me.DSF1.Text)
DSF2 = Val(Me.DSF2.Text)
QSF = Val(Me.QSF.Text)
SFu = Val(Me.SFu.Text)
SF = ASF * QSF * DSF1 * DSF2 * SFu
s1cor(k) = s1(k) * SF
s2cor(k) = s2(k) * SF
t12cor(k) = t12(k) * SF
a(k) = F11(k, v) * s1cor(k) ^ 2 + F22(k,v) * s2cor(k) ^ 2 - F11(k, v) * s1cor(k) * s2cor(k) + F66(k, v) * t12cor(k) ^
2
b(k) = F1(k, v) * s1cor(k) + F2(k, v) *s2cor(k)
```

```
SR(k) = (-b(k) + Sqrt((b(k) ^ 2 + 4 * a(k)))) / (2 * a(k))
MOS(k) = SR(k) – 1
```

The formulas implemented here are the same used ordinarily by company.
At the beginning five important parameters are read from user input in the GUI to obtain the safety factor used.

- ASF
- DSF = DSF1*DSF2
- QSF
- SFu

Then, stresses are multiplied by safety factor and they are used to evaluate factor *'a'* and *'b'*. These ones avail to find the strength ratio *'SR'* and finally the margin of safety of the element 'k'

Research of the minimum value of MOS in each ply and creation of minimum absolute MOS contour plot

```
'Minimum MOS vector and other information of the ply v-esima-
If k = 0 Then
MinhoffMosply(v) = MOS(k)
MinhoffMosplyElID(i, v) = ElIds(k)
ElseIf MinhoffMosply(v) > MOS(k) Then
MinhoffMosply(v) = MOS(k)
MinhoffMosplyElID(i, v) = ElIds(k)
PosElIds(i, v) = k
End If
'-------------------------------------------------------------------------------
'Minimums MOS of all plies for each element-------------------------
If v = 0 Then
MinMOScontour(k) = MOS(k)
ElseIf MinMOScontour(k) > MOS(k) Then
MinMOScontour(k) = MOS(k)
End If
'-------------------------------------------------------------------------------
```

This commands allow to save in variable *'MinhoffMosply'* the minimum margin of safety of each ply between all its elements. Besides, the ID of the element of the ply with the minimum value is saved in *'MinhoffMosplyElID'*. Finally, the output vector *'MinMOScontour'* is created, which is used to create *Femap contour plot* of the absolute minimum MOS of each element between all plies.

Creation of Femap output vectors to plot and to display main results in Famap message window

```
If k = nEl - 1 Then
If Me.EntireMod.Checked = True Then
vTitle = "Lam Ply" & v + 1 & "HOFFMAN MOS INDEX (FULL MODEL)"
rc = fmpOut.InitScalarAtElem(vOCids(i), 9519221 + v, vTitle, 0, True)
rc = fmpOut.PutScalarAtElem(nEl,ElIds, MOS)
rc = fmpOut.Put(-1)
```

```
vTitle = "Minimum MOS of ply" & v+ 1 & "(FULL MODEL): " & MinhoffMosply(v)
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, vTitle)
ElseIf Me.GroupMod.Checked = True Then
vTitle = "Lam Ply" & v + 1 & "HOFFMAN MOS INDEX GROUP" & grID
rc = fmpOut.InitScalarAtElem(vOCids(i), 8999000 + 1000 * grID + v, vTitle, 0, True)
rc = fmpOut.PutScalarAtElem(nEl,ElIds, MOS)
rc = fmpOut.Put(-1)
vTitle = "Minimum MOS of ply" & v+ 1 & "for GROUP ID " & grID & ": " & MinhoffMosply(v)
APP.feAppMessage (zMessageColor.FCM_HIGHLIGHT, vTitle)
End If
End If
End If ' MOS EVALUATION CHECK
Next k 'NEXT ELEMENT
```

```
If Me.EntireMod.Checked = True Then
vTitle = "Lam Ply" & v + 1 & " HOFFMAN FAILINDEX (FULL MODEL)"
rc = fmpOut.InitScalarAtElem(vOCids(i), 9520220 + v, vTitle, 0, True)
rc = fmpOut.PutScalarAtElem(nEl, ElIds,FI_Hoffman)
rc = fmpOut.Put(-1)

If v = maxPly - 1 Then
vTitle = "HOFFMAN MAX FAIL INDEX (WITHSIGN) (FULL MODEL)"
rc = fmpOut.InitScalarAtElem(vOCids(i),9521220, vTitle, 0, True)
rc = fmpOut.PutScalarAtElem(nEl, ElIds,FImax)
rc = fmpOut.Put(-1)
vTitle = "HOFFMAN MAX FAIL INDEX (ABS) (FULL MODEL) "
rc = fmpOut.InitScalarAtElem(vOCids(i),9521221, vTitle, 0, True)
rc = fmpOut.PutScalarAtElem(nEl, ElIds,FImax_abs)
rc = fmpOut.Put(-1)
If Me.CheckBox_MOS.Checked = True Then
vTitle = "HOFFMAN MINIMUMS RESPECT ALL PLY- MOS - (FULL MODEL) "
rc = fmpOut.InitScalarAtElem(vOCids(i), 9521222, vTitle, 0, True)
rc = fmpOut.PutScalarAtElem(nEl,ElIds, MinMOScontour)
rc = fmpOut.Put(-1)

Dim MinMOSLaminate(nOC - 1) As Double
MinMOSLaminate(i) = MinMOScontour(0)
Dim MinMOSLaminateElID(nOC - 1) As Double
MinMOSLaminateElID(i) = ElIds(0)
For k = 0 To nEl - 1
If MinMOSLaminate(i) > MinMOScontour(k) Then
MinMOSLaminate(i) = MinMOScontour(k)
MinMOSLaminateElID(i) = ElIds(k)
End If
Next k

If i = 0 Then
MinAbsoluteMosLaminate =MinMOSLaminate(0)
MinAbsoluteMosLaminateCASE =vOCids(i)
MinAbsoluteMosLaminateElID =MinMOSLaminateElID(i)
ElseIf MinMOSLaminate(i) <MinAbsoluteMosLaminate Then
MinAbsoluteMosLaminate =MinMOSLaminate(i)
```

```
MinAbsoluteMosLaminateCASE =vOCids(i)
MinAbsoluteMosLaminateElID =MinMOSLaminateElID(i)
End If
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, " ")
vTitle = "Minimum MOS of Laminate (FULL MODEL): " & MinMOSLaminate(i)
APP.feAppMessage (zMessageColor.FCM_HIGHLIGHT, vTitle)
End If
End If
ElseIf Me.GroupMod.Checked = True Then
...
End If ' Group or full mod
Next v
```

Subdivision of the minimum margins of safety basing on the materials of the plies

With commands below for each material of the model is determined the minimum value of margin of safety and in which ply and element is verified. In the evidence code few beginning and final rows are reported.

```
'Subdivision of MIN MOS basing on material-------------------------------------------------------------------
If CheckBox_MOS.Checked = True Then
Dim cont1 As Integer
Dim cont2 As Integer
Dim diff As Integer
Dim contmat As Integer
...
vTitle = "Minimum MOS of Material: " & matid(contmat) & "for GROUP ID " & grID & ": " &
MOSminMat(contmat)
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, vTitle)
vTitle = "IN PLY: " & MOSminMatPlyID(contmat)
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, vTitle)
vTitle = "In the element ID : " &MinhoffMosplyElID(i, MOSminMatPlyID(contmat) - 1)
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, vTitle)
End If 'Group or full mod
Next contmat
End If '-----------------------------------------------------------------------------------------------------------
Next i 'Next CASE SET
```

Eventually, if MOS check option was active, the minimum of the margins between all case load is obtained with following commands [803-843]. It is also reported the stress state that leads to it.

```
'MINIMUM MOS of all load case-----------------------------------
If CheckBox_MOS.Checked = True Then
ReDim MINMOSMATLC(nMatlam - 1)
ReDim MINMOSMATLCcase(nMatlam - 1)
ReDim MINMOSMATLCplyID(nMatlam - 1)
ReDim MINMOSMATLCelID(nMatlam - 1)
...
For contmat = 0 To nMatlam - 1
If MinAbsoluteMosLaminate = MINMOSMATLC(contmat)Then
```

```
MinAbsoluteMosLaminatePlyID =MINMOSMATLCplyID(contmat)
s1MinAbs = s1MinMatAbs(contmat)
s2MinAbs = s2MinMatAbs(contmat)
t12MinAbs = t12MinMatAbs(contmat)
End If
Next contmat
End If 'MOD checkbox
'-------------------------------------------------------------------------
```

2.2.2 – LaRC05 code [870-1709]

This code calculates failure indexes of LaRC05 theory, with the hypothesis of unidirectional composites (UD).

2.2.2.1 – Reading of model information

Like in Hoffman algorithm, news variables were firstly defined [877-952]. It is important to underline that the code evaluates the matrix failure for different values of angle α: 15°, 30°, 45°, 60°, 75°, 90°, 105°, 120°, 135°, 150°, 165°. So, to make easier the code the vectors with their *cos*, *sin*, cos^2 and sin^2 were created.

```
…
Dim cosa() As Double = {1, 0.965925826, 0.866025404, 0.707106781, 0.5, 0.258819045, 6.12574E-17,
-0.258819045, -0.5, -0.707106781, -0.866025404, -0.965925826}
Dim cos2a() As Double = {1, 0.866025404, 0.5, 6.12574E-17,-0.5, -0.866025404, -1, -0.866025404, -0.5,
-0.00000000000000183772, 0.5, 0.866025404}
Dim sina() As Double = {0, 0.258819045, 0.5, 0.707106781,0.866025404, 0.965925826, 1, 0.965925826,
0.866025404,0.707106781, 0.5, 0.258819045}
Dim sin2a() As Double = {0, 0.5, 0.866025404, 1, 0.866025404, 0.5, 0.00000000000000122515, -0.5, -
0.866025404, -1, -0.866025404, -0.5}
Dim cosa2() As Double = {1, 0.933012702, 0.75, 0.5, 0.25, 0.066987298, 3.75247E-33, 0.066987298, 0.25,
0.5, 0.75, 0.933012702}
Dim sina2() As Double = {0, 0.066987298, 0.25, 0.5, 0.75, 0.933012702, 1, 0.933012702, 0.75, 0.5, 0.25,
0.066987298}
…
```

Successively, the api evaluates the number of plies of the laminate reading the number of existing output vector, and it reads material information needed in failure indexes calculation. This operation is repeated for the number of materials created in the model("*nMat"*) [954 - :996]:

```
'Reading Material data-------------------------------------------
fmpOut.setID = vOCids(0)
OutID = 1000020
rc = -1
nPlies = 0
Do While rc = -1
rc = fmpOut.Exist(OutID)
If rc = -1 Then
OutID += 200
nPlies += 1
```

```
End If
Loop
...
For i = 0 To nMat - 1
rc = fmpMat.Get(matIDs(i))
matS1c(i) = fmpMat.CompressionLimit1
matS2c(i) = fmpMat.CompressionLimit2
matS1t(i) = fmpMat.TensionLimit1
matS2t(i) = fmpMat.TensionLimit2
matT12(i) = fmpMat.ShearLimit
matG12(i) = fmpMat.Gx
...
Next i
'--------------------------------------------------------------------------
```

Finally, for the entire model or a group of it some information about "pcomp elements" were imported and saved in the same way used for Hoffman [998 - 1151] and a message on video warning the user about the correct reading of the model and the starting of failure index evaluation.

```
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, " ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "FINISHED READING ANALYSED MODEL DATA - LaRC05")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, " ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "STARTING FAILURE INDEX EVALUATION - LaRC05")
```

1.2.2.2 – LaRC05 failure index calculation

At the beginning of this section, the variables, which will be used for evaluation and plotting of LaRC05 failure indexes, were declared [1158 - 1203]:

```
'Variable definition needed to LaRC05 Failure Index calulation----------------
Dim s1(nEl - 1) As Double
Dim s2(nEl - 1) As Double
Dim t12(nEl - 1) As Double
...
Dim a0 As Double, nT As Double, nL As Double
Dim Kst As Double, Knl As Double
Dim I6 As Double = 1 / 6
Dim I3 As Double = 1 / 3
Dim I2 As Double = 0.5
'-------------------------------------------------------------------------------------------
```

Then the reading of $\alpha_0$value from input user follows. If it is inserted a negative value the program uses default value of 53°, that was often reported in references about failure in composites material.

Besides, the parameter η^T is calculated by *Eq. 2.5.3-2* in *Ref.1:*

$$\eta_T = -\frac{1}{\tan(2\alpha_0)}$$

```
'Definition Alpha 0
If Val(Me.txtalpha0.Text) > 0 Then
a0 = PI / 180 * Val(Me.txtalpha0.Text)
```

```
Else
APP.feAppMessage(4, "Alpha0 invalid value. 53deg is used instead")
a0 = PI / 180 * Val(53)
End If
nT = -1 / Tan(2 * a0)
Kst = Cos(a0) * (Sin(a0) + Cos(a0) / Tan(2 * a0))
Knl = -Cos(2 * a0) / (Cos(a0)) ^ 2
```

Then, a '*while cycle'* about the selection of one or more different femap analysis studies begins [1218 – 1236].

```
'Reading the different analysis study----------------------------------------------------------------------------------------
Dim numb As Long
Dim MsgAStudy As String
'Dim AMngr As femap.AnalysisMgr
'Set AMngr = App.feAnalysisMgr
'Create Analysis Study Object
Dim AStudy As femap.AnalysisStudy
AStudy = APP.feAnalysisStudy
'Loop through all Analysis Studies and Print Number of
Results Sets in Each
While AStudy.Next()
'Activate each Analysis Study as you loop through
AStudy.Active = AStudy.ID
numb = AStudy.CountOutputSets()
MsgAStudy = "Analysis Study: " + Str$(AStudy.ID) + ", has" + Str$(numb) + " Results Sets"
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, " ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, MsgAStudy)
APP.feAppStatusShow(True, nOC)
```

For each analysis study the api executes the following operations for <u>each *case load*</u> selected before and <u>each plies</u> of the model analysed [1271 - 1425]:

1st) Reading Stresses [1280 – 1388];
2nd) Failure index of <u>each element</u> evaluation and the iterative calculation of the maximum absolute FI for every element respect all plies. [1390 - 1465];

```
For i = 0 To nOC - 1 ' Loop fo each CASE study
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, " ")
vTitle = "-------------------------------------------------CASE STUDY " & vOCids(i) &"----------------------------"
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, vTitle)
fmpOut.setID = vOCids(i)
APP.feAppStatusUpdate(i + 1)
APP.feAppStatusRedraw()
```

Reading stresses

In both cases of full model or a group the program can read stresses directly from *Femap* output vectors of the analysis with Nastran, or, they can be imported from an external file selected by user when the application evaluates the first ply.
The final result is the creation of s1, s2, t12, t23, t31, S1p and S2p vectors corresponding to stresses of *Femap* output vector in *Fig.1*.

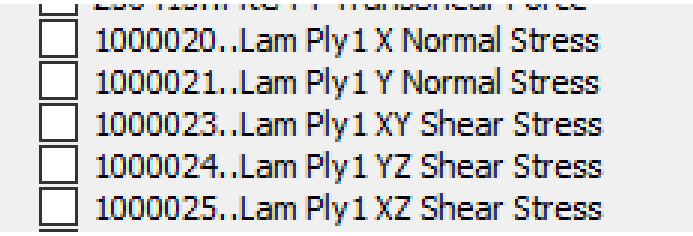

Figure 2 - Stresses reading by the api

```
For v = 0 To maxPly - 1
If Me.EntireMod.Checked = True Then
If ComboBox_Stress.SelectedIndex = 0 Then
'Reading Stress
s1 = Nothing
s2 = Nothing
t12 = Nothing
t23 = Nothing
t31 = Nothing
rc = fmpOut.Get(1000020 + 200 * v)
rc = fmpOut.GetScalarAtElemSet(fmpElSet.ID, s1)
rc = fmpOut.Get(1000021 + 200 * v)
rc = fmpOut.GetScalarAtElemSet(fmpElSet.ID, s2)
rc = fmpOut.Get(1000023 + 200 * v)
rc = fmpOut.GetScalarAtElemSet(fmpElSet.ID, t12)
rc = fmpOut.Get(1000024 + 200 * v)
rc = fmpOut.GetScalarAtElemSet(fmpElSet.ID, t23)
rc = fmpOut.Get(1000025 + 200 * v)
rc = fmpOut.GetScalarAtElemSet(fmpElSet.ID, t31)
ElseIf ComboBox_stressfile.SelectedIndex = 0 Then 'Import Stress
...
If v = 0 Then ' Selection of the stress file
rc = APP.feFileGetName("Select Stressfile", " ", "*.xlsx", True, StressFile) 'Selection of the file
excel = CreateObject ("Excel.Application")
'excel.Visible = 1
End If
cartella = excel.Workbooks.Open(StressFile)
foglio = excel.ActiveWorkbook.Sheets
CurSheet = foglio.Item(1)
For row = 2 To nEl + 1
s1(row - 2) = CurSheet.Cells(row, 2 +7 * v).Value
s2(row - 2) = CurSheet.Cells(row, 3 +7 * v).Value
t12(row - 2) = CurSheet.Cells(row, 4 + 7 * v).Value
t23(row - 2) = CurSheet.Cells(row, 5 + 7 * v).Value
t31(row - 2) = CurSheet.Cells(row, 6 + 7 * v).Value
Next row
```

```
End If 'Stress If
ElseIf Me.GroupMod.Checked = True Then
If ComboBox_Stress.SelectedIndex = 0 Then
'Reading Stress
...
ElseIf ComboBox_stressfile.SelectedIndex = 0 Then 'Import Stress
...
End If 'Stress If
End If
```

Failure index evaluation

In this section the process was described to evaluate the three failure indexes of LaRC05 theory. The *Eqs.LaRC05-1, LaRC05-2 and LaRC05-3* in *Ref. 1* were implemented:

$$FI_M = \left(\frac{\tau_T}{S_T^{is} - \eta_T \sigma_N}\right)^2 + \left(\frac{\tau_L}{S_L^{is} - \eta_L \sigma_N}\right)^2 + \left(\frac{\langle \sigma_N \rangle_+}{Y_T^{is}}\right)^2 \qquad (LaRC05-1)$$

$$FI_{KINK} = FI_{SPLIT} = \left(\frac{\tau_{23}^m}{S_T^{is} - \eta_T \sigma_2^m}\right)^2 + \left(\frac{\tau_{12}^m}{S_L^{is} - \eta_L \sigma_2^m}\right)^2 + \left(\frac{\langle \sigma_2^m \rangle_+}{Y_T^{is}}\right)^2 \quad (LaRC05-2)$$

$$FI_{FT} = \frac{\langle \sigma_1 \rangle_+}{X_T} \qquad (LaRC05-3)$$

Moreover, polymer mode failure was implemented, but it was not useful and so it was commented.

For every ply the code controls for each element what failure index it is highest and it saves it in *'FImax'*.

Besides, at each the failure mode was assigned a number:

- 0 = Fiber Tension
- 2 = Matrix Cracking (tension or compression)
- 3 = Kinking Mode

And this information about failure mode corresponding to maximum failure index in each element is saved in *'FImod_max'*.

To avoid misunderstanding, it is reminded to the reader that the failure index of each mode is evaluated in each element for each ply.

Fiber tension mode

```
For k = 0 To nEl - 1
FI = -1
s1pos = 0
FImax(k) = -1
FImod_max(k) = -1
If vMaxPly(k) > v Then
'Fiber tension mode
FImod_max(k) = 0
If s1(k) > 0 Then
```

```
s1pos = s1(k)
FI_Tension(k) = s1pos * mS1tI(k, v)'Verified
FImax(k) = FI_Tension(k)
```

Where *mS1tI(k, v)* = $\frac{1}{X^T}$ of the element.

Matrix mode

```
'Matrix mode
YT = mS2t(k, v)
iYT = 1 / YT
ST = mS2c(k, v) * Kst
SL = mT12(k, v)
nL = -SL * Cos(2 * a0) / mS2c(k, v) / Cos(a0) ^ 2

For r = 0 To 11
sn = (s2(k) + s3) * 0.5 + (s2(k) -s3) * 0.5 * cos2a(r) + t23(k) * sin2a(r)
tt = -(s2(k) - s3) * 0.5 * sin2a(r) +t23(k) * cos2a(r)
tl = t12(k) * cosa(r) + t31(k) * sina(r)
snpos = 0
If sn > 0 Then snpos = sn
FI_matrix(k, r) = (tt / (ST - nT * sn)) ^ 2 + (tl / (SL - nL * sn)) ^ 2 + (snpos * iYT) ^ 2

If FImax(k) < FI_matrix(k, r) Then
FImax(k) = FI_matrix(k, r)
FImod_max(k) = 2
End If
Next r
```

The strength S^L and S^T in matrix mode equations are 'in-situ' and they are obtained by table in *Fig. 17* in *Ref.1*. *'Kst'* was already evaluated in part where α_0 was defined. While, S^L was assumed to be equal to the shear limit in Femap definition of an orthotropic 2D material.

Kinking mode

```
'Kinking mode
iXc = mS1cI(k, v)
Xc = mS1c(k, v)
phic = Atan((1 - Math.Sqrt(1 - 4 * (SL * iXc + nL) * SL * iXc)) / (2 * (SL * iXc + nL)))
'gamma1m2mC = phic * Xc / mG12(k, v)
' phi0 = phic - gamma1m2mC
psi = Atan(a0 * t23(k) / (s2(k) - s3)) /2 'Angle that define the plane where klinking takes place
cospsi = Math.Cos(psi)
sinpsi = Math.Sin(psi)
t12psi(k) = t12(k) * cospsi + t31(k) *sinpsi
abst12psi = Math.Abs(t12psi(k))
abst12 = Math.Abs(t12(k))
phi = t12psi(k) / abst12psi * (abst12 + (mG12(k, v) - Xc) * phic) / (mG12(k, v) + s1(k) + s2(k))
cosphi = Math.Cos(phi)
sinphi = Math.Sin(phi)
s2psi(k) = cospsi ^ 2 * s2(k) + sinpsi ^2 * s3 + 2 * sinpsi * cospsi * t23(k)
t23psi(k) = -sinpsi * cospsi * s2(k) +sinpsi * cospsi * s3 + (cospsi ^ 2 - sinpsi ^ 2) * t23(k)
```

```
t31psi(k) = t31(k) * cospsi - t12(k) *sinpsi
s2m = s1(k) * sinphi ^ 2 + s2psi(k) *cosphi ^ 2 - 2 * t12psi(k) * sinphi * cosphi
t12m = -s1(k) * sinphi * cosphi + s2psi(k) * cosphi * sinphi + t12psi(k) * (cosphi ^ 2 - sinphi ^2)
t23m = t23psi(k) * cosphi - t31psi(k) *sinphi

s2mpos = 0
If s2m > 0 Then s2mpos = s2m

FI_klink(k) = (t23m / (ST - nT * s2m)) ^2 + (t12m / (SL - nL * s2m)) ^ 2 + (s2mpos * iYT) ^ 2

If FImax(k) < FI_klink(k) Then
FImax(k) = FI_klink(k)
FImod_max(k) = 3
End If
End If
Next k
```

In LaRC05 kinking mode of failure considers a 3D stress state, so to use it and not LaRC04 an important assumption was done: 's3' vector (σ_3 stress) was declared as null.
In fact, Nastran analysis to laminate element does not return this stress.
σ_3 stress compare in expression to evaluate angle ψ, σ_2^{ψ} and τ_{23}^{ψ}.

1.2.2.3 – Saving of output vectors in Femap and creation of their contours

Finally after that all FIs of the ply 'v' were calculated they are saved in many output vector in Femap, in this way they can be plotted as contour. An example of the commands used is in the code below.

```
'SAVING OUTPUT VECTORS----------------------------------------------------------------------------------------------------
If Me.EntireMod.Checked = True Then
vTitle = "Lam Ply" & v + 1 & " L05 FAILMAX INDEX"
rc = fmpOut.InitScalarAtElem(vOCids(i),9521223 + v, vTitle, 0, True)
rc = fmpOut.PutScalarAtElem(nEl, ElIds,FImax)
rc = fmpOut.Put(-1)
For r = 0 To nEl - 1
vFImod_max(r) = FImod_max(r)
Next r
…
vTitle = "Lam Ply" & v + 1 & " L05 FAIL INDEX tension Mode"
rc = fmpOut.InitScalarAtElem(vOCids(i), 9318041 + 1000 * grID + v, vTitle, 0, True)
rc = fmpOut.PutScalarAtElem(nEl, ElIds, FI_Tension)
rc = fmpOut.Put(-1)
End If ' group or full mod----------------------------------------------------------------------------------------------------
Next v 'Next PLY
Next i 'Next CASE SET
End While 'Next Analysis study
```

In *Fig.4* are listed all output vectors from Femap.

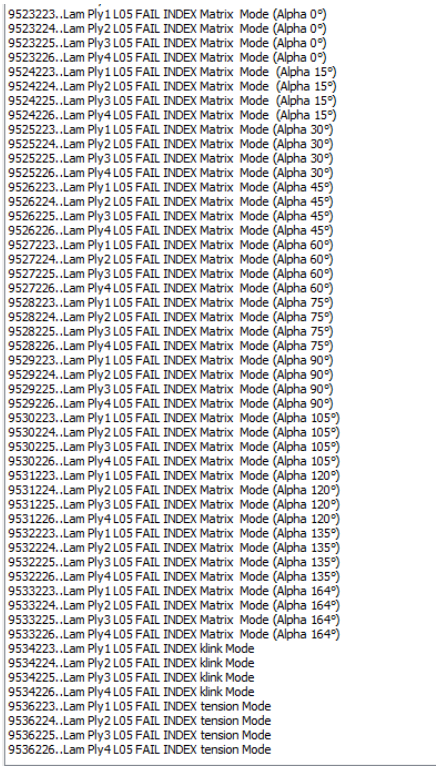

Figure 3 - Output vector in Femap after LaRC05 FIs evalutation

2.3 - Sandwich Panel Margin of Safety Calculation

The second main function of the API is the calculation of many margins of safety, which has been usually evaluated to analyse failure condition for sandwich panels using engineering best practice. They were organized in that relative skins and that for honeycomb.
In the same way already descripted for the previous operative mode, at the beginning the user must select the source of file of the stresses and what to analyse (the entire model or just a group of it). Moreover at least one mode of buckling must be chosen, else the following error message will appear on the Femap window:

```
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "ERROR: Choose at least one buckling form!!")
```

Then from row 2041 to 2088 has been defined some new variables, for example:

```
…
Dim QSF As Double
Dim DSF1 As Double
Dim DSF2 As Double 'DSF is divided usually in 2 parts
Dim ASF As Double
Dim SFu As Double
Dim SFy As Double
Dim vTitle As String
…
```

After similarly as in the previous function the API reads basic data of the model needed to the calculation. In this case also “pshell” elements has been read.

```
rc = fmpElSet.AddRule(17,zGroupDefinitionType.FGD_ELEM_BYTYPE) 'aggiungo elementi pshell ordine 1mo
rc = fmpElSet.AddRule(18,zGroupDefinitionType.FGD_ELEM_BYTYPE) 'aggiungo elementi pshell ordine 2Do
```

The next step [2222-2239] is the selection of the load case using and the definition of other variables:

```
'LOAD CASE SELECTION----------------------------------------------------------------------------
nOC = Nothing
vOCids = Nothing
APP.feSelectOutputSets("Select LOAD CASEs to reading",
fmpCASEset)
rc = fmpCASEset.GetArray(nOC, vOCids)
'-----------------------------------------------------------------------------------------------------------------------
Dim mEc(nEl - 1, 0) As Double
Dim mnuxy(nEl - 1, 0) As Double
Dim Skth(nEl - 1, 0) As Double 'Skin thickness
Dim t23(nEl - 1) As Double
Dim t13(nEl - 1) As Double
Dim MinorPrincStress(nEl - 1) As Double
Dim VonMisesStress(nEl - 1) As Double
Dim PLATEelNum As Double
Dim PLATEelIDs() As Double
PLATEelNum = 0
```

At this point the program executes different operation if the structure was modelled by laminate, plate or solid elements.

2.3.1 - Skins: Laminate and Plate Elements

```
...
If CheckBox_Solid.Checked = False Then
cont = 0

If elType(0) = 21 Or elType(0) = 22 Then ' ALL the group MUST HAVE LAMINATED ELEMENTS!!!!!!
Me.ComboBox_elementstype.SelectedIndex = 1
fmpOut.setID = vOCids(0)
OutID = 1000020
rc = -1
nPlies = 0

Do While rc = -1
rc = fmpOut.Exist(OutID)
 rc = -1 Then
OutID += 200
nPlies += 1
End If
Loop
ReDim mEc(nEl - 1, nPlies - 1)
ReDim mnuxy(nEl - 1, nPlies - 1)
ReDim Skth(nEl - 1, 1) 'Skin thickness
ReDim vMaxPly(nEl - 1)
End If
...
For i = 0 To nEl - 1
If elType(i) = 21 Or elType(i) = 22 Then
rc = fmpPr.Get(propID(i))
...
ElseIf elType(i) = 17 Or elType(i) = 18 Then
Dim MatIDel As Integer
maxPly = 1
rc = fmpPr.Get(propID(i))
MatIDel = fmpPr.matlID
Skth(i, 0) = fmpPr.pval(0)
mEc(i, 0) = matEc(Array.IndexOf(matIDs, MatIDel))
mnuxy(i, 0) = matnu(Array.IndexOf(matIDs, MatIDel))
PLATEelNum = PLATEelNum + 1
cont = cont + 1
End If 'eltype
Next i
End If ' Solid check false
....
```

The rows above [2475-2559] allow to read from Femap specific material distribution, thickness and number of plies for laminate or plate elements.

Then, similarly to algorithms of previous chapters for each single analysis study have been run the following commands. At the beginning new variables used in this part have been defined. [2583-2636].

After for both type of elements the API begins to calculate engineering best practice margin of safety checked in the GUI[1]:

[1] Laminates - [2647-3193], Plate - [3194-3673]

1. Dimpling
2. Wrinkling
3. Tension Yielding
4. Compression Yielding
5. Ultimate tension

These ones have been evaluated for each load case for each element of the ply and for each ply.

Moreover, the overall minimum MOS is researched by the code and available by the user.

To avoid an useless repetition for the aim of this document, the procedure has been only reported and described for case of laminates, the other is available in *Ref.2*.

Firstly, the API read data input of 'skins ID', which have as default value 0 and 2 for sandwich structure like in *Fig.5*.

But sometimes panels have been modelled with many plies of reinforce, so the user needs to select the ID of the ply with direct contact with the honeycomb.

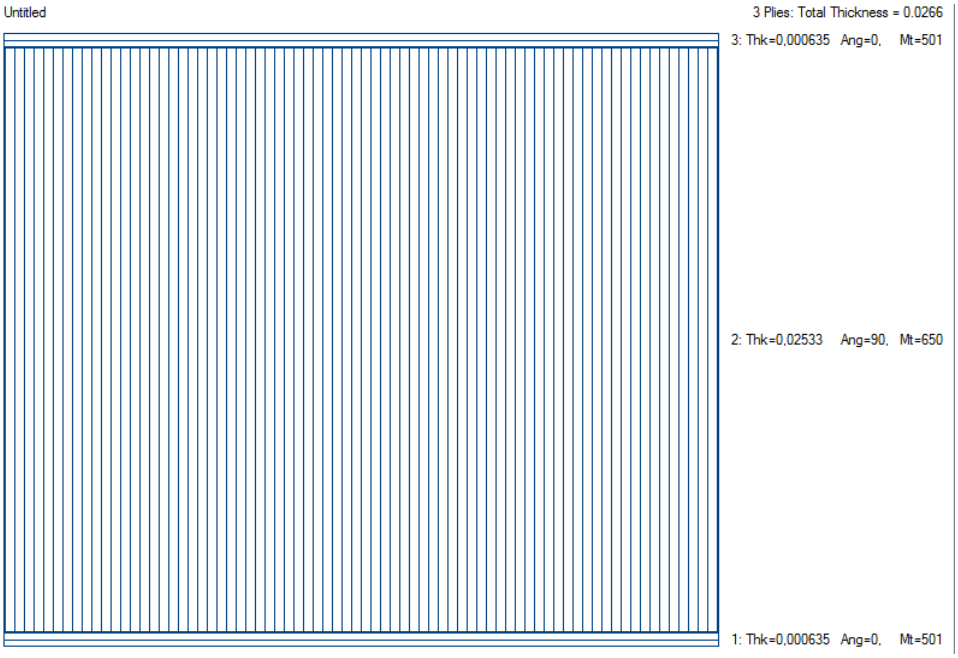

Figure 4 - Sandwich layup defualt case

```
…
Plyskin1 = Val(Me.Plyskin1.Text)
Plyskin2 = Val(Me.Plyskin2.Text)
…
```

Then the stress state has been read from the source selected by the user[2] analogously to other function [2659-2731].

[2] For plate laminate Femap stresses are only usable. The importing from another source wasn't implemented yet.

2.3.1.1 - Dimpling

To execute routine for dimpling [2734-2855] the user must insert the parameter “Cell Dim”. The evaluation of dimpling MoS has been separately run for upper skin and bottom skin. After that the value for each element of ply ‘v’ has been found, the minimum of the entire ply has been update and it has been saved in matrix *‘MinDimplMos’*. Also the element ID, where the minimum was verified, it was saved. Finally the minimum respect all load case, which is contained in *‘MINABSOLUTEMosDimpl’*, was updated.

```
If Me.CheckBox_dimpling.Checked = True Then
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "  ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "Starting Dimpling calcualtion")
Dim DebLength(nEl - 1) As Double 'Bonded Length
Dim DimpStressCrit(nEl - 1) As Double 'Skin dimpling critical stress
Dim DimpStressDebonded(nEl - 1) As Double 'Skin dimpling critical stress withd ebonded stress
...
Celldim = Val(Me.CellDim.Text)
For k = 0 To nEl - 1
DebLength(k) = 3 * Celldim
Skth(k, 0) = fmpPly.thickness(Plyskin1)
Skth(k, 1) = fmpPly.thickness(Plyskin2)
If v = Plyskin1 Then 'SKIN 1
DimpStressCrit(k) = (2 * Sqrt(mEc(k, v) * mEc(k, v + 2)) / (1 - mnuxy(k, v) * mnuxy(k, v + 2))) * (Skth(k, 0) / Celldim)
^ 2
DimpStressDebonded(k) = (2 * Sqrt(mEc(k, v) * mEc(k, v + 2)) / (1 - mnuxy(k, v) * mnuxy(k, v + 2))) * (Skth(k, 0) /
DebLength(k)) ^ 2
Else 'SKIN 2 +
DimpStressCrit(k) = (2 * Sqrt(mEc(k, v) * mEc(k, v - 2)) / (1 - mnuxy(k, v) * mnuxy(k, v - 2))) * (Skth(k, 1) / Celldim)
^ 2
DimpStressDebonded(k) = (2 * Sqrt(mEc(k, v) * mEc(k, v - 2)) / (1 - mnuxy(k, v) * mnuxy(k, v - 2))) * (Skth(k, 1) /
DebLength(k)) ^ 2
End If

ComprPrincStress(k) = Abs(MinorPrincStress(k))
DimplMOS(k) = (DimpStressDebonded(k) / (ComprPrincStress(k) * QSF * DSF1 * DSF2 * ASF * SFu)) - 1
signDimplMOS(k) = (DimpStressDebonded(k) / (MinorPrincStress(k) * QSF * DSF1 * DSF2 * ASF * SFu)) - 1

'Evaluation of the minimum value in the entire ply----------------------------
If k = 0 Then
MinPrincStress = MinorPrincStress(k)
MINDimpElID(v, i) = ElIds(k)
Else
If MinPrincStress > MinorPrincStress(k) Then
MinPrincStress = MinorPrincStress(k)
MINDimpElID(v, i) = ElIds(k)
End If
End If
MaxComprPrincStress(i, v) = Abs(MinPrincStress)

'Evaluate MIN MOS of the entire ply
MinDimplMos(v, i) = (DimpStressDebonded(0) / (MaxComprPrincStress(i, v) * QSF * DSF1 * DSF2 * ASF * SFu)) - 1
'--------------------------------------------------------------------------------
Next k 'End Elements for cycle

'ABSOLUTE MINIMUM MOS OF ALL CASE SETS-----------------------------------------
If i = 0 Then
```

```
MINABSOLUTEMosDimpl(v) = MinDimplMos(v, i)
MINCASEDimp(v) = vOCids(i)
ABSOLUTEDimplElID(v) = MINDimpElID(v, i)
MaxComprPrincStressABS(v) = MaxComprPrincStress(i, v)
ElseIf MINABSOLUTEMosDimpl(v) > MinDimplMos(v, i) Then
MINABSOLUTEMosDimpl(v) = MinDimplMos(v,i)
MINCASEDimp(v) = vOCids(i)
ABSOLUTEDimplElID(v) = MINDimpElID(v, i)
MaxComprPrincStressABS(v) = MaxComprPrincStress(i, v)
End If
```

After all the following results have been displayed on Femap window:

- Dimpling Critical Stress
- Dimplimg Critical Stress with debonded Stress
- Minor Principal Stress
- Max Compression Principal Stress
- MOS Dimpling
- Dimpling Critical Stress MOS
- Dimpling Critical Stress MOS (Sign)

```
'DISPLAY RESULTS AND SAVING-------------------------------------------------------
If Me.EntireMod.Checked = True Then
vTitle = "Lam Ply" & v + 1 & " Dimpling Critical Stress (FULL MODEL): " & DimpStressCrit(0)
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, vTitle)
…
ElseIf Me.GroupMod.Checked = True Then
                        APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "Finished Dimpling calcualtion")
End If ' FINISHED DIMPLING CALCULATION of the ply n° v
```

2.3.1.2 - Wrinkling

The entire routine to evaluate margin of safety for wrinkling failure is included between lines 2857 and 2992. Also here are declared needed variables and read input data about allowable stresses.

```
If Me.CheckBox_wrinkling.Checked = True Then
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "  ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "Starting Wrinkling calcualtion")

Dim WrinkAllowStress(nEl - 1) As Double 'Wrinkling Allowable Stress
Dim Q As Double
…
…
PlyHon = Val(Me.PlyHoney.Text) 'ID PLY HONEYCOMB
Q = Val(Me.Qparam.Text)
SkTenYield = Val(Me.SkTenYieldAllStress.Text)
SkCompYield = Val(Me.SkCompYieldAllStress.Text)
SkTenUlt = Val(Me.SkTenUltAllStress.Text)
HonCompAllStress = Val(Me.HonCompAllStress.Text)
```

Then starts the cycle of MoS prediction for each element:

```
For k = 0 To nEl - 1
Skth(k, 0) = fmpPly.thickness(Plyskin1)
Skth(k, 1) = fmpPly.thickness(Plyskin2)
SkthHon = fmpPly.thickness(PlyHon)
```

The three rows above read the thickness of each skins and of the honeycomb. Then MOS is evaluated, as dimpling, for skin 1or skin 2 saving the value in *'WrinklMOS'*.

```
If v = Plyskin1 Then 'Skin 1
SkWave(k) = Skth(k, 0) * 0.1
Kparam(k) = (SkWave(k) * mEc(k, v + 1)) / (SkthHon * HonCompAllStress)
WrinkAllowStress(k) = Q * (Sqrt((mEc(k, v + 1) * Skth(k, 0)) / (mEc(k, v) * SkthHon)) * Sqrt(mEc(k, v) * mEc(k, v +
2))) / (1 + 0.64 * Kparam(k))
Else 'Skin 2
SkWave(k) = Skth(k, 1) * 0.1
Kparam(k) = (SkWave(k) * mEc(k, v - 1)) / (SkthHon * HonCompAllStress)
WrinkAllowStress(k) = Q * (Sqrt((mEc(k, v - 1) * Skth(k, 1)) / (mEc(k, v) * SkthHon)) * Sqrt(mEc(k, v) * mEc(k, v -
2))) / (1 + 0.64 * Kparam(k))
End If

ComprPrincStress(k) = Abs(MinorPrincStress(k))
WrinklMOS(k) = (WrinkAllowStress(k) / (ComprPrincStress(k) * QSF * DSF1 * DSF2 * ASF * SFu)) - 1
signWrinklMOS(k) = (WrinkAllowStress(k) / (MinorPrincStress(k) * QSF * DSF1 * DSF2 * ASF * SFu)) - 1
```

At the end minimum MoS estimate has been do in the same way of dimpling case.

```
'Evaluation of the minimum value in the entire ply----------------------------------------------------------------------
If k = 0 Then
MinPrincStress = MinorPrincStress(k)
MINWrinElID(v, i) = ElIds(k)
Else
If MinPrincStress > MinorPrincStress(k) Then
MinPrincStress = MinorPrincStress(k)
MINWrinElID(v, i) = ElIds(k)
End If
End If

MaxComprPrincStress(i, v) = Abs(MinPrincStress)

'Evaluate MIN MOS of the entire ply
MinWrinklMos(v, i) = (WrinkAllowStress(0) / (MaxComprPrincStress(i, v) * QSF * DSF1 * DSF2 * ASF * SFu)) - 1
'-----------------------------------------------------------------------------------------------------------------------
Next k 'End Elements for cycle

'ABSOLUTE MINIMUM MOS OF ALL CASE SETS----------------------------------------------------------------------------------
If i = 0 Then
MINABSOLUTEMosWrink(v) = MinWrinklMos(v, i)
MINCASEWrin(v) = vOCids(i)
ABSOLUTEWrinklElID(v) = MINWrinElID(v, i)
MaxComprPrincStressABS(v) = MaxComprPrincStress(i, v)
ElseIf MINABSOLUTEMosWrink(v) > MinWrinklMos(v, i) Then
MINABSOLUTEMosWrink(v) = MinWrinklMos(v, i)
MINCASEWrin(v) = vOCids(i)
ABSOLUTEWrinklElID(v) = MINWrinElID(v, i)
MaxComprPrincStressABS(v) = MaxComprPrincStress(i, v)
End If
```

Also the visualization method of results is equal, even if the output are now:

- Wrinkling Critical Stress
- Minor Principal Stress
- Max Compression Principal Stress
- MOS wrinkling
- Wrinkling Critical Stress MOS
- Wrinkling Critical Stress MOS (Sign)

```
'DISPLAY RESULTS AND SAVING ----------
If Me.EntireMod.Checked = True Then
vTitle = "Lam Ply" & v + 1 & " Wrinkling Allowable Stress (FULL MODEL): " & WrinkAllowStress(0)
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, vTitle)
vTitle = "Lam Ply" & v + 1 & " Minor Principal Stress (FULL MODEL): " & MinPrincStress
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, vTitle)
...
ElseIf Me.GroupMod.Checked = True Then
...
End If ' GROUP OR FULLL MOD

APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "  ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "Finished Wrinkling calcualtion")
End If ' Finish Wrinklinkg calculation of the ply n° v
```

The algorithms to calculate the next MOS [2994-3192] are the same. There is a starting a part where new variables are defined and user input data have been read from the GUI. Then MOS has been computed, and, furthermore the minimum value research. To find it each turn of the cycle was updated the maximum stress state correlated to it. Then it was used in the formula of the margin.

2.3.1.3 - Tension Yielding

```
    'TENSION YIELDING----------
If Me.CheckBox_tensionyield.Checked = True Then
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "  ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "Starting Tension Yielding calcualtion")

Dim SkTenYield As Double
Dim MOStenYield(nEl - 1) As Double
SkTenYield = Val(Me.SkTenYieldAllStress.Text)

For k = 0 To nEl - 1
MOStenYield(k) = (SkTenYield / (VonMisesStress(k) * SFy * QSF * DSF1 * DSF2 * ASF)) - 1
```

```
'Evaluation of the minimum value in the entire ply----------
If k = 0 Then
MaxVonMisesStress(i, v) = VonMisesStress(k)
MINTenYieldElID(v, i) = ElIds(k)
Else
If MaxVonMisesStress(i, v) < VonMisesStress(k) Then
MaxVonMisesStress(i, v) = VonMisesStress(k)
MINTenYieldElID(v, i) = ElIds(k)
End If
```

```
End If


'Evaluate MIN MOS of the entire ply
MinMOStenYield(v, i) = (SkTenYield / (MaxVonMisesStress(i, v) * SFy * QSF * DSF1 * DSF2 * ASF)) - 1
'-------------------------------------------------------------------------------------------------------------------------------------------
Next k 'Next element

'ABSOLUTE MINIMUM MOS OF ALL CASE SETS-----------------------------------------------------------------------------------------------------
If i = 0 Then
MINABSOLUTEMosTenyield(v) = MinMOStenYield(v, i)
MINCASETenYield(v) = vOCids(i)
ABSOLUTETenYieldElID(v) = MINTenYieldElID(v, i)
MaxVonMisesStressABS(v) = MaxVonMisesStress(i, v)
ElseIf MINABSOLUTEMosTenyield(v) > MinMOStenYield(v, i) Then
MINABSOLUTEMosTenyield(v) = MinMOStenYield(v, i)
MINCASETenYield(v) = vOCids(i)
ABSOLUTETenYieldElID(v) = MINTenYieldElID(v, i)
MaxVonMisesStressABS(v) = MaxVonMisesStress(i, v)
End If '-----------------------------------------------------------------------------------------
```

```
'DISPLAY RESULTS AND SAVING-----------------------------------------------------------------------
If Me.EntireMod.Checked = True Then
vTitle = "Lam Ply" & v + 1 & " Minimum MOS Tension Yielding: " & MinMOStenYield(v, i)
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, vTitle)

vTitle = "Lam Ply" & v + 1 & " TENSION YIELDING MOS (FULL MODEL)"
rc = fmpOut.InitScalarAtElem(vOCids(i), 9541223 + v, vTitle, 0, True)
rc = fmpOut.PutScalarAtElem(nEl, ElIds, MOStenYield)
rc = fmpOut.Put(-1)

APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "  ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "Finished Tension Yielding calcualtion")
ElseIf Me.GroupMod.Checked = True Then
...
End If
End If ' Finish tension yielding calculation of the ply n° v
```

2.3.1.4 - Compression Yielding

The code is in *Ref.2*

2.3.1.5 - Ultimate tension

The code is in *Ref.2*

2.3.2 - Honeycomb: Laminate or Solid elements

In this paragraph will be shown and commented the code to evaluate margins of safety for the honeycomb of a sandwich panel [3675- 4064]. The honeycomb model can be anaylised togheter with skins or alone.

```
                'HONEYCOMB BUCKLING----------------------------------------------------------------------------------------------------
If Me.CheckBox_honeycombBuckling.Checked = True Then
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "  ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "Starting Honeycomb Buckling Full model calcualtion")
Dim HoneyMosUltL(nEl - 1) As Double 'Margin of Safety (Ultimate, L-direction)
Dim HoneyMosUltW(nEl - 1) As Double 'Margin of Safety (Ultimate, W-direction)
Dim t13crit As Double
Dim t23crit As Double
Dim Consist23max As Double
Dim Consist13max As Double
Dim Consist23min As Double
Dim Consist13min As Double
...
```

In this case the reading of the stresses necessary in the process is a little different than before. In fact if a solid mush was used the output stresses from Nastran would be saved in different output vectors of Femap. Likewise the others functions there is the opportunity to import the sress state, and, to work on the entire model or on an itsgroup.

```
'Reading Stress----------------------------------------------------------------------------------------------
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "  ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "Reading  Honeycomb Stress ")

If CheckBox_Solid.Checked = True Then
If Me.EntireMod.Checked = True Then
If ComboBox_Stress.SelectedIndex = 0 Then 'Reading Stress
t23H = Nothing
t13H = Nothing
rc = fmpOut.Get(60014)
rc = fmpOut.GetScalarAtElemSet(fmpElSet.ID, t23H)
rc = fmpOut.Get(60015)
rc = fmpOut.GetScalarAtElemSet(fmpElSet.ID, t13H)
ElseIf ComboBox_stressfile.SelectedIndex = 0 Then   'Import Stress
...
End If 'Stress If
...
End If 'group or full mod
End If
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "  ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "Finished Reading Honeycomb Stress")
```

Subsequently maximum and minimum τ_{13} and τ_{23}, and the respective consistent stress[3]. It is saved the element ID, where the condition is verified. Now, the different on the mesh doesn't matter anymore.

```
t13crit = Val(Me.CritTauXZ_L.Text)
t23crit = Val(Me.CritTauYZ_W.Text)
Maxt13 = t13H(0)
Mint13 = t13H(0)
Maxt23 = t23H(0)
Mint23 = t23H(0)
Consist23max = t23H(0)
Consist13max = t13H(0)
Consist23min = t23H(0)
```

[3] With the term consistent is indicated the other stress that is associated to a maximum or minimum condition. For example for τ_{13} maximum, its consistenti is τ_{23}.

```
Consist13min = t13H(0)
Consist23min = Nothing
Consist13min = Nothing
For k = 0 To nEl - 1
If Maxt13 < t13H(k) Then
Maxt13 = t13H(k)
Consist23max = t23H(k)
MAX13ElID = ElIds(k)
End If
...
```

Thus, the API has all information to find MOS for each element using the following rows of code

```
For k = 0 To nEl - 1
'L Direction ---------------------------------------------------------------------
HoneyMosUltL(k) = (t13crit / (Abs(t13H(k)) * SFu * QSF * DSF1 * DSF2 * ASF)) - 1
'W Direction----------------------------------------------------------------------
HoneyMosUltW(k) = (t23crit / (Abs(t23H(k)) * SFu * QSF * DSF1 * DSF2 * ASF)) - 1
Next k
'Evaluate MIN MOS of the entire honeycomb-----------------------------------------------------------------------------------------
HoneyStress13(i) = Abs(Max(Maxt13, Abs(Mint13)))
HoneyStress23(i) = Abs(Max(Maxt23, Abs(Mint23)))
MinHoneylMosUltL(i) = (t13crit / (Abs(Max(Maxt13, Abs(Mint13))) * SFu * QSF * DSF1 * DSF2 * ASF)) - 1
MinHoneylMosUltW(i) = (t23crit / (Abs(Max(Maxt23, Abs(Mint23))) * SFu * QSF * DSF1 * DSF2 * ASF)) - 1
'--------------------------------------------------------------------------------------------------------------------------------
'Evaluate MOS for L and W combination'--------------------------------------------------------------------------------------------
HoneyMosUlt_WLTxz(i) = (1 / (((QSF * DSF1 * DSF2 * SFu * Consist23(i)) / (t23crit)) ^ 2 + ((QSF * DSF1 * DSF2 *
SFu * Abs(Max(Maxt13, Abs(Mint13)))) / t13crit) ^ 2) ^ (0.5)) - 1
HoneyMosUlt_WLTyz(i) = (1 / (((QSF * DSF1 * DSF2 * SFu * Abs(Max(Maxt23, Abs(Mint23)))) / (t23crit)) ^ 2 +
((QSF * DSF1 * DSF2 * SFu * Consist13(i)) / t13crit) ^ 2) ^ (0.5)) - 1
'--------------------------------------------------------------------------------------------------------------------------------
```

In the end, it has been determined the minimum respect al load cases for both four MOS. As example has been reported the lines of code for ‘Ultimate-L MOS’ :

```
'ABSOLUTE MINIMUM MOS OF ALL CASE SETS-------------------------------------------------------------------------------------------

If i = 0 Then
MINABSOLUTEMosUltL(v) = MinHoneylMosUltL(i)
MINCASEMosUltL(v) = vOCids(i)
ABSOLUTEMosUltLElID = L_ElID(i)
ElseIf MINABSOLUTEMosUltL(v) > MinHoneylMosUltL(i) Then
MINABSOLUTEMosUltL(v) = MinHoneylMosUltL(i)
MINCASEMosUltL(v) = vOCids(i)
ABSOLUTEMosUltLElID = L_ElID(i)
End If '-------------------------------------------------------------------------------------------
...
```

All results are screened on femap window and some of them are used to create contour plots.

```
'DISPLAY RESULTS AND SAVING-------------------------------------------------------------------------
If Me.EntireMod.Checked = True Then
If CheckBox_Solid.Checked = False Then
vTitle = "Lam Ply" & v + 1 & " 'Minimum Margin of Safety (Ultimate, L-direction) (FULL MODEL): " &
MinHoneylMosUltL(i)
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, vTitle)
...
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, " ")
APP.feAppMessage(zMessageColor.FCM_HIGHLIGHT, "Finished Honeycomb Buckling Group calcualtion")
```

```
End If ' Honeycomb buckling
```

From line 4067 to 4342 are contained the commands to print on video the absolute final results of each type of margin.

2.4 - Exportation of results

A latter function is the creation of automatic data report in Excel workbook. However, provision has been made for the possibility to export in others file formats. In a future the user could select it froma a popup menu on the GUI.

References

[1] Grasso Amedeo - Master's Degree Thesis - Implementation of classical and advanced failure criteria for composite layered structures in FEMAP and assessment of results (2018)
[2] Grasso Amedeo – API for Femap: source code (2018)