

```

1 % #####
2 % Description
3 % #####
4
5 % This function runs a parametric Finite Element Method Analysis (Abaqus
6 % 6.14) of wood bilayer gridshells in dependence of number of bilayer
7 % strips, length, thicknesses of active and passive layers, and width. The
8 % results are stored as raw data (displacements) in the matrix C. The
9 % displacements are further processed to curvatures along the x- and
10 % y-directions of the gridshell system
11
12 % Works with:
13 %   Microsoft Windows 10 Enterprise
14 %   Matlab R2016b
15 %   Abaqus/CAE 6.14-1
16
17 % Author:
18 % Philippe Grönquist (pgroenquist@ethz.ch), August 2019
19
20 % Initialize Matlab
21 clc; clear; close all;
22
23
24 % #####
25 % Define the parameters
26 % #####
27
28 % Define size of parameter space
29 Size = 11;
30
31 % Fixed parameters
32 n = 6;      % Number of strips in total, n/2 must be an odd number
33 L = 500;    % Bilayer length in mm
34
35 % Variable parameters
36 h1 = linspace(1,5,Size); % Thickness of passive layer
37 h2 = linspace(4,20,Size); % Thickness of active layer
38 w  = linspace(5,45,Size); % Width of bilayer strip
39
40
41 % #####
42 % Run analysis
43 % #####
44
45 % Create a 4D matrix, according to size of results matrix (RES),
46 % this needs to be adapted manually in terms of the fixed parameters and FE model
47 % setup in Python
48 C = zeros(Size,Size,18,4);
49
50 % Loop
51 for i = 1:Size
52     for j = 1:Size
53         % Re-arrange variables in a single vector
54         X(1) = n;
55         X(2) = L;
56         X(3) = w(i);
57         X(4) = h1(j);
58         X(5) = h2(j);
59

```

```

60     try
61
62         % Call the FE execution function
63         RES = Gridshell_Analysis(X);
64
65         % Store raw results matrix of FE analysis in a 4D matrix C
66         C(i,j, :, :) = RES;
67
68     catch Me
69
70         % If error during FE analysis, fill containers with zeros
71         C(i,j, :, :) = zeros(18,4);
72
73         % Print warning message
74         disp('#####');
75         disp(['ERROR for i = ', num2str(i), ', and j = ', num2str(j)]);
76         disp('#####');
77
78     end
79 end
80
81 end
82
83
84 % #####
85 % Calculate curvatures (for 6x6 Gridshell)
86 % #####
87
88 % Natural curvatures (from single strip FE model) in [1/m]
89 kk0 = [4.0368, 2.8834, 2.2427, 1.8349, 1.5526, 1.3456, 1.1873, 1.0623, 0.9611,
0.8776, 0.8074];
90
91 for i = 1:Size
92     for j = 1:Size
93
94         % Access results of 4D matrix
95         RES = squeeze(C(i,j, :, :));
96
97         % call curvature calculation function
98         [Ax,By,Cxy] = Gridshell_Curvature(RES,L); % in 1/mm
99
100        % curvatures along directions x and y
101        Kx(i,j) = Ax*1000; % in 1/m
102        Ky(i,j) = By*1000; % in 1/m
103
104        % Gaussian curvature
105        K(i,j) = Cxy*1000*1000; % in 1/m2
106
107        % Dimensionless curvature by thickness
108        h = (h1(j)+h2(j))./1000; % in m
109        Kdl(i,j) = K(i,j).*h^2;
110
111        % make a matrix of natural curvatures
112        k0(i,j) = kk0(j);
113
114        % Dimensionless curvature by natural curvature
115        Kdl(i,j) = K(i,j)./k0(i,j);
116
117    end
118 end

```

```

119
120 % (manuscript figure)
121 % Plot the results by using "imagesc(X)" where X is Kx, Ky, K, k0, Kdl, etc.
122
123
124 % #####
125 % Results (RAW DATA, NOT PART OF THIS SCRIPT)
126 % #####
127
128 % Gridshell curvature along x in [1/m]
129 Kx =
130
131 Columns 1 through 6
132
133     3.2150     2.7978     2.3233     1.9642     1.6825     1.4722
134     2.5765     2.4441     2.1795     1.8962     1.6574     1.4566
135     2.0974     2.2070     2.0371     1.8265     1.6220     1.4274
136     1.6125     1.9787     1.9310     1.7772     1.5873     1.4167
137     0.9150     1.7683     1.8058     1.7234     1.5565     1.4061
138     0.5771     1.6250     1.6960     1.6598     1.5369     1.3968
139     0.4141     1.5338     1.6171     1.5962     1.5115     1.3845
140     3.3261     1.4526     1.5522     1.5636     1.4785     1.3717
141     3.4062     1.4442     1.5248     1.5311     1.4518     1.3590
142     3.4313     1.4357     1.4712     1.4790     1.4269     1.3414
143     3.4426     1.3997     1.4440     1.4652     1.4124     1.3388
144
145 Columns 7 through 11
146
147     1.2985     1.1744     1.0551     0.9768     0.8989
148     1.2864     1.1620     1.0540     0.9743     0.8913
149     1.2802     1.1558     1.0517     0.9618     0.9002
150     1.2811     1.1536     1.0516     0.9700     0.8885
151     1.2682     1.1515     1.0487     0.9604     0.8891
152     1.2682     1.1469     1.0561     0.9663     0.9008
153     1.2626     1.1481     1.0568     0.9699     0.8865
154     1.2628     1.1471     1.0469     0.9638     0.8877
155     1.2611     1.1454     1.0460     0.9571     0.8944
156     1.2443     1.1362     1.0466     0.9585     0.8944
157     1.2443     1.1385     1.0452     0.9629     0.8944
158
159 % Gridshell curvature along y in [1/m]
160 ky =
161
162 Columns 1 through 6
163
164     3.2944     2.8356     2.3886     2.0182     1.7271     1.5137
165     2.6608     2.5010     2.2053     1.9262     1.6869     1.4875
166     2.2718     2.2608     2.0657     1.8666     1.6503     1.4623
167     2.1344     2.0551     1.9819     1.8002     1.6273     1.4507
168     2.5650     1.8622     1.8606     1.7479     1.5959     1.4392
169     2.8483     1.7327     1.7614     1.6884     1.5649     1.4279
170     2.9286     1.6285     1.6692     1.6290     1.5383     1.4191
171     0.0150     1.5629     1.6214     1.5983     1.5046     1.4080
172     0.1648     1.4736     1.5757     1.5677     1.5029     1.3968
173     0.1405     1.3843     1.5224     1.5233     1.4647     1.3722
174     0.1605     1.3430     1.4637     1.4950     1.4477     1.3461
175
176 Columns 7 through 11
177
178     1.3328     1.1965     1.0922     0.9932     0.9158

```

179	1.3191	1.1898	1.0808	0.9894	0.9135
180	1.3142	1.1747	1.0779	0.9866	0.9085
181	1.3004	1.1791	1.0756	0.9819	0.9099
182	1.2945	1.1761	1.0759	0.9928	0.9075
183	1.2989	1.1836	1.0789	0.9870	0.9089
184	1.2880	1.1745	1.0711	0.9882	0.9048
185	1.2779	1.1718	1.0809	0.9797	0.9162
186	1.2729	1.1787	1.0690	0.9798	0.9095
187	1.2713	1.1707	1.0668	0.9846	0.9095
188	1.2713	1.1708	1.0664	0.9812	0.9095

```

1 % #####
2 % Description
3 % #####
4
5 % This function interfaces the Gridshell analysis parameters to abaqus, runs the
6 % analysis, and retrieves the output.
7
8 % Author:
9 % Philippe Grönquist (pgroenquist@ethz.ch), August 2019
10
11
12 function Y = Gridshell_Analysis(X)
13
14 % #####
15 % Initialize
16 % #####
17
18 % Set current directory
19 cd C:\Temp
20
21 % Initialize, delete files from previous runs in case of error
22 delete('GridshellInput.txt');
23 delete('GridshellOutput.txt');
24 delete('Gridshell.odb');
25 delete('Gridshell.lck');
26
27 % Rename values from input vector for clarity, as in "Gridshell_Setup.m"
28 n = X(1);
29 L = X(2);
30 w = X(3);
31 h1 = X(4);
32 h2 = X(5);
33
34
35 % #####
36 % Interface to Abaqus
37 % #####
38
39 % Write input for Abaqus
40 data.cont= [n,...           % dataCont[0]
41             L,...          % dataCont[1]
42             w,...          % dataCont[2]
43             h1,...         % dataCont[3]
44             h2,...         % dataCont[4]
45             ];
46
47 % Define path of the necessary files
48 path = 'C:\Temp';
49
50 % Define name of the .txt input file
51 name='GridshellInput.txt';
52
53 % Call function to create txt interface file with input for python scripts
54 CreateTxt(data, name, path);
55
56
57 % #####
58 % Run the Finite element analysis
59 % #####
60

```

```

61 % (Abaqus with user subroutines (UMAT) works only when opening matlab
62 % from Intel Composer cmd!)
63
64 % Option without Abaqus GUI
65 ! abaqus cae noGUI=Gridshell_ParametricFEmodel.py
66
67 % Option with Abaqus GUI
68 ! abaqus cae script=Gridshell_ParametricFEmodel.py
69
70 % Python script waits until job is completed
71 disp('Analysis completed')
72
73
74 % #####
75 % Retrieve output of FE analysis
76 % #####
77
78 % Initialize variables.
79 filename = 'C:\Temp\GridshellOutput.txt';
80 delimiter = ' ';
81 startRow = 2;
82
83 % Format for each line of text:
84 % column1: double (%f)
85 % column2: double (%f)
86 % column3: double (%f)
87 % column4: double (%f)
88 formatSpec = '%f%f%f%f%[\n\r]';
89
90 % Open the text file.
91 fileID = fopen(filename,'r');
92
93 % Read columns of data according to the format.
94 dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter, ↵
'MultipleDelimsAsOne', true, 'HeaderLines', startRow-1, 'ReturnOnError', false, ↵
'EndOfLine', '\r\n');
95
96 % Close the text file.
97 fclose(fileID);
98
99 % Create output variable
100 Results = [dataArray{1:end-1}];
101
102
103 % #####
104 % Function output
105 % #####
106 Y = Results;
107
108 end
109
110
111 % #####
112 % Function: Generate the .txt interface file to Abaqus
113 % #####
114
115 function [] = CreateTxt(data, name, path)
116
117 % Create file and open it
118 fileName = fullfile(path,name);

```

```
119     fileID = fopen(fileName,'w');
120
121     % Header
122     fprintf(fileID,'%s\n','Values to pass');
123     fprintf(fileID,'\n');
124
125     % Print variables
126     fprintf(fileID,'%s\n','Continuous variables');
127     fprintf(fileID,'%1.12e\n',data.cont);
128     fprintf(fileID,'\n');
129
130     % Print end
131     fprintf(fileID,'%s','End');
132
133     % Close file
134     fclose(fileID);
135
136 end
```

```

1 % #####
2 % Description
3 % #####
4
5 % This function returns the curvatures along the mid-strips of the gridshell
bilayer strips,
6 % calculated from their coordinate retrieved from FE model displacements.
7 % !! This function is custom to the number of strips chosen and needs to be
8 % adapted accordingly for a number of strips in the gridshell other than
9 % 6 !!
10
11 % Author:
12 % Philippe Grönquist (pgroenquist@ethz.ch), August 2019
13
14 function [Kx,Ky,Y] = Gridshell_Curvature(X,L)
15
16 % Description of input vector X
17 % Row 1: Reference Point number
18 % Row 2: X displacement
19 % Row 3: Y displacement
20 % Row 4: Z displacement
21
22 length      = L;
23 screwdistance = length/4;
24
25
26 % #####
27 % Create matrix of strating coordinates
28 % #####
29
30 count = 1;
31 for i = 1:3
32     for j = 1:3
33
34         count = count + 2;
35
36         RPy = screwdistance*j;
37         RPx = screwdistance*i;
38
39         RP(count,1) = screwdistance*i;
40         RP(count,2) = screwdistance*j;
41
42         RP(count+1,1) = screwdistance*i;
43         RP(count+1,2) = screwdistance*j;
44
45     end
46
47 end
48
49 RP(1:2,:) = [];
50
51 % add zeros as staring points
52 RP = [RP zeros(size(X,1),1)];
53
54
55 % #####
56 % Update with displacement values from FE model
57 % #####
58
59 % End coordinates of points = starting + displacement coords from FE

```



```

60 % analysis
61 A = RP(:,1) + X(:,2);
62 B = RP(:,2) + X(:,3);
63 C = RP(:,3) + X(:,4);
64
65
66 % #####
67 % X-direction
68 % #####
69
70 % select RP 4,10,16 and 8,10,12
71 % x-direction strip
72 P1 = [A(4) B(4) C(4)];
73 P2 = [A(10) B(10) C(10)];
74 P3 = [A(16) B(16) C(16)];
75
76 % distances between the 3 points
77 a = ((P2(1)-P1(1))^2 + (P2(2)-P1(2))^2 + (P2(3)-P1(3))^2)^(1/2);
78 b = ((P3(1)-P1(1))^2 + (P3(2)-P1(2))^2 + (P3(3)-P1(3))^2)^(1/2);
79 c = ((P3(1)-P2(1))^2 + (P3(2)-P2(2))^2 + (P3(3)-P2(3))^2)^(1/2);
80
81 % radius of circle in mm
82 Rx = (a*b*c)*(2*a^2*b^2 + 2*b^2*c^2 + 2*c^2*a^2 - a^4 - b^4 - c^4)^(-1/2);
83 Kx = 1/Rx;
84
85
86 % #####
87 % Y-direction
88 % #####
89
90 % y-direction strip
91 P4 = [A(8) B(8) C(8)];
92 P5 = [A(10) B(10) C(10)];
93 P6 = [A(12) B(12) C(12)];
94
95 % distances between the 3 points
96 a = ((P5(1)-P4(1))^2 + (P5(2)-P4(2))^2 + (P5(3)-P4(3))^2)^(1/2);
97 b = ((P6(1)-P4(1))^2 + (P6(2)-P4(2))^2 + (P6(3)-P4(3))^2)^(1/2);
98 c = ((P6(1)-P5(1))^2 + (P6(2)-P5(2))^2 + (P6(3)-P5(3))^2)^(1/2);
99
100 % radius of circle in mm
101 clear a b c
102 Ry = (a*b*c)*(2*a^2*b^2 + 2*b^2*c^2 + 2*c^2*a^2 - a^4 - b^4 - c^4)^(-1/2);
103 Ky = 1/Ry;
104
105
106 % #####
107 % Function output
108 % #####
109
110 % Gaussian curvature
111 Y = Kx*Ky;
112
113 end

```

```

1  # #####
2  # Description
3  # #####
4
5  # This python script generates a parametric FE model of a beech wood self-shaping
  gridshell in Abaqus 6.14.
6
7  # Authors:
8  # Prijanthy Panchadcharam, Mai 2018
9  # Falk Wittel, Mai 2018
10 # Philippe Grönquist (pgroenquist@ethz.ch), Mai 2018
11
12 # Abaqus default initialization
13 from part import *
14 from material import *
15 from section import *
16 from assembly import *
17 from step import *
18 from interaction import *
19 from load import *
20 from mesh import *
21 from optimization import *
22 from job import *
23 from sketch import *
24 from visualization import *
25 from connectorBehavior import *
26 from abaqus import *
27 from abaqusConstants import *
28
29 # Import for accessing the results (Abaqus specific)
30 import odbAccess
31
32 # Import for python
33 import sys
34 import os
35 import numpy as np
36 import math
37 from textRepr import *
38
39 # Name model
40 mdb.models.changeKey(fromName='Model-1', toName='Grid shell')
41 myModel=mdb.models['Grid shell']
42
43 #####
44 # Import input parameters from Matlab
45 #####
46
47 # Open file
48 path='C:\Temp'
49 name='GridshellInput.txt'
50 fileNameComplete = os.path.join(path, name)
51 file = open(fileNameComplete, 'r')
52
53 # Store information of all lines
54 lineList = file.read().split('\n')
55
56 # Close the file
57 file.close()
58
59 # Find headers
60 tableTitleLineIndex = []
61 for i,line in enumerate(lineList):
62     if re.match('^Continuous', line) != None:
63         tableTitleLineIndex.append(['Continuous',i])
64     if re.match('^End', line) != None:
65         tableTitleLineIndex.append(['End',i])
66
67 # Initialize the result variables
68 dataCont=[]
69
70 # Get Floats
71 for line in lineList[tableTitleLineIndex[1][1]+1:tableTitleLineIndex[2][1]-1]:

```

```

72     dataCont.append(float(line.split(',')[0]))
73
74
75 #####
76 # Parameters setup
77 #####
78 n      = int(dataCont[0]/2+1)          # n-1 = number of screws on a strip
79 length = dataCont[1]                  # Strip length in mm
80 width  = dataCont[2]                  # Strip width in mm
81 passivethickness = dataCont[3]        # Thickness of passive layer in mm
82 activethickness  = dataCont[4]        # Thickness of active layer in mm
83 Thickness = passivethickness + activethickness
84 screwdistance = length/n              # Distance between 2 screws
85 screwdiameter = 2.0                  # Diameter of screws
86 tempinf = 20                         # Initial temperature
87 tempsup = 12                         # End temperature
88 MeshSize = width/2                   # Mesh size in mm
89 Steptime = 1                         # Step time unitless
90
91
92 #####
93 # Parts
94 #####
95
96 # Generate strips with holes (for screw connections)
97
98 # Active layer
99 myModel.ConstrainedSketch(name='__profile__', sheetSize=200.0)
100 myModel.sketches['__profile__'].rectangle(point1=(0.0, 0.0), point2=(length, width))
101
102 for i in range(1,n):
103     myModel.sketches['__profile__'].CircleByCenterPerimeter(center=(i*screwdistance,
104         width/2), point1=(i*screwdistance, width/2+screwdiameter/2))
105
106 myModel.Part(dimensionality=THREE_D, name='AL', type= DEFORMABLE_BODY)
107 myModel.parts['AL'].BaseSolidExtrude(depth=activethickness, sketch=
108 myModel.sketches['__profile__'])
109
110 # Passive layer
111 myModel.ConstrainedSketch(name='__profile__', sheetSize=200.0)
112 myModel.sketches['__profile__'].rectangle(point1=(0.0, 0.0), point2=(length, width))
113
114 for j in range(1,n):
115     myModel.sketches['__profile__'].CircleByCenterPerimeter(center=(j*screwdistance,
116         width/2), point1=(j*screwdistance, width/2+screwdiameter/2))
117
118 myModel.Part(dimensionality=THREE_D, name='PL', type= DEFORMABLE_BODY)
119 myModel.parts['PL'].BaseSolidExtrude(depth=passivethickness, sketch=
120 myModel.sketches['__profile__'])
121
122 #####
123 # Create sets & naming
124 #####
125
126 # Name surfaces to be tied later
127 topFace = myModel.parts['AL'].faces.findAt(((1,width/2,0),) )
128 botFace = myModel.parts['PL'].faces.findAt(((1,width/2,passivethickness),) )
129 topRegion = (topFace,)
130 botRegion = (botFace,)
131 myModel.parts['AL'].Surface(name='Surf-AL', side1Faces=topRegion)
132 myModel.parts['PL'].Surface(name='Surf-PL', side1Faces=botRegion)
133
134 #####
135 # Modification: For surfaces instead of edges: Used for RP coupling to layers
136 for i in range(1,n):
137     topedgeS =
138     myModel.parts['AL'].faces.findAt(((i*screwdistance,width/2-screwdiameter/2,activet
139         hickness-0.1),) )
140     botedgeS =
141     myModel.parts['PL'].faces.findAt(((i*screwdistance,width/2-screwdiameter/2,0.1),)
142     )
143     topEdgesS = (topedgeS,)

```

```

136     botEdgesS = (botedgesS,)
137     myModel.parts['AL'].Surface(name='LA ' + str(i), sidelFaces=topEdgesS)
138     myModel.parts['PL'].Surface(name='LP ' + str(i), sidelFaces=botEdgesS)
139
140     # Find and name parts, necessary for the supports
141     topedge1 = myModel.parts['PL'].vertices.findAt(((0,width,0),) )
142     botedge1 = myModel.parts['PL'].vertices.findAt(((0,0,0),) )
143     topedge2 = myModel.parts['PL'].vertices.findAt(((length,width,0),) )
144     botedge2 = myModel.parts['PL'].vertices.findAt(((length,0,0),) )
145     topEdges1 = (topedge1,)
146     botEdges1 = (botedge1,)
147     topEdges2 = (topedge2,)
148     botEdges2 = (botedge2,)
149     myModel.parts['PL'].Set(name='AUFLAGER 1', vertices=topEdges1)
150     myModel.parts['PL'].Set(name='AUFLAGER 2', vertices=botEdges1)
151     myModel.parts['PL'].Set(name='AUFLAGER 3', vertices=topEdges2)
152     myModel.parts['PL'].Set(name='AUFLAGER 4', vertices=botEdges2)
153
154     # Name surfaces, for surface contacts between vertical and horizontal strips
155     topSurface = myModel.parts['AL'].faces.findAt(((1,width/2,activethickness),) )
156     botSurface = myModel.parts['PL'].faces.findAt(((1,width/2,0),) )
157     topRegion = (topSurface,)
158     botRegion = (botSurface,)
159     myModel.parts['AL'].Surface(name='Surf-contact-AL', sidelFaces=topRegion)
160     myModel.parts['PL'].Surface(name='Surf-contact-PL', sidelFaces=botRegion)
161
162
163     #####
164     # Material
165     #####
166
167     # Define swelling coefficients
168     myModel.Material(name='Material-1')
169     myModel.materials['Material-1'].Expansion(table=((0.002, 0.004, 0.0001), ),
170     type=ORTHOTROPIC)
171
172     # Define moisture-dependent engineering constants between 0 and 30 % moisture content
173     myModel.materials['Material-1'].Elastic(table=((2505.9, 862.0,
174     16854.3, 0.849703595, 0.055647712, 0.016761444, 652.61, 1466.74, 1082.28,
175     1.0), (2446.2, 838.6, 16571.9, 0.849653412, 0.053960203, 0.016125284,
176     637.42, 1451.48, 1064.56, 2.0), (2386.5, 815.2, 16289.5, 0.849748572,
177     0.052278114, 0.015493129, 622.23, 1436.22, 1046.84, 3.0), (2326.8, 791.8,
178     16007.1, 0.850001962, 0.050601733, 0.014865189, 607.04, 1420.96, 1029.12,
179     4.0), (2267.1, 768.4, 15724.7, 0.850428037, 0.048931367, 0.014241692,
180     591.85, 1405.7, 1011.4, 5.0), (2207.4, 745.0, 15442.3, 0.85104307,
181     0.047267346, 0.013622882, 576.66, 1390.44, 993.68, 6.0), (2147.7, 721.6,
182     15159.9, 0.851865442, 0.045610025, 0.01300902, 561.47, 1375.18, 975.96,
183     7.0), (2088.0, 698.2, 14877.5, 0.852916001, 0.043959786, 0.012400389,
184     546.28, 1359.92, 958.24, 8.0), (2028.3, 674.8, 14595.1, 0.854218485,
185     0.042317038, 0.011797291, 531.09, 1344.66, 940.52, 9.0), (1968.6, 651.4,
186     14312.7, 0.855800043, 0.040682227, 0.011200055, 515.9, 1329.4, 922.8,
187     10.0), (1908.9, 628.0, 14030.3, 0.857691871, 0.03905583, 0.010609034,
188     500.71, 1314.14, 905.08, 11.0), (1849.2, 604.6, 13747.9, 0.859929995,
189     0.037438367, 0.010024612, 485.52, 1298.88, 887.36, 12.0), (1789.5, 581.2,
190     13465.5, 0.862556242, 0.0358304, 0.009447203, 470.33, 1283.62, 869.64,
191     13.0), (1729.8, 557.8, 13183.1, 0.865619458, 0.034232539, 0.008877259,
192     455.14, 1268.36, 851.92, 14.0), (1670.1, 534.4, 12900.7, 0.869177043,
193     0.032645447, 0.008315268, 439.95, 1253.1, 834.2, 15.0), (1610.4, 511.0,
194     12618.3, 0.873296914, 0.031069848, 0.007761767, 424.76, 1237.84, 816.48,
195     16.0), (1550.7, 487.6, 12335.9, 0.878060023, 0.02950653, 0.007217337,
196     409.57, 1222.58, 798.76, 17.0), (1491.0, 464.2, 12053.5, 0.883563645,
197     0.027956358, 0.006682616, 394.38, 1207.32, 781.04, 18.0), (1431.3, 440.8,
198     11771.1, 0.889925711, 0.026420277, 0.006158303, 379.19, 1192.06, 763.32,
199     19.0), (1371.6, 417.4, 11488.7, 0.897290599, 0.024899327, 0.005645165,
200     364.0, 1176.8, 745.6, 20.0), (1311.9, 394.0, 11206.3, 0.905836983,
201     0.023394651, 0.005144048, 348.81, 1161.54, 727.88, 21.0), (1252.2, 370.6,
202     10923.9, 0.915788665, 0.021907511, 0.004655883, 333.62, 1146.28, 710.16,
203     22.0), (1192.5, 347.2, 10641.5, 0.927429781, 0.020439303, 0.004181702,
204     318.43, 1131.02, 692.44, 23.0), (1132.8, 323.8, 10359.1, 0.941126602,
205     0.018991577, 0.003722648, 303.24, 1115.76, 674.72, 24.0), (1073.1, 300.4,
206     10076.7, 0.957359521, 0.017566053, 0.003279993, 288.05, 1100.5, 657.0,
207     25.0), (1013.4, 277.0, 9794.3, 0.976771261, 0.016164652, 0.002855156,

```

```

207     272.86, 1085.24, 639.28, 26.0), (953.7, 253.6, 9511.9, 1.000241763,
208     0.014789524, 0.002449724, 257.67, 1069.98, 621.56, 27.0), (894.0, 230.2,
209     9229.5, 1.029008758, 0.013443079, 0.002065477, 242.48, 1054.72, 603.84,
210     28.0), (834.3, 206.8, 8947.1, 1.064870182, 0.012128033, 0.001704423,
211     227.29, 1039.46, 586.12, 29.0), (774.6, 183.4, 8664.7, 1.11054157,
212     0.010847457, 0.001368827, 212.1, 1024.2, 568.4, 30.0)),
213     temperatureDependency=ON, type=ENGINEERING_CONSTANTS)
214
215 myModel.HomogeneousSolidSection(material='Material-1', name= 'Section-1',
    thickness=None)
216
217 # Assign material to strips
218 myModel.parts['AL'].SectionAssignment(offset=0.0, offsetField= '',
    offsetType=MIDDLE_SURFACE, region= (myModel.parts['AL'].cells,),
    sectionName='Section-1' , thicknessAssignment=FROM_SECTION)
219 myModel.parts['PL'].SectionAssignment(offset=0.0, offsetField= '',
    offsetType=MIDDLE_SURFACE, region= (myModel.parts['PL'].cells,),
    sectionName='Section-1' , thicknessAssignment=FROM_SECTION)
220
221
222 #####
223 # Define local coordinate system
224 #####
225
226 # Radial = x-direction; Tangential = y-direction; Longitudinal = z-direction;
227
228 # Active layer
229 myModel.parts['AL'].DatumCsysByThreePoints(coordSysType= CARTESIAN,
    name='AL_coordinate', origin=(0,width,activethickness), point1=
230     (length,width,activethickness), point2=(0,width,0))
231 a=myModel.parts['AL'].datums.keys()
232 myModel.parts['AL'].MaterialOrientation( additionalRotationField='',
    additionalRotationType=ROTATION_NONE, angle=0.0 , axis=AXIS_3, fieldName='',
    localCsys= myModel.parts['AL'].datums[a[-1]], orientationType=SYSTEM, region=
    (myModel.parts['AL'].cells,), stackDirection=STACK_3)
233
234 # Passive layer
235 myModel.parts['PL'].DatumCsysByThreePoints(coordSysType= CARTESIAN,
    name='PL_coordinate', origin= (0,width,passivethickness), point1=
    (0,0,passivethickness), point2= (0,width,0))
236 a=myModel.parts['PL'].datums.keys()
237 myModel.parts['PL'].MaterialOrientation( additionalRotationField='',
    additionalRotationType=ROTATION_NONE, angle=0.0 , axis=AXIS_3, fieldName='',
    localCsys= myModel.parts['PL'].datums[a[-1]], orientationType=SYSTEM, region=
    (myModel.parts['PL'].cells,), stackDirection=STACK_3)
238
239
240 #####
241 # Bilayer assembly
242 #####
243
244 # Displace active and passive layer strips to match
245 myModel.rootAssembly.DatumCsysByDefault(CARTESIAN)
246 myAssembly=myModel.rootAssembly
247 myAssembly.Instance(dependent=ON, name='AL-1', part= myModel.parts['AL'])
248 myAssembly.Instance(dependent=ON, name='PL-1', part= myModel.parts['PL'])
249 myAssembly.translate(instanceList=('PL-1', ), vector=(0.0, 0.0, -passivethickness))
250
251
252 #####
253 # Step
254 #####
255
256 myModel.StaticStep(initialInc=0.01, maxNumInc=1000, minInc= 1e-09, name='Step-1',
    previous='Initial', timePeriod=StepTime)
257 myModel.steps['Step-1'].setValues(nlgeom=ON)
258 myModel.fieldOutputRequests['F-Output-1'].setValues(variables=
259     ('S', 'NE', 'U', 'TEMP'))
260
261
262 #####

```

```

263 # BC (Temperature) (1)
264 #####

265
266 # Set the amplitude of temperature
267 myModel.TabularAmplitude(data=((0.0, tempinf), (1.0, tempsup)), name= 'Amp-1',
smooth=SOLVER_DEFAULT, timeSpan=STEP)

268
269 # Assign temeperature to strips
270 myAssembly.Set(cells=myAssembly.instances['AL-1'].cells +
myAssembly.instances['PL-1'].cells , name= 'Set-1' )
271 myModel.Temperature(createStepName='Initial',
crossSectionDistribution=CONSTANT_THROUGH_THICKNESS, distributionType= UNIFORM,
magnitudes=(tempinf, ), name='Predefined Field-1', region= myAssembly.sets['Set-1'])
272 myModel.predefinedFields['Predefined Field-1'].setValuesInStep( amplitude='Amp-1',
magnitudes=(1.0, ), stepName='Step-1')

273
274
275 #####
276 # Gridshell assembly
277 #####

278
279 # Multiply the strip and displace each by distance of screws
280 myAssembly.LinearInstancePattern(direction1=(1.0, 0.0, 0.0), direction2=(0.0, 1.0,
0.0), instanceList=('AL-1', 'PL-1'), number1=2, number2=n-1, spacing1=length,
spacing2=screwdistance)

281
282 # Rotate half of the displaced strips by 90 degrees
283 myList=()
284 for i in range(1,n):
285     myList=myList+('AL-1-lin-2-'+str(i),'PL-1-lin-2-'+str(i),)
286 myAssembly.rotate(angle=90.0, axisDirection=(0.0, 0.0,
-activethickness-passivethickness), axisPoint=(length, (n-2)*screwdistance+width,
activethickness), instanceList=myList)

287
288 # Displace the rotated strips such that holes match
289 myAssembly.translate(instanceList=myList, vector=(-screwdistance+width/2,
screwdistance - width/2, -activethickness-passivethickness))

290
291 # Tie the active and passive layers
292 # Two cases: For not using same nodes as slaves for tie and for later hinge
293 for key in myAssembly.instances.keys():
294     if key.find("AL")>=0:
295         if key.find("AL-1-lin-2")>=0:
296             surf1=myAssembly.instances[key].surfaces['Surf-AL']
297             key1="P"+key[1:]
298             surf2=myAssembly.instances[key1].surfaces['Surf-PL']
299             myModel.Tie(adjust=ON, slave= surf1, name= 'Constraint-'+key,
positionToleranceMethod=COMPUTED, master= surf2, thickness=ON,
tieRotations=ON)
300         else:
301             surf1=myAssembly.instances[key].surfaces['Surf-AL']
302             key1="P"+key[1:]
303             surf2=myAssembly.instances[key1].surfaces['Surf-PL']
304             myModel.Tie(adjust=ON, master= surf1, name= 'Constraint-'+key,
positionToleranceMethod=COMPUTED, slave= surf2, thickness=ON,
tieRotations=ON)

305
306
307 #####
308 # Connection of strips via hinged wires
309 #####

310
311 # Define reference points necessary for wire connections
312 RPlist=[]
313 for i in range(1,n):
314     for j in range(0,n-1):
315         point1,point2=(i*screwdistance,width/2+j*screwdistance,activethickness),
(i*screwdistance,width/2+j*screwdistance,-2*passivethickness-activethickness)
316         myAssembly.ReferencePoint(point=point1)
317         myAssembly.ReferencePoint(point=point2)
318         myKeys=myAssembly.referencePoints.keys()

```

```

319         RPlist=RPlist+[myKeys[0:2]] # append an element in RPlist composed of the
           two first elements in myKeys
320
321     # Reference number, retrieve internal numbering, minimum is set to bottom
322     x = min(RPlist[0])
323
324     # Connect the reference points with edges of holes
325     for i in range(1,((n-1)**2)*2+1):
326
327         myAssembly.Set(name='Set-RP-'+str(i),referencePoints=(myAssembly.referencePoints[x
           +i-1],))
328
329     # Connect reference points with even numbers
330     dc = n - 1
331     j = 1
332     for i in range(1,(n-1)**2+1):
333         myModel.Coupling(controlPoint= myAssembly.sets['Set-RP-'+str(i*2)],
           couplingType= KINEMATIC, influenceRadius=WHOLE_SURFACE, localCsys=None, name=
           'Coupling-gerade RP '+str(i), surface=
334         myAssembly.instances['PL-1-lin-2-'+str(j)].surfaces['LP '+str(dc)], u1=ON,
           u2=ON, u3=ON, url=ON, ur2=ON, ur3=ON)
335         if dc > 1:
336             dc = dc -1
337         else:
338             dc = dc + (n-2)
339             j = j + 1
340
341     # Connect reference points with odd numbers
342     for i in range(1,n):
343         myModel.Coupling(controlPoint= myAssembly.sets['Set-RP-'+str(1 +
           (n-1)*2*(i-1))], couplingType= DISTRIBUTING, influenceRadius=WHOLE_SURFACE,
           localCsys=None, name=
344         'Coupling-ungereade RP 1'+str(i), surface=
           myAssembly.instances['AL-1'].surfaces['LA '+str(i)], u1=ON, u2=ON, u3=ON,
           url=ON, ur2=ON, ur3=ON)
345
346     dc = 1
347     for j in range(3,(n-1)*2,2):
348         dc = dc + 1
349         for i in range(1,n):
350             myModel.Coupling(controlPoint= myAssembly.sets['Set-RP-'+str(j +
           (n-1)*2*(i-1))], couplingType= DISTRIBUTING, influenceRadius=WHOLE_SURFACE,
           localCsys=None, name=
351             'Coupling-ungereade RP '+str(j)+ str(i), surface=
           myAssembly.instances['AL-1-lin-1-'+str(dc)].surfaces['LA '+str(i)], u1=ON,
           u2=ON, u3=ON, url=ON, ur2=ON, ur3=ON)
352
353     # Create wire connection
354     for i in range(0,(n-1)**2*2,2):
355         myAssembly.WirePolyLine(mergeType=IMPRINT, meshable= OFF,
           points=((myAssembly.referencePoints[x+i], myAssembly.referencePoints[x+1+i]),))
356         myAssembly.Set(edges= myAssembly.edges, name='Wire-'+str(i))
357
358     # Define the screw connection as a hinge
359     myModel.ConnectorSection(assembledType=HINGE, name='hinge')
360
361     # Assign hinge to wire connection and define a local hinge coordinate system
362     myAssembly.DatumCsysByThreePoints(coordSysType= CARTESIAN, name='hinge coordinate',
           origin= (screwdistance,width/2,activethickness), point1= (screwdistance,width/2,0),
           point2= (screwdistance,width/2-screwdiameter/2,activethickness))
363
364     # Last wire definition contains all previous wires
365     myAssembly.SectionAssignment(region= myAssembly.sets['Wire-'+str((n-1)**2*2-2)],
           sectionName='hinge')
366     a=myAssembly.datums.keys()
367     myAssembly.ConnectorOrientation( localCsys1= myAssembly.datums[a[-1]], region=
           myAssembly.allSets['Wire-'+str((n-1)**2*2-2)])
368
369     #####
370     # Contact of surfaces
371     #####

```

```

372
373 # Choose surfaces for contact
374 Asurfacecontact=[]
375 Akey=[]
376 for i in range(1,n):
377     Asurfacecontact = Asurfacecontact+['Surf-contact-AL',]
378     Akey = Akey + ['AL-1-lin-2-'+str(i),]
379
380 Psurfacecontact=[]
381 Pkey=[]
382 for i in range(2,n):
383     Psurfacecontact = Psurfacecontact + ['Surf-contact-PL',]
384     Pkey = Pkey + ['PL-1-lin-1-'+str(i),]
385
386 Pkey = Pkey + ['PL-1',]
387 Psurfacecontact = Psurfacecontact + ['Surf-contact-PL',]
388
389 # Define interaction of surfaces
390 myModel.ContactProperty('IntProp-1')
391 myModel.interactionProperties['IntProp-1'].NormalBehavior(allowSeparation=ON,
392 constraintEnforcementMethod=DEFAULT, pressureOverclosure=HARD)
393
394 for i in range(0,n-1):
395     for j in range(0,n-1):
396         myModel.SurfaceToSurfaceContactStd(adjustMethod=NONE, clearanceRegion=None,
397         createStepName='Step-1', datumAxis=None, initialClearance=OMIT,
398         interactionProperty='IntProp-1',
399         master=myAssembly.instances[Pkey[j]].surfaces[Psurfacecontact[j]],
400         name='surface contact'+str(i)+str(j),
401         slave=myAssembly.instances[Akey[i]].surfaces[Asurfacecontact[i]],
402         sliding=FINITE, thickness=ON)
403
404 #####
405 # BC (Temperature) (2)
406 #####
407
408 # Assign temperature Boundary Condition (BC)
409 for i in range(2,n):
410     myAssembly.Set(cells=myAssembly.instances['AL-1-lin-1-'+str(i)].cells +
411     myAssembly.instances['PL-1-lin-1-'+str(i)].cells , name = 'Set-'+str(i) )
412
413 for i in range(1,n):
414     myAssembly.Set(cells=myAssembly.instances['AL-1-lin-2-'+str(i)].cells +
415     myAssembly.instances['PL-1-lin-2-'+str(i)].cells , name = 'Set-'+str((n-1)+i) )
416
417 for i in range(2,(n-1)*2+1):
418     myModel.Temperature(createStepName='Initial',
419     crossSectionDistribution=CONSTANT_THROUGH_THICKNESS, distributionType= UNIFORM,
420     magnitudes=(tempinf, ), name='Predefined Field-'+ str(i), region=
421     myAssembly.sets['Set-'+str(i)])
422     myModel.predefinedFields['Predefined Field-' + str(i)].setValuesInStep(
423     amplitude='Amp-1', magnitudes=(1.0, ), stepName='Step-1')
424
425 #####
426 # Restraints and supports
427 #####
428
429 # Define 4 points for supports of the gridshell
430 p1 =
431 (screwdistance-width/2,-screwdistance+width/2,-2*passivethickness-activethickness)
432 p2 =
433 ((n-1)*screwdistance+width/2,-screwdistance+width/2,-2*passivethickness-activethicknes
434 s)
435 p3 =
436 (screwdistance-width/2,(n-1)*screwdistance+width/2,-2*passivethickness-activethickness
437 )
438 p4 =
439 ((n-1)*screwdistance+width/2,(n-1)*screwdistance+width/2,-2*passivethickness-activethi

```



```

ckness)
425 myAssembly.ReferencePoint(point=p1)
426 myAssembly.ReferencePoint(point=p2)
427 myAssembly.ReferencePoint(point=p3)
428 myAssembly.ReferencePoint(point=p4)
429 myKeys=myAssembly.referencePoints.keys()
430
431 y = max(myKeys)
432 for i in range(1,5):
433
434     myAssembly.Set(name='Set-AUFLAGER-'+str(i),referencePoints=(myAssembly.referencePo
435     ints[y-i+1],))
436
437 # Connect the support points to the strips
438 for i in range(2,5,2):
439     myModel.Coupling(controlPoint= myAssembly.sets['Set-AUFLAGER-'+str(i)],
440     couplingType= KINEMATIC, influenceRadius=WHOLE_SURFACE, localCsys=None, name=
441     'Coupling-Auflager RP '+str(i), surface= myAssembly.sets['PL-1-lin-2-1.AUFLAGER
442     '+str(i)], u1=ON, u2=ON, u3=ON, url=ON, ur2=ON, ur3=ON)
443
444 for i in range(1,4,2):
445     myModel.Coupling(controlPoint= myAssembly.sets['Set-AUFLAGER-'+str(i)],
446     couplingType= KINEMATIC, influenceRadius=WHOLE_SURFACE, localCsys=None, name=
447     'Coupling-Auflager RP '+str(i), surface=
448     myAssembly.sets['PL-1-lin-2-'+str(n-1)+'AUFLAGER '+str(i)], u1=ON, u2=ON,
449     u3=ON, url=ON, ur2=ON, ur3=ON)
450
451 # 3 sliding supports (sliding in global x- and y-direction)
452 for i in range(1,4):
453     myModel.DisplacementBC(amplitude=UNSET, createStepName= 'Step-1',
454     distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=None,
455     name='verschiebliches Auflager ' + str(i),
456     region= myAssembly.sets['Set-AUFLAGER-'+str(i)], u1=UNSET, u2=UNSET, u3=0.0,
457     url=UNSET, ur2=UNSET, ur3=UNSET)
458
459 # 1 fixed support
460 myModel.DisplacementBC(amplitude=UNSET, createStepName= 'Step-1',
461 distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=None ,
462 name='unverschiebliches Auflager 1',
463 region= myAssembly.sets['Set-AUFLAGER-4'], u1=0.0, u2=0.0, u3= 0.0, url=UNSET,
464 ur2=UNSET, ur3=UNSET)
465
466 #####
467 # Mesh
468 #####
469
470 # Active layer
471 myModel.parts['AL'].seedPart(deviationFactor=0.1, minSizeFactor=0.1, size=MeshSize)
472 myModel.parts['AL'].generateMesh()
473 myModel.parts['AL'].setElementType(elemTypes=(ElemType( elemCode=C3D20R,
474 elemLibrary=STANDARD), ElemType(elemCode=C3D15, elemLibrary=STANDARD),
475 ElemType(elemCode=C3D10, elemLibrary=STANDARD)), regions=(myModel.parts['AL'].cells,
476 ))
477
478 # Passive layer
479 myModel.parts['PL'].seedPart(deviationFactor=0.1, minSizeFactor=0.1, size=MeshSize)
480 myModel.parts['PL'].generateMesh()
481 myModel.parts['PL'].setElementType(elemTypes=(ElemType( elemCode=C3D20R,
482 elemLibrary=STANDARD), ElemType(elemCode=C3D15, elemLibrary=STANDARD),
483 ElemType(elemCode=C3D10, elemLibrary=STANDARD)), regions=(myModel.parts['PL'].cells,
484 ))
485
486 #####
487 # Job
488 #####
489
490 ODBname = 'Gridshell'
491 mdb.Job(model='Grid shell', name=ODBname)
492 mdb.jobs[ODBname].setValues(numCpus=12, numDomains=12, numGPUs=1)
493 mdb.jobs[ODBname].submit(consistencyChecking=ON)

```

```

477 mdb.jobs[ODBname].waitForCompletion()
478
479
480 #####
481 # Process ODB
482 #####
483
484 # Access results file (odb) and retrieve z-displacement at evaluation point
485 odbPath = ODBname + '.odb'
486
487 # open the odb
488 myOdb = odbAccess.openOdb(path=odbPath)
489
490 # choose the time frame (negative indexing -> gets last element)
491 frame = myOdb.steps['Step-1'].frames[-1]
492
493 # container for results
494 results = []
495
496 # choose location
497 for i in range(1,2*(n-1)**2+1): # if n=4, loops from 1 to 18
498
499     # evaluation point
500     PointName = 'SET-RP-' + str(i)
501     EndPoint =
502     myOdb.rootAssembly.nodeSets[PointName]
503
504     # choose what field output
505     dispField = frame.fieldOutputs['U']
506     dispSubField = dispField.getSubset(region=EndPoint)
507
508     # retrieve displacements
509     dataX =
510     dispSubField.values[0].data[0]
511     dataY = dispSubField.values[0].data[1]
512     dataZ = dispSubField.values[0].data[2]
513
514     results.append([i,dataX,dataY,dataZ])
515
516 # Close current odb file
517 myOdb.close()
518
519 #####
520 # Write results to a .txt file
521 #####
522 f = open('GridshellOutput.txt','w')
523 f.write('RPi Xcoord Ycoord Zcoord\n')
524 for s in range(0,len(results)):
525     f.write('%d %d %d %e\n' %
526             (results[s][0],results[s][1],results[s][2],results[s][3]))
527 f.close()

```