User-documentation for computational micro-mechanics

Van Dung NGUYEN

October 8, 2020

1 Computational micromechanics - description

In general, the nonlinear solver can solve a general multi-physics boundary value problem. We consider at each material point the following fields:

- Displacement field (first 3 degrees of freedom (DOFs));
- Nonlocal fields (next N_{nl} nonlocal DOFs); and
- Extra-fields (next N_{extra} extra-DOFs) .

Each field possesses its own governing equations and constitutive equations, which can be formulated in fully coupled forms, leading to a fully-coupled multi-physics boundary value problem.

The constitutive behavior is obtained using the homogenization procedure of a microscopic boundary value problem (micro-BVP) defined on a representative volume element (RVE) with a suitable microscopic boundary condition (micro-BC). The evolution of these fields are driven by so-called macroscopic kinematic variables, *e.g.* deformation gradient, temperature, temperature gradient, *etc.* Note that some micro-fields do not have a macroscopic driven kinematic variable, *e.g.* the nonlocal fields. To extract homogenized properties, *e.g.* macroscopic (so-called homogenized) stress, macroscopic fluxes, on can distinguish two different cases:

- The state of the micro-BVP is fully driven by the macroscopic kinematic variables, *e.g* macroscopic deformation gradient $\mathbf{F}$ is known and macroscopic stress $\mathbf{P}$ is sought. In this case, the use of a micro-BC, which can be either periodic BC, minimal kinematic BC, displacement BC, or mixed BC, is preferred. This possibility, so-called microscopic solver, is detailed in Section 2.
- Sometimes, a particular virtual test should be performed, *e.g.* uniaxial stress, plane stress state, *etc.*, in which the macroscopic kinematic variable is not fully known (*e.g.* in uniaxial test, we control only the strain following in one direction and the stress is constrained to be uniaxial). This possibility, so-called macroscopic solver, is detailed in Section 3. Moreover, by the fact that the microscopic solver is still limited to the continuous Galerkin formulation, the use of the macroscopic solver allows the possibility to use the discontinuous Galerkin formulation with different numerical developments, *e.g.* cohesive elements, crack propagation.

First, we define different physicals (corresponding the physicals in geo file in Gmsh) which should be available to define the BC, see Fig. 1:

- For 2D problems:

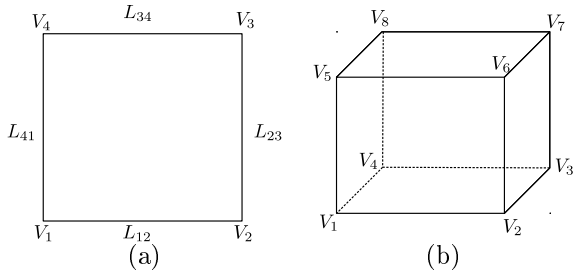

Figure 1: RVE in 2D (a) and 3D (b) problems.

  - Nodes: V_1, V_2, V_3, V_4;
  - Edges: L_{12}, L_{23}, L_{34}, L_{41};
  - Control nodes: V_1, V_2, V_4;
  - Negative edges (including the control nodes): L_{12} and L_{14};
  - Positive edges: remaining edges.

  Note: the convention about control nodes and negative edges are not unique but a similar architecture should be respect, *e.g.* the control nodes V_1, V_2, V_3 combining with the negative edges L_{12}, L_{23} also work.

- For 3D problems:
  - Nodes: V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8;
  - Edges: L_{12}, L_{23}, L_{34}, L_{41}, L_{56}, L_{67}, L_{78}, L_{85}, L_{15}, L_{26}, L_{37}, L_{48} (we use two node indexes to nominate the edges);
  - Faces: S_{1234}, S_{1562}, S_{1584}, S_{5678}, S_{4873}, S_{2673} (we use 4 node indexes to nominate the faces);
  - Control nodes: V_1, V_2, V_4, V_5;
  - Negative edges (including the control nodes): L_{12}, L_{14}, L_{15}.
  - Positive edges: remaining edges;
  - Negative faces (including the control edges): S_{1234}, S_{1562}, S_{1584};
  - Positive faces: remaining faces.

2 Microscopic solver

Will be completed

3 Macroscopic solver

3.1 Periodic BC with with “test" option

The basis idea of this possibility is to express the periodic BC is terms of multiple linear constraints between the degrees of freedom at the external boundary and control nodes

3.1.1 For 2D analyses

Note that in 2-dimensional analysis, the out-of-plan displacement (z-direction) must be eliminated by

```
# phys is the physical of the domain,
mysolver.displacementBC("Face",phys,2,0.)
# if multiple domains are available, multiple constraints must be applied
#               for all physicals
```

First, a periodic BC is defined an added into the solver:

```
# create micro BC
# tag: an integer to identify the BC
# addDofPerVertex (0 by default): number of additional fields,
#                  which does not governed by any macroscopic kinematic variables,
#                  e.g. nonlocal variables
microBC = nonLinearPeriodicBC(tag,2,addDofPerVertex)
# set homogenization order
microBC.setOrder(order) # 1 or 2
# set boundary support
microBC.setBCPhysical(L12,L41,L34,L23)
# other options for periodic BC
# method (0,1,2,3,4,5,6 are available) for constructing linear constraints
# degree (5 by default) : degree of interpolation
#          when the interpolation method is used (method = 1,2,3,4,6)
# addvertex (False by default): boolean to specify
#      if new dof is used for interpolation and
#          when the interpolation method is used (method = 1,2,3,4)
microBC.setPeriodicBCOptions(method, degree, addvertex)
# add to solver
mysolver.addMicroBC(microBC)
# activate test
mysolver.activateTest(True)
```

Then a testing mode can be obtained by prescribing displacement or force on control nodes.

```
# node V1 is always fixed
mysolver.displacementBC("Node",V1,0,0.)
mysolver.displacementBC("Node",V1,1,0.)
# other control nodes (V2, V4) should be contrained to obtain a prescibed mode
# and also to eliminate the rigid body motion
#---------------
# for example in a uniaxial test following direction V1-V2
# fixe V2 following y-direction
mysolver.displacementBC("Node",V2,1,0.)
# prescribed displacement following x-direction
mysolver.displacementBC("Node",V2,0,disp)
```

3.1.2 For 3D analyses

First, a periodic BC is defined an added into the solver:

```
# create micro BC
# tag: an integer to identify the BC
# addDofPerVertex (0 by default): number of additional fields,
#                  which does not governed by any macroscopic kinematic variables,
#                  e.g. nonlocal variables
microBC = nonLinearPeriodicBC(tag,3,addDofPerVertex)
# set homogenization order
microBC.setOrder(order) # 1 or 2
# set boundary support
microBC.setBCPhysical(S1234,S1562,S1584,S5678,S4873,S2673)
# other options for periodic BC
# method (0,1,2,3,4,5,6 are available) for constructing linear constraints
# degree (5 by default) : degree of interpolation
#          when the interpolation method is used (method = 1,2,3,4,6)
# addvertex (False by default): boolean to specify
#      if new dof is used for interpolation and
#          when the interpolation method is used (method = 1,2,3,4)
microBC.setPeriodicBCOptions(method, degree, addvertex)
# add to solver
mysolver.addMicroBC(microBC)
# activate test
mysolver.activateTest(True)
```

Then a testing mode can be obtained by prescribing displacement or force on control nodes.

```
# node V1 is always fixed
mysolver.displacementBC("Node",V1,0,0.)
mysolver.displacementBC("Node",V1,1,0.)
mysolver.displacementBC("Node",V1,2,0.)
# other control nodes (V2, V4, V5) should be contrained to obtain a prescibed mode
# and also to eliminate the rigid body motion
#---------------
# for example in a uniaxial test following direction V1-V2
mysolver.displacementBC("Node",V2,1,0.)
mysolver.displacementBC("Node",V2,2,0.)
mysolver.displacementBC("Node",V4,2,0.)
# prescribed displacement following x-direction
mysolver.displacementBC("Node",V2,0,disp)
```

3.2 Tailoring micro-BCs using different following BC implemented in the nonlinear solver

Different micro-BCs can be obtained using different following BC implemented in the nonlinear solver:

- Periodic BC between two groups:

```
# onwhat: location of BC ("Node", "Edge","Face")
# phys1, phys2: physicals of negative and positive parts
# v1, v2: physicals of the corresponding control nodes
# comp: component to be constrained
```

```
# meaning of the PBC:
# unknown[comp]_phys2 - unknown[comp]_phys1 =
#         unknown[comp]_v2 - unknown[comp]_v1
mysolver.periodicBC(onwhat,phys1,phys2,v1,v2,comp)
```

- Average periodic BC between two groups:

```
# onwhat: location of BC ("Node", "Edge","Face")
# phys1, phys2: physicals of negative and positive parts
# v1, v2: physicals of the corresponding control nodes
# comp: component to be constrained
# meaning of the average PBC:
# average<unknown[comp]>_phys2 - average<unknown[comp]>_phys1 =
#           unknown[comp]_v2 - unknown[comp]_v1
mysolver.averagePeriodicBC(onwhat,phys1,phys2,v1,v2,comp)
```

- Same unknown between two groups:

```
# onwhat: location of BC ("Node", "Edge","Face")
# phys1, phys2: physicals of negative and positive parts
# comp: component to be constrained
# meaning: unknown[comp]_phys2 - unknown[comp]_phys1 = 0
mysolver.sameDisplacementBCBetweenTwoGroups(onwhat,phys1,phys2,comp)
```

- Same unknown between a group and a node:

```
# onwhat: location of BC ("Node", "Edge","Face")
# phys: physical of the group
# rootphy: physical of a node
# comp: component to be constrained
# fact (1 by default): scaling factor
# meaning: unknown[comp]_phy - fact*unknown[comp]_rootphy = 0
mysolver.sameDisplacementBC(onwhat,phy,rootphy,comp,fact=1)
```

Note: it very important to create a well-posed system of constraints. Whn using periodicBC, sameDisplacementBCBetweenTwoGroups, intersection between different physicals can be present, leading to the duplication of constraints. To manage this situation, two functions are used:

```
# periodicBC and sameDisplacementBCBetweenTwoGroups with faces (onwhat = "Face")
# will not consider the nodes in the edge physicals added to the solver
# multiple calls if multiple edges are related
# periodicBC and sameDisplacementBCBetweenTwoGroups with these edges
#            must be created seperately
mysolver.addPeriodicEdgePhysicals(edge)
```

and

```
# periodicBC and sameDisplacementBCBetweenTwoGroups with edges (onwhat = "Edge")
# will not consider the nodes in the node physicals added to the solver
# multiple calls if multiple nodes are related
# periodicBC and sameDisplacementBCBetweenTwoGroups with these nodes
#            must be created seperately
mysolver.addPeriodicNodePhysicals(node)
```

3.2.1 Periodic BC

For 2D analysis **********:

```
# phys is the physical of the domain,
# if multiple domains are available, multiple constraints must be applied
#               for all physicals
mysolver.displacementBC("Face",phys,2,0.)
# add intersection nodes
mysolver.addPeriodicNodePhysicals(V1)
mysolver.addPeriodicNodePhysicals(V2)
mysolver.addPeriodicNodePhysicals(V3)
mysolver.addPeriodicNodePhysicals(V4)
# PBC for all edges
mysolver.periodicBC("Edge",L12,L34,V1,V4,0)
mysolver.periodicBC("Edge",L12,L34,V1,V4,1)
mysolver.periodicBC("Edge",L41,L23,V1,V2,0)
mysolver.periodicBC("Edge",L41,L23,V1,V2,1)
# PBC for nodes
mysolver.periodicBC("Node",V4,V3,V1,V2,0)
mysolver.periodicBC("Node",V4,V3,V1,V2,1)
# for uniaxial test in x -direction
# node V1 is always fixed
mysolver.displacementBC("Node",V1,0,0.)
mysolver.displacementBC("Node",V1,1,0.)
mysolver.displacementBC("Node",V1,2,0.)
# fixe V2 following y-direction
mysolver.displacementBC("Node",V2,1,0.)
# prescribed displacement following x-direction
mysolver.displacementBC("Node",V2,0,disp)
#
# if nonlocal variable is present, component 3 for example
# multiple constraint must be considered with multiple nonlocal variables
mysolver.sameDisplacementBCBetweenTwoGroups("Edge",L12,L34,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Edge",L41,L23,3)
```

For 3D analysis **********:

```
# pbc for faces
mysolver.periodicBC("Face",S1234,S5678,V1,V5,0)
mysolver.periodicBC("Face",S1562,S4873,V1,V4,0)
mysolver.periodicBC("Face",S1584,S2673,V1,V2,0)

mysolver.periodicBC("Face",S1234,S5678,V1,V5,1)
```

```
mysolver.periodicBC("Face",S1562,S4873,V1,V4,1)
mysolver.periodicBC("Face",S1584,S2673,V1,V2,1)

mysolver.periodicBC("Face",S1234,S5678,V1,V5,2)
mysolver.periodicBC("Face",S1562,S4873,V1,V4,2)
mysolver.periodicBC("Face",S1584,S2673,V1,V2,2)

# eliminating edges intersection in faces
mysolver.addPeriodicEdgePhysicals(L12)
mysolver.addPeriodicEdgePhysicals(L23)
mysolver.addPeriodicEdgePhysicals(L34)
mysolver.addPeriodicEdgePhysicals(L41)
mysolver.addPeriodicEdgePhysicals(L56)
mysolver.addPeriodicEdgePhysicals(L67)
mysolver.addPeriodicEdgePhysicals(L78)
mysolver.addPeriodicEdgePhysicals(L85)
mysolver.addPeriodicEdgePhysicals(L15)
mysolver.addPeriodicEdgePhysicals(L26)
mysolver.addPeriodicEdgePhysicals(L37)
mysolver.addPeriodicEdgePhysicals(L48)

# add edge PBCs
mysolver.periodicBC("Edge",L12,L34,V1,V4,0)
mysolver.periodicBC("Edge",L12,L78,V1,V8,0)
mysolver.periodicBC("Edge",L12,L56,V1,V5,0)

mysolver.periodicBC("Edge",L12,L34,V1,V4,1)
mysolver.periodicBC("Edge",L12,L78,V1,V8,1)
mysolver.periodicBC("Edge",L12,L56,V1,V5,1)

mysolver.periodicBC("Edge",L12,L34,V1,V4,2)
mysolver.periodicBC("Edge",L12,L78,V1,V8,2)
mysolver.periodicBC("Edge",L12,L56,V1,V5,2)

mysolver.periodicBC("Edge",L41,L23,V1,V2,0)
mysolver.periodicBC("Edge",L41,L67,V1,V6,0)
mysolver.periodicBC("Edge",L41,L85,V1,V5,0)

mysolver.periodicBC("Edge",L41,L23,V1,V2,1)
mysolver.periodicBC("Edge",L41,L67,V1,V6,1)
mysolver.periodicBC("Edge",L41,L85,V1,V5,1)

mysolver.periodicBC("Edge",L41,L23,V1,V2,2)
mysolver.periodicBC("Edge",L41,L67,V1,V6,2)
mysolver.periodicBC("Edge",L41,L85,V1,V5,2)

mysolver.periodicBC("Edge",L15,L26,V1,V2,0)
mysolver.periodicBC("Edge",L15,L37,V1,V3,0)
mysolver.periodicBC("Edge",L15,L48,V1,V4,0)
```

```
mysolver.periodicBC("Edge",L15,L26,V1,V2,1)
mysolver.periodicBC("Edge",L15,L37,V1,V3,1)
mysolver.periodicBC("Edge",L15,L48,V1,V4,1)

mysolver.periodicBC("Edge",L15,L26,V1,V2,2)
mysolver.periodicBC("Edge",L15,L37,V1,V3,2)
mysolver.periodicBC("Edge",L15,L48,V1,V4,2)

# eliminating nodes intersection in edges
mysolver.addPeriodicNodePhysicals(V1)
mysolver.addPeriodicNodePhysicals(V2)
mysolver.addPeriodicNodePhysicals(V3)
mysolver.addPeriodicNodePhysicals(V4)
mysolver.addPeriodicNodePhysicals(V5)
mysolver.addPeriodicNodePhysicals(V6)
mysolver.addPeriodicNodePhysicals(V7)
mysolver.addPeriodicNodePhysicals(V8)

#node PBC
mysolver.periodicBC("Node",V4,V3,V1,V2,0)
mysolver.periodicBC("Node",V4,V3,V1,V2,1)
mysolver.periodicBC("Node",V4,V3,V1,V2,2)

mysolver.periodicBC("Node",V2,V6,V1,V5,0)
mysolver.periodicBC("Node",V2,V6,V1,V5,1)
mysolver.periodicBC("Node",V2,V6,V1,V5,2)

mysolver.periodicBC("Node",V3,V7,V1,V5,0)
mysolver.periodicBC("Node",V3,V7,V1,V5,1)
mysolver.periodicBC("Node",V3,V7,V1,V5,2)

mysolver.periodicBC("Node",V4,V8,V1,V5,0)
mysolver.periodicBC("Node",V4,V8,V1,V5,1)
mysolver.periodicBC("Node",V4,V8,V1,V5,2)

# prescribe deformaion mode on V1, V2, V4, V5
# uniaxial test following direction V1-V2
# node V1 is always fixed
mysolver.displacementBC("Node",V1,0,0.)
mysolver.displacementBC("Node",V1,1,0.)
mysolver.displacementBC("Node",V1,2,0.)
mysolver.displacementBC("Node",V2,1,0.)
mysolver.displacementBC("Node",V2,2,0.)
mysolver.displacementBC("Node",V4,2,0.)
# prescribed displacement following x-direction
mysolver.displacementBC("Node",V2,0,disp)

# if nonlocal variable is considered with PBC
```

```
#
mysolver.sameDisplacementBCBetweenTwoGroups("Face",S1234,S5678,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Face",S1562,S4873,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Face",S1584,S2673,3)

mysolver.sameDisplacementBCBetweenTwoGroups("Edge",L12,L34,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Edge",L12,L78,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Edge",L12,L56,3)

mysolver.sameDisplacementBCBetweenTwoGroups("Edge",L41,L23,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Edge",L41,L67,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Edge",L41,L85,3)

mysolver.sameDisplacementBCBetweenTwoGroups("Edge",L15,L26,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Edge",L15,L37,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Edge",L15,L48,3)

mysolver.sameDisplacementBCBetweenTwoGroups("Node",V1,V2,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Node",V1,V3,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Node",V1,V4,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Node",V1,V5,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Node",V1,V6,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Node",V1,V7,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Node",V1,V8,3)
```

3.2.2 Periodic BC combined with minimal kinematic BC

In a 3D analysis, the PBC is constrained in x-y direction and stress-free in z-direction

```
# pbc for faces
mysolver.periodicBC("Face",S1562,S4873,V1,V4,0)
mysolver.periodicBC("Face",S1584,S2673,V1,V2,0)
mysolver.periodicBC("Face",S1562,S4873,V1,V4,1)
mysolver.periodicBC("Face",S1584,S2673,V1,V2,1)
mysolver.periodicBC("Face",S1562,S4873,V1,V4,2)
mysolver.periodicBC("Face",S1584,S2673,V1,V2,2)

# eliminating edges intersection in faces
mysolver.addPeriodicEdgePhysicals(L15)
mysolver.addPeriodicEdgePhysicals(L26)
mysolver.addPeriodicEdgePhysicals(L37)
mysolver.addPeriodicEdgePhysicals(L48)

# add edge PBCs
mysolver.periodicBC("Edge",L15,L26,V1,V2,0)
mysolver.periodicBC("Edge",L15,L37,V1,V3,0)
mysolver.periodicBC("Edge",L15,L48,V1,V4,0)

mysolver.periodicBC("Edge",L15,L26,V1,V2,1)
mysolver.periodicBC("Edge",L15,L37,V1,V3,1)
```

```
mysolver.periodicBC("Edge",L15,L48,V1,V4,1)

mysolver.periodicBC("Edge",L15,L26,V1,V2,2)
mysolver.periodicBC("Edge",L15,L37,V1,V3,2)
mysolver.periodicBC("Edge",L15,L48,V1,V4,2)

# eliminating nodes intersection in edges
mysolver.addPeriodicNodePhysicals(V1)
mysolver.addPeriodicNodePhysicals(V2)
mysolver.addPeriodicNodePhysicals(V3)
mysolver.addPeriodicNodePhysicals(V4)

#node PBC
mysolver.periodicBC("Node",V4,V3,V1,V2,0)
mysolver.periodicBC("Node",V4,V3,V1,V2,1)
mysolver.periodicBC("Node",V4,V3,V1,V2,2)

# prescribe deformaion mode on V1, V2, V4, V5
...
..

# if nonlocal variable is considered with PBC
#
mysolver.sameDisplacementBCBetweenTwoGroups("Face",S1562,S4873,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Face",S1584,S2673,3)

mysolver.sameDisplacementBCBetweenTwoGroups("Edge",L15,L26,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Edge",L15,L37,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Edge",L15,L48,3)

mysolver.sameDisplacementBCBetweenTwoGroups("Node",V1,V2,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Node",V1,V3,3)
mysolver.sameDisplacementBCBetweenTwoGroups("Node",V1,V4,3)
```