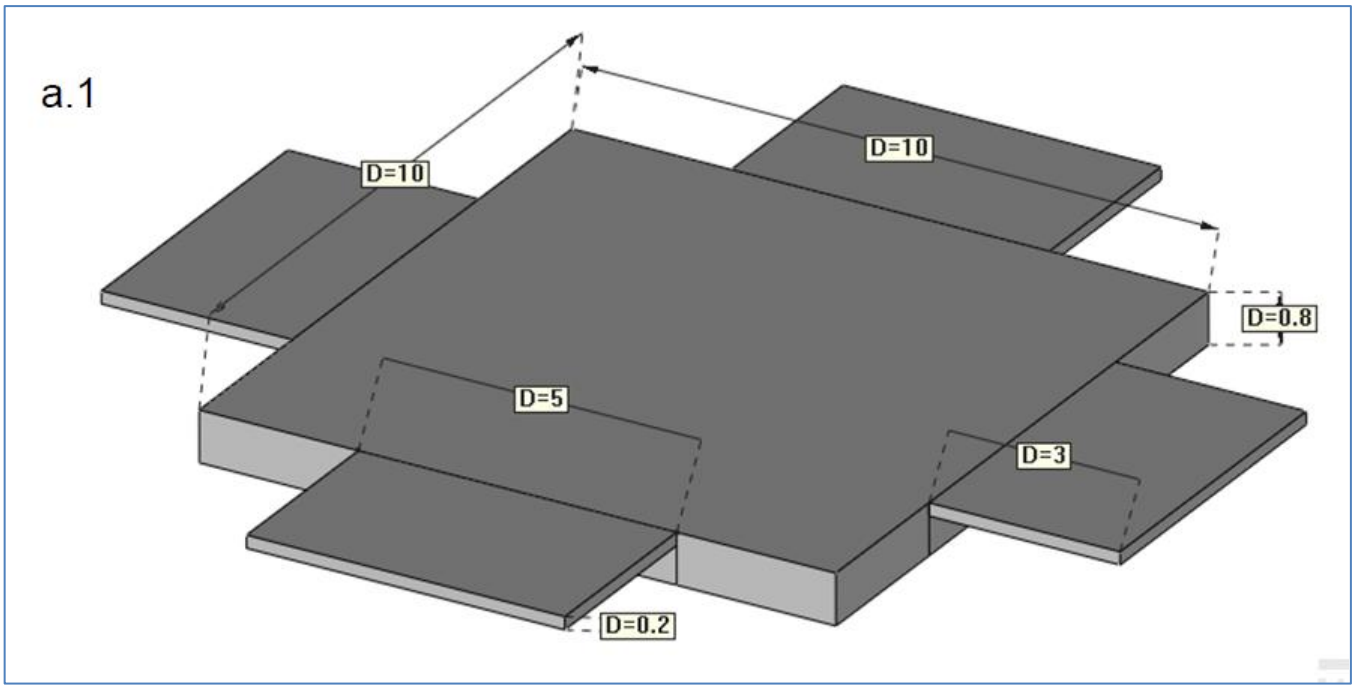
Computational Structural Mechanics & Dynamics

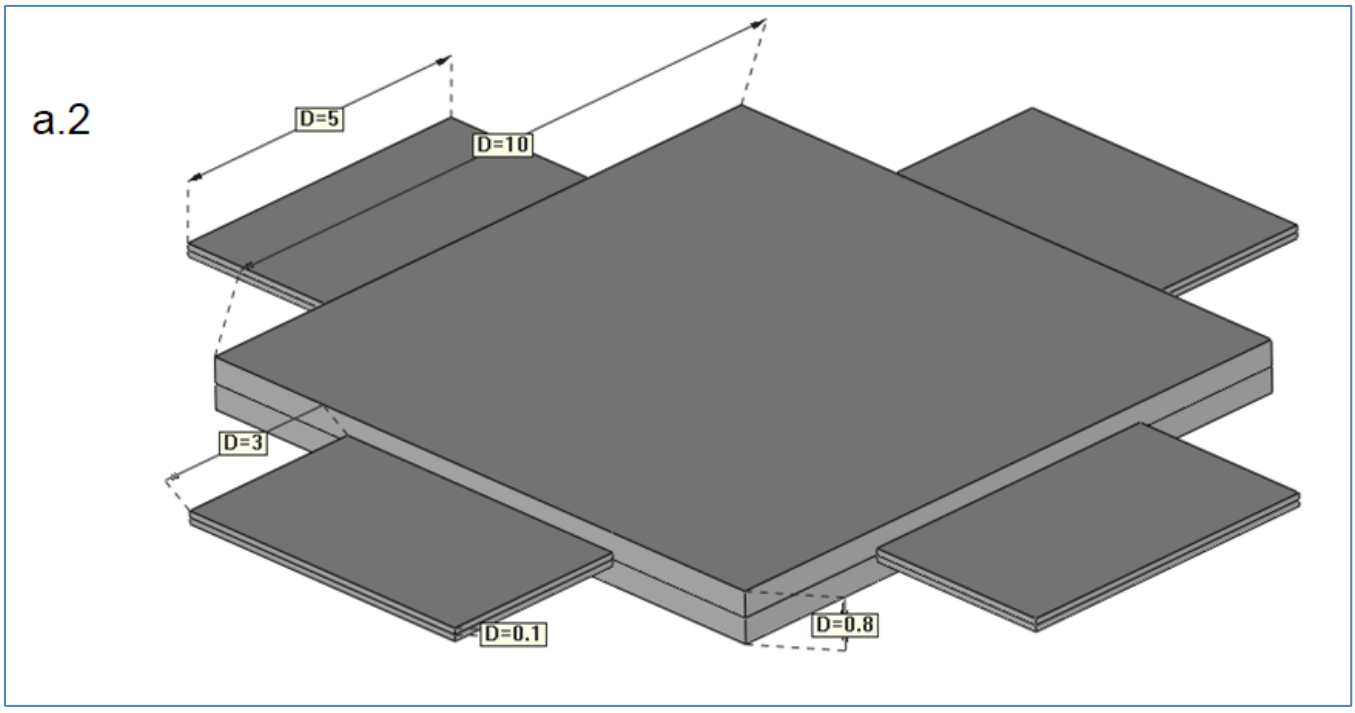
Plate Assignment

Mohammad Mohsen Zadehkamand

27 March 2017

❖ **Task A: What kind of strategy (theory, elements, integration rule, boundary conditions, etc.) will you use for solving the following problems:**





In this problem due to the slim ratios of thickness/width one should firstly check if it is possible to treat the problem as the plate case or not? Based on the theory we have:

- Thickness/width (1) = 0.8/10 = 0.080 < 0.1
- Thickness/width (1) = 0.2/3 = 0.0667 < 0.1

So both plates are included in thin plate theories (ratio<=0.1)

As we know for thin plates both "classical Kirchhof plate theory" and "Reissner-Mindlin plate theory" are applicable. **However** for the **[a1] case** due to the **unsymmetrical orientation of small plates regarding the mid-plane surface**, there would be problems for using these methods which would be discussed further in the following.

Classical Kichhoff plate theory is equivalent to Euler Bernoulli beam theory and **Reissner-Mindlin plate theory** is equivalent to Timoshenko beam theory. They have some common assumptions as following:

- The points belonging to the middle move only vertically and not in x and y direction.
- The points along a normal to the middle plane have the same vertical displacement
- The 3D normal stress is negligible.

But they differ in some others, namely:

- **Classical Kirchhof plate theory** deals only with thin plates and assumes negligible shear energy. The points laying on a normal to the plate middle plane remain on straight line also orthogonal to the middle plane after deformation **(Orthogonally condition)**

- **Reissner-Mindlin plate theory**, deals whit thick and sometimes thin plates and considers shear energy. The points laying on a normal to the plate middle plane remain on straight line but not necessarily orthogonal to the middle plane after deformation **(No orthogonally condition)**

- In **Classical Kirchhof plate theory** the integral of the **PVW** contains 2^{nd} derivatives of deflection so both the deflection and deflectin gradient must have C1 continuity. However in the **Reissner-Mindlin plate theory** only we need to maintain Co continuity.

Due to the fact that in **[a1] case** the mid-plane surfaces for large and small plates are not coinciding on each other there would be some inconsistencies if one would like to use the Classical Kichhoff plate theory for this problem.

Otherwise one would be expect the points laying on the normal to the plate middle plane at intersection line of large and small plate remain orthogonal to the middle plane after deformation (Orthogonally condition) but this would not necessarily been satisfied, just because the mid-plane of large and small plates are not unique and this theory cannot capture also the shear stress to compensate the dislocation of these two planes. So for the **[a1] case** it seems that only **Reissner-Mindlin plate theory** could be used considering the shear stress initiation in the interaction line of different sized plates. However for the **[a2] case** the **Classical Kirchhof plate theory** would completely satisfy for the modeling.

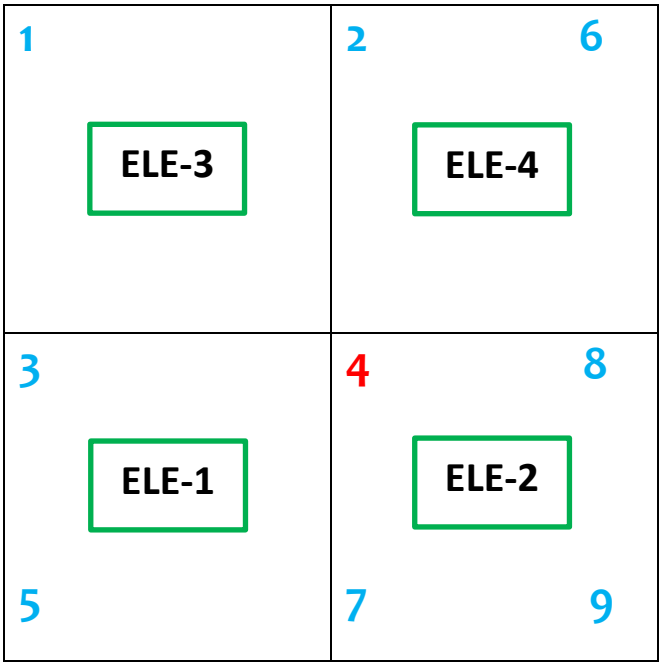
❖ Task B: Define and verify a patch test mesh for the MCZ element.

The patch test is based on imposing a displacement field on a patch of elements at the boundary which can be exactly reproduced by the shape functions. The patch test is satisfied if the displacement and strains within the patch coincide with the exact values deduced from the prescribed displacement field.

The test is based on selecting an arbitrary patch of elements and imposing upon it nodal displacements corresponding to any state of constant strain. If nodal equilibrium is achieved without a need for imposing external force and the pre-calculated constant strain field is achieved in the all domain then the patch test is successful. [1]

The patch test also includes the satisfaction of rigid body motion, in which we should apply a zero strain field on the boundaries of domain and could capture the zero strain field on all the domain. [1]

For doing the patch test, one simple Input file is used in the Matlab code. This file is provided as the final annex in this report. This is a 4 element plate each of elements containing 4 nodes. Boundary nodes are portrayed in blue and the only middle point is in red.



[1]. Introduction to the Finite Element Method, Lectures for Erasmus Mundus Master Course. Part 3.18 - Page 135

Table1. Nodal Coordinate

x	y
0.0	10.0
5.0	10.0
0.0	5.0
5.0	5.0
0.0	0.0
10.0	10.0
5.0	0.0
10.0	5.0
10.0	0.0

Table2. Element connectivity

Element	Node1	Node2	Node3	Node4
1	5	7	4	3
2	7	9	8	4
3	3	4	2	1
4	4	8	6	2

- For the first Patch test we use rigid body motion and impose the unit boundary condition movement for all boundary nodes. In this case we anticipate to have the same unite displacement in z direction at node 4.
  But the important point is that for this test the density is assumed to be equal to zero, since we are just checking the displacement boundary conditions and no loads should be applied to be able to predict the field behavior.

  Doing tis patch test, as we expected program declares for "u vector" that:

  val = All zero sparse: 27-by-1
  this "27" is related to dof*nnode = 3*9 = 27 [all degrees of freedom]

- For the second patch test we put a one-way bending in x-direction. [on the plate in 1-3-5 and 6-8-9 edges] as follows:
  Edge 6-8-9 would have a vertical displacement of 1 in z direction and the other edge would be stationary. On the other hand both edges would have complete resistance in front of X direction moment; it means that no rotation is allowed. In this case we expect to have a vertical displacement of 0.5 in the middle line 2-4-7 and a fixed equal rotation for all those 3 nodes.

  Doing tis patch test, as we expected program declares for "u vector" that:

  val =

(4, 1)	0.5000	> vertical displacement for node 2
(5, 1)	0.1500	> x-direction rotation for node 2
(6, 1)	0.0000	> y-direction rotation for node 2
(10, 1)	0.5000	> vertical displacement for node 4
(11, 1)	0.1500	> x-direction rotation for node 4
(12, 1)	0.0000	> y-direction rotation for node 4
(19, 1)	0.5000	> vertical displacement for node 7
(20, 1)	0.1500	> x-direction rotation for node 7
(21, 1)	-0.0000	> y-direction rotation for node 7

- So these two simple tests, proves that MZC element with rectangular shape satisfies the patch test properly

❖ Annex B: Patch Test files:

❖ Patch1

```
% Material Properties
young  = 10.92 ;
 poiss  = 0.3 ;
 thick  = 0.01;
 denss  = 0.000000000 ;
%
% Coordinates
global coordinates
coordinates = [
     0.0,      10.0 ;
     5.0,      10.0 ;
     0.0,       5.0 ;
     5.0,       5.0 ;
     0.0,       0.0 ;
    10.0,      10.0 ;
     5.0,       0.0 ;
    10.0,       5.0 ;
    10.0,       0.0 ];
%
% Elements
global elements
elements = [
       5,   7,   4,   3;
       7,   9,   8,   4;
       3,   4,   2,   1;
       4,   8,   6,   2];
%
global fixdesp
% Fixed Nodes
fixdesp  = [
     1, 1,  0.0;
     1, 2,  0.0;
     1, 3,  0.0;
     2, 1,  0.0;
     2, 2,  0.0;
     2, 3,  0.0;
     3, 1,  0.0;
     3, 2,  0.0;
     3, 3,  0.0;
     5, 1,  0.0;
     5, 2,  0.0;
     5, 3,  0.0;
     6, 1,  0.0;
     6, 2,  0.0;
     6, 3,  0.0;
     7, 1,  0.0;
     7, 2,  0.0;
     7, 3,  0.0;
     8, 1,  0.0;
     8, 2,  0.0;
     8, 3,  0.0;
     9, 1,  0.0;
     9, 2,  0.0;
     9, 3,  0.0];
%
% Point loads
pointload = [ ];
```

❖ Patch2%

```
% Material Properties
%
 young  = 10.92 ;
 poiss  = 0.3 ;
 thick  = 0.01;
 denss  = 0.000000000 ;
%
% Coordinates
global coordinates
coordinates = [
     0.0,      10.0 ;
     5.0,      10.0 ;
     0.0,       5.0 ;
     5.0,       5.0 ;
     0.0,       0.0 ;
    10.0,      10.0 ;
     5.0,       0.0 ;
    10.0,       5.0 ;
    10.0,       0.0 ];
%
% Elements
global elements
elements = [
       5,   7,   4,   3;
       7,   9,   8,   4;
       3,   4,   2,   1;
       4,   8,   6,   2];
%
global fixdesp
%
% Fixed Nodes
fixdesp  = [
     1, 1,  0.0;
     1, 2,  0.0;
     1, 3,  0.0;
     3, 1,  0.0;
     3, 2,  0.0;
     3, 3,  0.0;
     5, 1,  0.0;
     5, 2,  0.0;
     5, 3,  0.0;
     6, 1,  1.0;
     6, 2,  0.0;
     6, 3,  0.0;
     8, 1,  1.0;
     8, 2,  0.0;
     8, 3,  0.0;
     9, 1,  1.0;
     9, 2,  0.0;
     9, 3,  0.0];
%
% Point loads
%
pointload = [ ];
```