Scripts

David Hernández-Aristizábal

September 20, 2022

Contents

1 Main python scripts **1**
- 1.1 Run analysis . . . 1
  - 1.1.1 Execution . . . 1
  - 1.1.2 Required packages . . . 1
  - 1.1.3 General function . . . 2
- 1.2 Initial geometry . . . 2
  - 1.2.1 Execution . . . 2
  - 1.2.2 Required packages . . . 2
  - 1.2.3 General function . . . 3
- 1.3 Remesh . . . 3
  - 1.3.1 Execution . . . 3
  - 1.3.2 Required packages . . . 3
  - 1.3.3 General function . . . 4

2 Code_Aster script: Bio–inspired growth **4**
- 2.1 Execution . . . 4
- 2.2 Required packages . . . 4
- 2.3 General function . . . 5

1 Main python scripts

1.1 Run analysis

1.1.1 Execution

```
python3 RunAnalysis.py -i PARAMS.json
```

1.1.2 Required packages

```
# Python libraries
import os
import sys
import json
```

```
import getopt
import time
# In-house modules
sys.path.append('../PythonUtilities')
import AsterStudyUtilities as asus
```

1.1.3 General function

```
# Read parameters
params = json.load("PARAMS.json")
# Create Code_Aster study
study = asus.AsterStudy()
# Set up study parameters
study.PARAMETER = PARAMETER
study.comm = ".../BioInspiredGrowth.comm"
study.dumm = ".../BioInspiredGrowth.dumm"
# Create EXPORT file
study.CreateStudy()
# Create initial geometry
os.system("python3 .../InitialGeometry.py -i PARAMS.json")
# Run Code_Aster study
study.RunStudy(outSalome = True)
while notFinish:
    # Remesh geometry
    os.system("python3 .../Remesh.py -i PARAMS.json")
    # Run Code_Aster study
    study.RunStudy(outSalome = True)
```

1.2 Initial geometry

1.2.1 Execution

```
python3 InitialGeometry.py -i PARAMS.json
```

1.2.2 Required packages

```
# Python libraries
import os
import sys
import json
import getopt
import gmsh
import math
import numpy
# In-house modules
sys.path.append('../PythonUtilities')
import ReadGmsh as rgmsh
```

1.2.3 General function

```
# Read parameters
params = json.load("PARAMS.json")
# Make the deformable geometry
gmsh.initialize()
wp_face = geo.addPlaneSurface([wp_loop])
# Approximation of contact length
fine_x = closestPoint()
# Definition of the field of element sizes
mesh.field.setAsBackgroundMesh(bacFieldNum)
# Mesh of the deformable geometry
mesh.generate(2)
gmsh.write("geomwp.unv")
gmsh.finalize()
# Make the non-deformable geometry
gmsh.initialize()
base_face = geo.addPlaneSurface([base_loop])
# Definition of the field of element sizes
bsFields = rgmsh.BoxFieldsIncreasingSize(...fine_x...)
# Mesh the non-deformable geometry
mesh.generate(2)
gmsh.write("geombs.unv")
gmsh.finalize()
```

1.3 Remesh

1.3.1 Execution

```
python3 Remesh.py -i PARAMS.json
```

1.3.2 Required packages

```
# Python libraries
import os
import sys
import json
import getopt
import gmsh
import numpy
# In-house modules
sys.path.append('../PythonUtilities')
import ReadGmsh as rgmsh
```

1.3.3 General function

```
# Read parameters
params = json.load("PARAMS.json")
# Reconstruction of the deformable geometry
gmsh.initialize()
model = rgmsh.Remesh(...)
# Approximation of contact length
fine_x = closestPoint()
# Definition of the field of element sizes
mesh.field.setAsBackgroundMesh(bacFieldNum)
# Remesh the deformable geometry
mesh.generate(2)
gmsh.write("geomwp.unv")
gmsh.finalize()
# Make the non-deformable geometry
gmsh.initialize()
base_face = geo.addPlaneSurface([base_loop])
# Definition of the field of element sizes
bsFields = rgmsh.BoxFieldsIncreasingSize(...fine_x...)
# Remesh the non-deformable geometry
mesh.generate(2)
gmsh.write("geombs.unv")
gmsh.finalize()
```

2 Code_Aster script: Bio–inspired growth

2.1 Execution

```
.../salome shell -- as_run setup.export
```

2.2 Required packages

```
# Python libraries
import os
import math
import sys
import json
import numpy as np
import time
# Code_Aster libraries
from Utilitai.partition import *
# In-house modules
sys.path.append('../PythonUtilities')
import MorphoDesignFunctions as mdf
```

```
from FemMesh2D import FemMesh2D as fm2
from FemVtk import FemVtk as fvt
```

2.3 General function

```
# Read parameters
params = json.load("PARAMS.json")
# Definition of Code_Aster static concepts: material, mesh and model
mesh_wp = LIRE_MAILLAGE(FORMAT = 'IDEAS', ...)
mesh_bs = LIRE_MAILLAGE(FORMAT = 'IDEAS', ...)
mesh    = CREA_MAILLAGE(DECOUPE_LAC = _F(...), ...)
mode    = AFFE_MODELE(AFFE = _F(MODELISATION = ('D_PLAN'), ...)
mater   = DEFI_MATERIAU(ELAS = (...))
matf    = AFFE_MATERIAU(AFFE = _F(MATER= mater, ...), MODELE = mode)
# Set time--dependent control variables
ite = ...
bestContPre = ...
# Define initial soft field (where the material can grow)
toGrw0 = CREA_CHAMP(...)
# Adjust the location of the geometries (necessary to ensure contact
# convergence)
if not interpenetration:
    reloc = CREA_CHAMP(...)
    mesh  = MODI_MAILLAGE(...)
# Set up mesh and define loads
mesh    = DEFI_GROUP(...)
...     = AFFE_CHAR_MECA(...)
contact = DEFI_CONTACT(...)
# Definition of formulas and function fields
... = FORMULE(...)
... = CREA_CHAMP(...)
# Bio--inspired growth iterations
for k1 in range(number of iterations):
    # Compute the stress field that induces growth (with contact
    # conditions)
    resu = STAT_NON_LINE(...)
    # Compute the hydrostatic and the shear stress
    resu = CALC_CHAMP(CONTRAINTE = ('SIGM_NOEU', 'SIGM_ELGA',
                                    'SIEF_NOEU'),
                      CRITERES = ('SIEQ_ELGA'),...)
    resu = CALC_CHAMP(CHAM_UTIL = _F(FORMULE = (Shyd_f, Svon_f),...)
    # Compute the contact pressure and its quality
    cont_p = CALC_PRESSION(...)
    Rp = mdf.Rp_OrderedData(...)
    # Compute the quality of the shear stress
    Rv = 1.0 - (maxTau - tauRef)/(maxTau + tauRef)
```

```
# Compute the growth function
Sgrw_f  = FORMULE(func = mdf.SgrowthDesign, ...)
resu    = CALC_CHAMP(CHAM_UTIL = (_F(FORMULE = (Sgrw_f), ...)))
grw_fi = CREA_CHAMP(OPERATION = 'EXTR', ...)
toGrw  = CREA_CHAMP(OPERATION = 'EVAL', ...)
# Get field norm
grwStrVtk = fvt(...)
grwNorm = grwStrVtk.LpNormScalar(0)
# Scale field with the norm
toGrwSca = CREA_CHAMP(OPERATION = 'COMB', ...)
# Compute growth stress
growth_t = CREA_CHAMP(OPERATION = 'ASSE', ...)
# Compute the displacements driven by the growth stress
resu = MECA_STATIQUE(...)
# Apply displacements
disp_ap1 = CREA_CHAMP(OPERATION = 'EXTR', ...)
disp_ap2 = CREA_CHAMP(OPERATION = 'EVAL', ...)
disp_ap3 = CREA_CHAMP(OPERATION = 'ASSE', ...)
mesh = MODI_MAILLAGE(reuse = mesh, DEFORME = (disp_ap3))
```