DEVELOPING PYTHON SCRIPTING FOR MESH CONVERGENCE IN ABAQUS

Duc Hieu Vu[1,*]

[1]*Le Quy Don Technical University*

Abstract

The difficulty in solving finite element programs is to estimate the appropriate element size to obtain a reliable solution. Therefore, the element(s) size and meshing density is required to be determined reasonably. However, almost models ignored that problem or have to solve it by hand. In addition, the processes recreate meshing, and updating the total model is a really big deal because of time-consuming. To establish a proper mesh convergence method, it is required to plot the curve for a critical result (typically stress or displacement) in a specific part or point in the considered model versus mesh density. To deal with it, the article is aimed to introduce how to use the Python script to estimate mesh convergence. The method uses the Python script automatically to refine the mesh and update the system. In addition, at least three convergence results are displayed to plot a curve which can be used to indicate the achieved convergence. Finally, the article also suggests some critical conditions for implicit and explicit problems to archive the optimum convergence.

Keywords: *Abaqus/PDE; mesh convergence; Python; CDP model; blast loading.*

1. Introduction

In finite element method (FEM), convergence means arriving at the true solution of the partial differential equations (PDEs) as the geometry, or spatial domain is meshed more finely [1-4]. The process of mesh convergence includes decreasing the element size and analyzing the impact of this process on the accuracy of the solution.

Typically, the smaller the mesh size is, the more accurate the solution will be reached. However, the higher the accuracy is, the larger the simulations can become in terms of data to store and handle, which translates to longer runtimes. Sometimes, because of misunderstanding physical problems and the kind of required accuracy, engineers spend a lot of time and effort trying to solve a problem. Abaqus and other FEA software have introduced sub-modeling techniques and adaptive mesh for engineers to use it. Unlike these complex ways, the article focuses on how to apply automatically Python scripting for refinement. The procedure can semi-automatically create, submit and calculate models to achieve the proper result.

[*] Email: vuduchieu.v2@gmail.com

https://doi.org/10.56651/lqdtu.jst.v5.n01.365.sce

2. Mesh convergence problems

To ensure the results of Abaqus simulation are satisfactory, mesh density has to refine sufficiently. Coarse meshes can yield inaccurate results in analyses using implicit or explicit methods. The numerical solution provided by the user model will tend toward a unique value as increasing the mesh density. However, the computational resources required to run the model increase as the mesh is finely refined. The mesh is said to be converged when further mesh refinement produces a negligible change in the solution [4].

There is no way detail describe how to set up the size and the density of the mesh. Abaqus experts advise that users should predict the locations of the highly stressed regions of a model - and, hence, the regions where a fine mesh is required - using their knowledge of similar components or with hand calculations.

This information can also be gained by using a coarse mesh initially to identify the regions of high stress and then refining the mesh in these regions. It is simple to mesh the geometry coarsely for the initial simulation and then refine the mesh in the appropriate regions, as indicated by the results from the coarse simulation.

Because of different algorithms, both explicit and implicit solvers impact the mesh convergence of the model also differently.

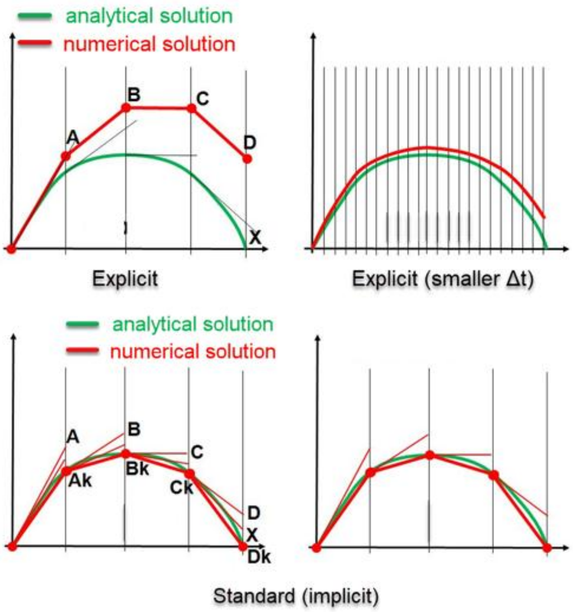


Figure 1. Time incrementation with explicit and implicit (standard) solver vs accuracy [3].

In Figure 1, larger time increments lead to larger errors on the solution for explicit (top row). For implicit, larger time increments do not lead to larger errors in the solution, due to the iterative approach used (Newton-Raphson method). However, the accuracy only reaches the mesh points. Engineers normally refine the mesh only in the interesting region to get negligible results.

The best way to deal with it is using meshing adaptive, sub modeling technique, or changing mesh size technique [2, 3, 5, 6]. The article is only focused on introducing and giving the algorithm to develop automatically resize and refind global meshing by using Python.

To clarify that, the article researched and showed two case studies: the Abaqus standard and explicit.

3. Mesh convergence for Abaqus standard

To study how to get mesh convergence, both generating model and then editing Python code have been introduced.

3.1. Generating model

Abaqus standard, the implicit solver, can solve for static equilibrium: the state where the sum of the forces is zero. It requires non-linear solution algorithms, typically the Newton-Raphson method. Therefore, Abaqus/Standard is more beneficial for slow and relatively linear problems like structures under the effect of static and semi-dynamic loading [7].

Hence, mesh convergence for the solver can be acceptable when negligible change of stress or displacement.

The basic model of a steel rectangular beam has been surveyed to clarify and apply the Python code. The model will be shown in the following figure.

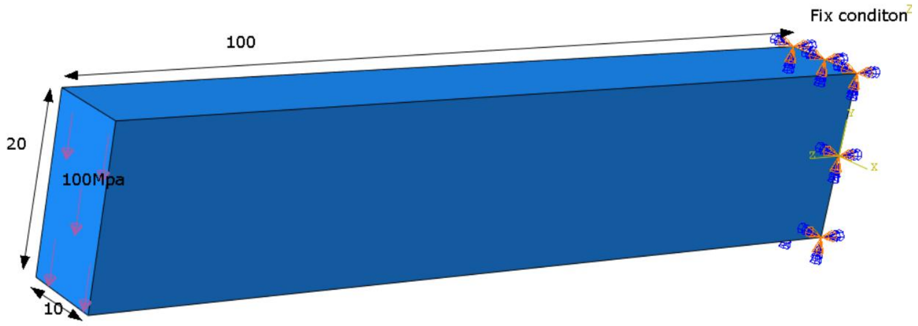
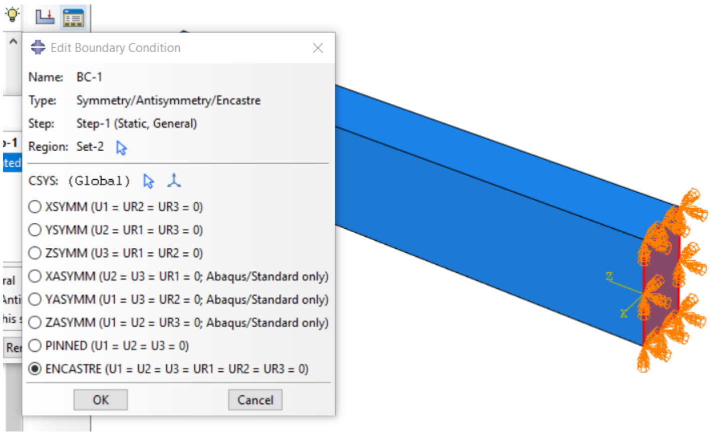


Figure 2. The beam's geometry.

- Material: Steel, elasticity model with 200e3 MPa for Young's modulus and 0.3 for Poisson's ratio.

- Section Properties and assignment: The beam is assumed to be homogenous. The section has to be created and apply for the beam in Section part.

- Define boundary Condition: In the model, One end was clamped. It means one surface has an encastre condition.



Edit Boundary Condition

Name: BC-1

Type: Symmetry/Antisymmetry/Encastre

Step: Step-1 (Static, General)

Region: Set-2

CSYS: (Global)

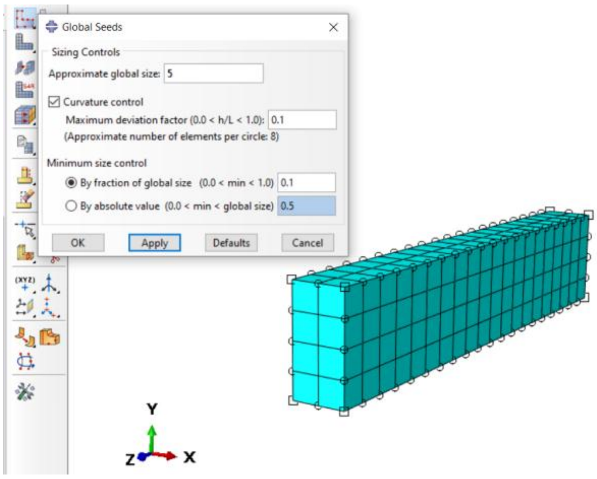
○ XSYMM (U1 = UR2 = UR3 = 0)
○ YSYMM (U2 = UR1 = UR3 = 0)
○ ZSYMM (U3 = UR1 = UR2 = 0)
○ XASYMM (U2 = U3 = UR1 = 0; Abaqus/Standard only)
○ YASYMM (U1 = U3 = UR2 = 0; Abaqus/Standard only)
○ ZASYMM (U1 = U2 = UR3 = 0; Abaqus/Standard only)
○ PINNED (U1 = U2 = U3 = 0)
● ENCASTRE (U1 = U2 = U3 = UR1 = UR2 = UR3 = 0)

Figure 3. Defining the clamped boundary condition.

Step module: The static shear load is applied on the beam, while displacements should be considered in the analysis. However, this is a nonlinear problem. Nonlinear analysis is presented in General analysis in Abaqus and created in Step part.

Load module: The beam has under the impact of 100 MPa distributed shear area loading (Figure 1).

Mesh module: The most challenging part of this problem is the mesh size. The main challenge here is what element size is suitable for the problem. The issue will be answered using scripting. In the first step, the mesh seed has a size of 5 mm.



Global Seeds

Sizing Controls

Approximate global size: 5

☑ Curvature control

Maximum deviation factor (0.0 < h/L < 1.0): 0.1

(Approximate number of elements per circle: 8)

Minimum size control

● By fraction of global size (0.0 < min < 1.0) 0.1
○ By absolute value (0.0 < min < global size) 0.5

Figure 4. Seeding and meshing part.

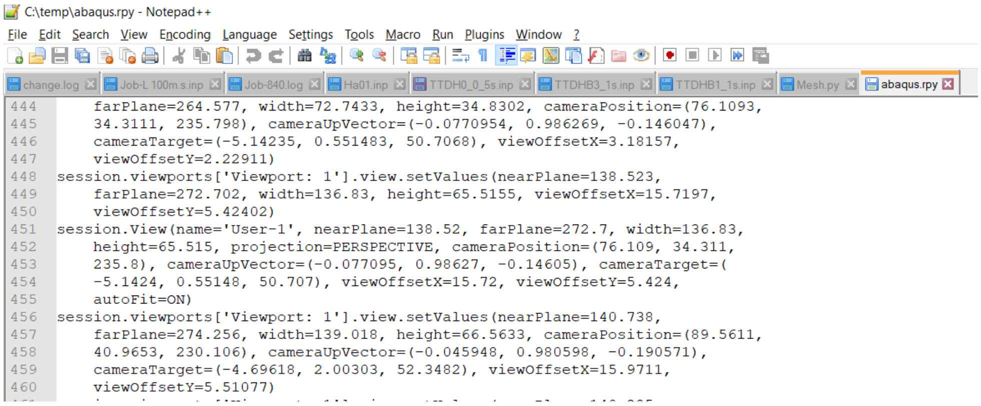
Analysis: Job module: Create the job name job-1 and submit the model to the kernel for calculation. The result can be shown after running without any errors.

The above procedure is typical for any problems. The big challenge here is how to know the proper size for the model. The user can repeat action by refined mesh, repair the whole model, resubmit and run the model again and again. It takes a lot of time and can be solved by applying the Python code method below.

3.2. Using Python code to create a multi-mesh model

Abaqus supports Python language in the PDE environment to create, edit and run models. After using the Abaqus/CAE user interface to create and submit a model for Abaqus kernel checking and calculating, Abaqus automatically saves model in Python langague (Abaqus. rpy file). It contains all information on how to create a geometric model and change the settings in all windows that are used in each module [8].

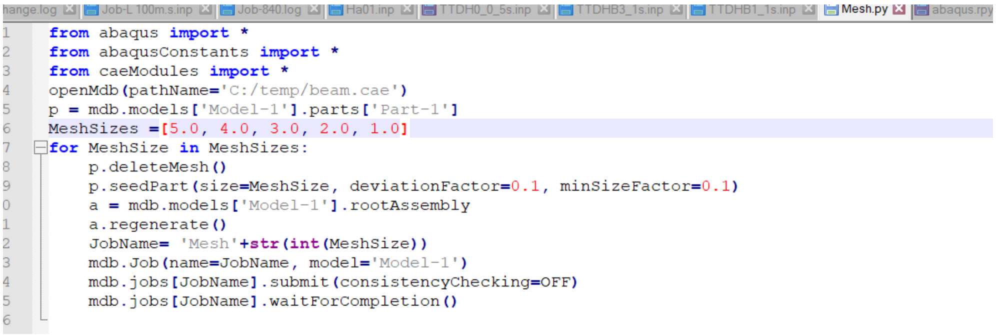
Open Abaqus.rpy by notepad ++ editor and shown in Figure 5.


C:\temp\abaqus.rpy - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
change.log
Job-L 100m.s.inp
Job-840.log
Ha01.inp
TTDH0_0_5s.inp
TTDHB3_1s.inp
TTDHB1_1s.inp
Mesh.py
abaqus.rpy

```
    farPlane=264.577, width=72.7433, height=34.8302, cameraPosition=(76.1093,
    34.3111, 235.798), cameraUpVector=(-0.0770954, 0.986269, -0.146047),
    cameraTarget=(-5.14235, 0.551483, 50.7068), viewOffsetX=3.18157,
    viewOffsetY=2.22911)
session.viewports['Viewport: 1'].view.setValues(nearPlane=138.523,
    farPlane=272.702, width=136.83, height=65.5155, viewOffsetX=15.7197,
    viewOffsetY=5.42402)
session.View(name='User-1', nearPlane=138.52, farPlane=272.7, width=136.83,
    height=65.515, projection=PERSPECTIVE, cameraPosition=(76.109, 34.311,
    235.8), cameraUpVector=(-0.077095, 0.98627, -0.14605), cameraTarget=(
    -5.1424, 0.55148, 50.707), viewOffsetX=15.72, viewOffsetY=5.424,
    autoFit=ON)
session.viewports['Viewport: 1'].view.setValues(nearPlane=140.738,
    farPlane=274.256, width=139.018, height=66.5633, cameraPosition=(89.5611,
    40.9653, 230.106), cameraUpVector=(-0.045948, 0.980598, -0.190571),
    cameraTarget=(-4.69618, 2.00303, 52.3482), viewOffsetX=15.9711,
    viewOffsetY=5.51077)
```

*Figure 5. Opening *.rpy file contains Python scripts.*

Saving Abaqus.rpy as a python file (for example mesh.py). Then, insert the new codes that had developed at the end of your file to create the new size of the mesh, regenerating the model again and automatically submitting the model.



```
from abaqus import *
from abaqusConstants import *
from caeModules import *
openMdb(pathName='C:/temp/beam.cae')
p = mdb.models['Model-1'].parts['Part-1']
MeshSizes =[5.0, 4.0, 3.0, 2.0, 1.0]
for MeshSize in MeshSizes:
    p.deleteMesh()
    p.seedPart(size=MeshSize, deviationFactor=0.1, minSizeFactor=0.1)
    a = mdb.models['Model-1'].rootAssembly
    a.regenerate()
    JobName= 'Mesh'+str(int(MeshSize))
    mdb.Job(name=JobName, model='Model-1')
    mdb.jobs[JobName].submit(consistencyChecking=OFF)
    mdb.jobs[JobName].waitForCompletion()
```

Figure 6. Final Python code.

Finally, the code can run in PDE environment. The introduced results in the following figure show the Mises stress at integration points:

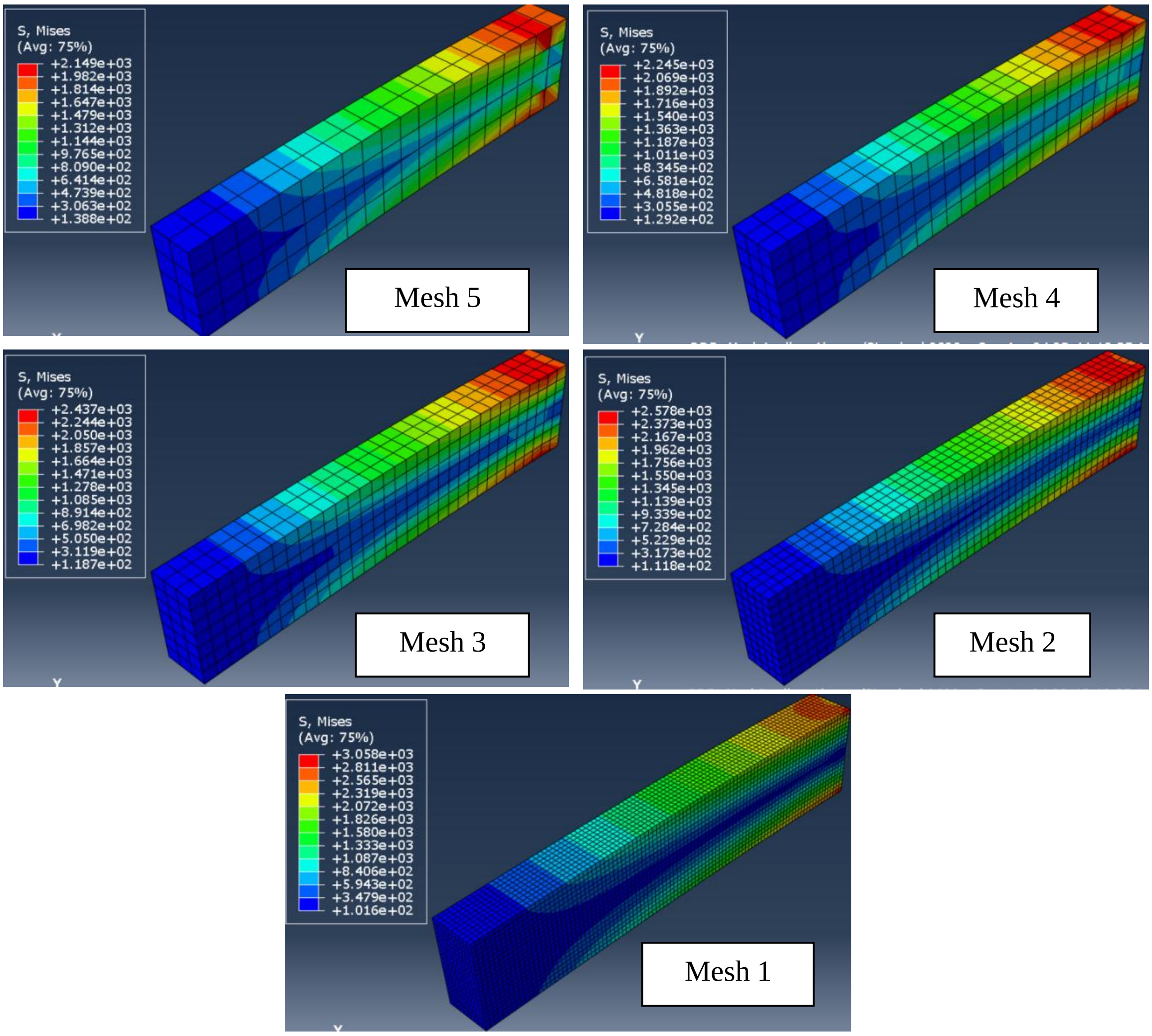


Figure 7. Different results are based on different mesh sizes.

The maximum displacements and Mises stress are also illustrated in Table 1.

Table 1. Maximum displacement for each mesh size

Mesh size	Maximum Mises stress (MPa)	Maximum displacement (mm)
5	2149	5.131
4	2245	5.140
3	2437	5.149
2	2578	5.157
1	3058	5.164

Table 1 shows that the displacement results are convergence. However, in the case of using Mises stress for calculating strength and design, the mesh size of 1 mm or 2 mm is higher accuracy.

4. Mesh covergance for Abaqus Explicit

Explicit dynamic analysis is computationally efficient for the analysis of large models with relatively short dynamic response times and the analysis of extremely discontinuous events or processes such as impact, blast, contact, car test, etc. Unlike implicit, explicit maybe has large error problems if the density of mesh is not enough. To explore it, let’s survey a blast loading impact on the reinforced concrete.

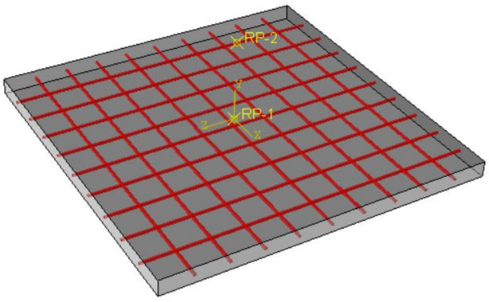
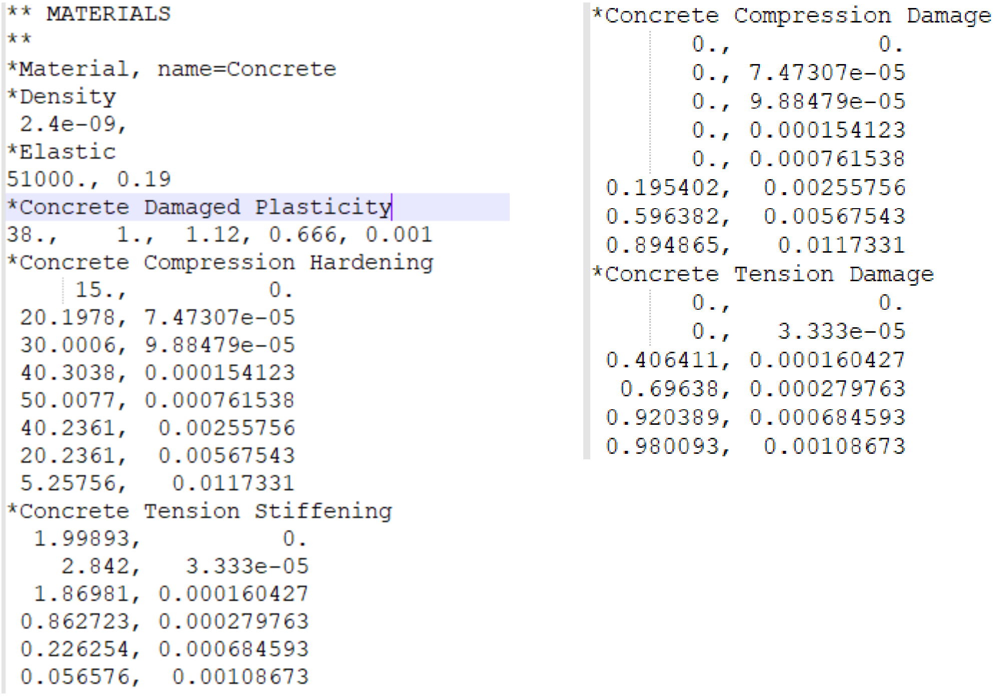


Figure 8. Different results are based on different mesh sizes.

Geometry: Reinforced concrete plate with 1000 x 1000 x 100 mm.

Steel mesh: One layer with a size of 100 x 100 mm.

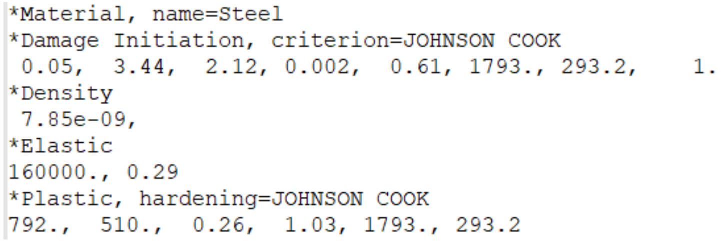
Materials: Using concrete damage plasticity for concrete parts [9] and the Johnson-Cook model for steel parts. The parameter of the materials model will be indicated in the following figure [10, 11].



```
** MATERIALS
**
*Material, name=Concrete
*Density
 2.4e-09,
*Elastic
51000., 0.19
*Concrete Damaged Plasticity
38.,    1.,  1.12, 0.666, 0.001
*Concrete Compression Hardening
    15.,           0.
 20.1978, 7.47307e-05
 30.0006, 9.88479e-05
 40.3038, 0.000154123
 50.0077, 0.000761538
 40.2361,  0.00255756
 20.2361,  0.00567543
 5.25756,   0.0117331
*Concrete Tension Stiffening
  1.99893,          0.
    2.842,   3.333e-05
  1.86981, 0.000160427
 0.862723, 0.000279763
 0.226254, 0.000684593
 0.056576,  0.00108673
```

```
*Concrete Compression Damage
      0.,          0.
      0., 7.47307e-05
      0., 9.88479e-05
      0., 0.000154123
      0., 0.000761538
 0.195402,  0.00255756
 0.596382,  0.00567543
 0.894865,   0.0117331
*Concrete Tension Damage
      0.,          0.
      0.,   3.333e-05
 0.406411, 0.000160427
  0.69638, 0.000279763
 0.920389, 0.000684593
 0.980093,  0.00108673
```

Figure 9. Parameters of concrete material model.

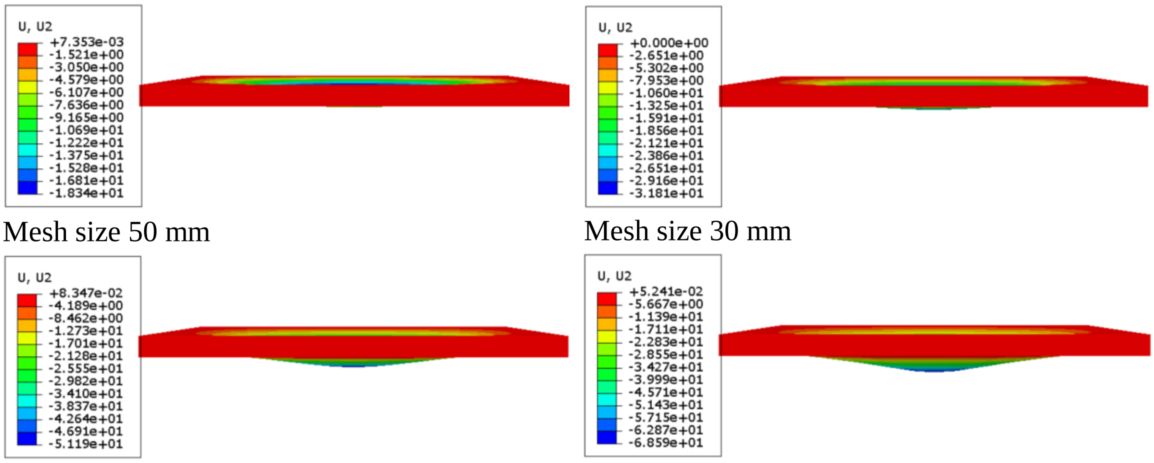


```
*Material, name=Steel
*Damage Initiation, criterion=JOHNSON COOK
 0.05,  3.44,  2.12, 0.002,  0.61, 1793., 293.2,    1.
*Density
 7.85e-09,
*Elastic
160000., 0.29
*Plastic, hardening=JOHNSON COOK
792.,  510.,  0.26,  1.03, 1793., 293.2
```

Figure 10. Parameters of steel material model.

Blast interaction: The structure has been affected by blast loading that was created CONWEP module. The charge of detonation is 1 kg and the distance between the source point and the surface of the plate is 1000 mm.

The mesh convergence has been conducted by using global mesh size (50; 30; 10; 5 mm). Similarly, using the Python script above, the result for different sizes will be introduced below.



Mesh size 50 mm

Mesh size 30 mm

Mesh size 10 mm

Mesh size 5 mm

Figure 11. Graphic Maximum displacement.

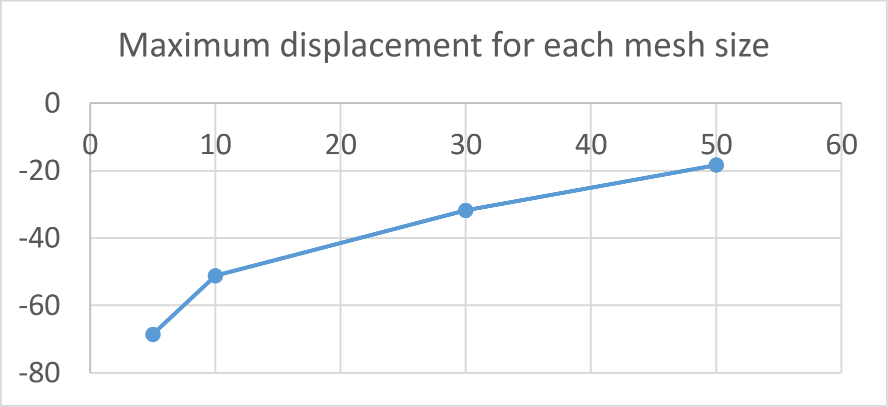


Figure 12. Maximum displacement for each mesh size.

Although the mesh size keeps going small, the maximum displacement still increases rapidly. The error caused in the case comes from the plasticity state of concrete and lacking erosion conditions. Instead of damage and blowing up, the concrete elements are still working and are tensile like metal. To deal with it, the damage to tensile should be discovered and shown in the next figure.

The result shows that the mesh size is acceptable with size of 5 or 10 mm.

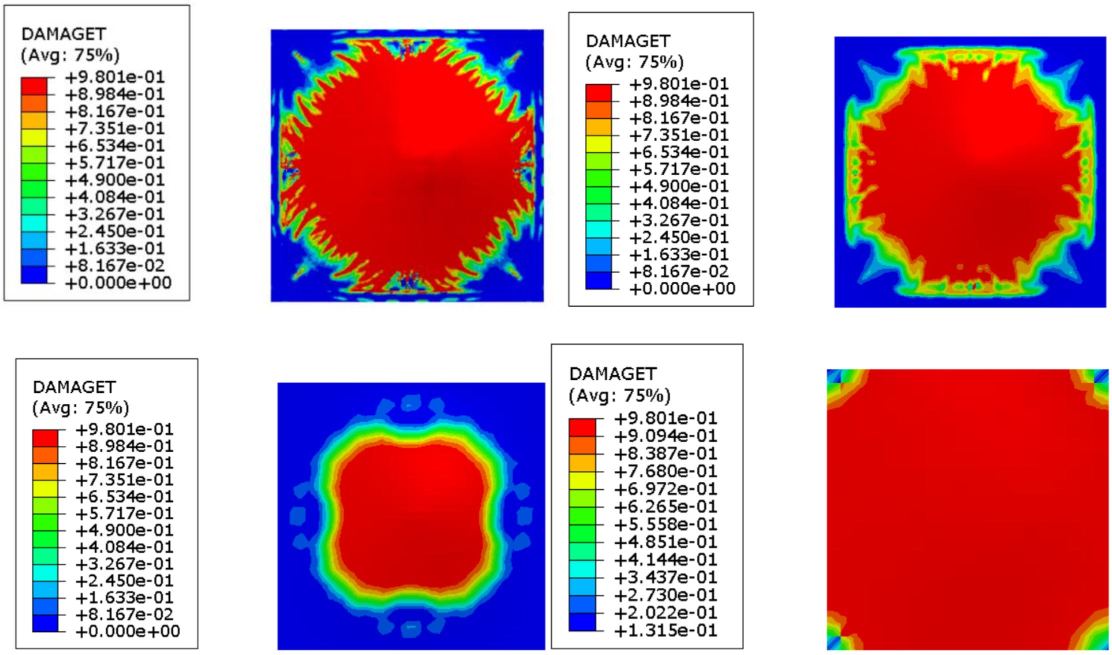


Figure 13. Damage of tensile of different mesh sizes.

5. Conclusion

The article provides the concept of the Abaqus script for users who want to learn Python in Abaqus. Based on the additional script, two cases of mesh independency study have been investigated. The results show that mesh convergence is important and different between implicit and explicit solvers. Engineers have to estimate mesh density carefully based on the adaptable condition to reach accuracy. The article is aimed to build an example and easy using code for engineers. Developing more complex algorism, especially creating adaptive mesh or meshing with different sizes for different parts will be researched and introduced in the future.

References

[1] R. J. Boulbes, “Analysis Convergence Guidelines,” in *Troubleshooting Finite-Element Modeling with Abaqus*, Cham: Springer International Publishing, 2020, pp. 17-23. doi: 10.1007/978-3-030-26740-7_2.

[2] M. Acin, Stress singularities, stress concentrations and mesh convergence, Apr. 2015.

[3] ABAQUS, Abaqus Theory Manual, 2003.

[4] Simulia D., ABAQUS 6.11 analysis user's manual, 2011.

[5] R. J. Boulbes, "Analysis Convergence Guidelines," in *Troubleshooting Finite-Element Modeling with Abaqus*, Cham: Springer International Publishing, 2020, pp. 17-23. doi: 10.1007/978-3-030-26740-7_2.

[6] M. K. Thompson and J. M. Thompson, "Determining the Mesh Convergence of a Heated Plate With a Central Hole," in *ANSYS Mechanical APDL for Finite Element Analysis*, Elsevier, 2017, pp. 201-238. doi: 10.1016/B978-0-12-812981-4.00023-X.

[7] "Introduction: From Implicit to Explicit Theories," in *Contemporary Sociological Theory*, New York, NY: Springer New York, 2008, pp. 3-21. doi: 10.1007/978-0-387-76522-8_1.

[8] Puri G. M., *Python Scripts for ABAQUS: Learn by Example*, 2011.

[9] https://abaqus-docs.mit.edu/2017/English/SIMACAEMATRefMap/simamat-c-concretedamaged.htm.

[10] M. Hafezolghorani, F. Hejazi, R. Vaghei, M. S. bin Jaafar, and K. Karimzade, "Simplified Damage Plasticity Model for Concrete," *Structural Engineering International*, Vol. 27, No. 1, pp. 68-78, Feb. 2017, doi: 10.2749/101686616X1081.

[11] M. Žmindák, Z. Pelagić, P. Pastorek, M. Močilan, and M. Vyboštʼok, "Finite Element Modelling of High-Velocity Impact on Plate Structures," *Procedia Engineering*, Vol. 136, pp. 162-168, 2016, doi: 10.1016/j.proeng.2016.01.191.

PHÁT TRIỂN ĐOẠN MÃ PYTHON CHO TÍNH TOÁN HỘI TỤ LƯỚI CHIA TRONG ABAQUS

Vũ Đức Hiếu

Tóm tắt: *Một thách thức lớn trong việc xử lý các bài toán phần tử hữu hạn là xác định kích thước lưới phù hợp. Vì vậy, kích thước phần tử và mật độ lưới cần được định nghĩa hợp lý. Tuy nhiên, hầu hết các bài toán đều bỏ qua vấn đề này hoặc phải tính thủ công nhiều lần. Mặt khác, việc tạo lưới và cập nhật lại toàn bộ mô hình bài toán là một vấn đề lớn và tiêu tốn nhiều thời gian. Để thiết lập được phương pháp lưới hội tụ, cần phải vẽ đường đồ thị mô tả một giá trị kết quả quan trọng (thường là ứng suất hoặc chuyển vị) của một phần hoặc một điểm được cân nhắc ở mô hình với các mật độ lưới đã cho. Yêu cầu ít nhất ba điểm trên đồ thị để thể hiện sự hội tụ kết quả hoặc cần thiết phải định nghĩa sự hội tụ tối ưu tạo ra bởi kích thước lưới. Bài báo tập trung vào cách phát triển đoạn mã Python bổ sung để đánh giá sự hội tụ của lưới.*

Từ khóa: Abaqus/PDE; hội tụ lưới; Python; mô hình CDP; tải trọng nổ.

Received: 06/04/2022; *Revised:* 23/05/2022*; Accepted for publication:* 20/06/2022

❑