Adaptive 3D Remeshing for Metal Forming Simulation

S. Krishnamoorthi and M.C. Gean

nanoPrecision Products Inc, 411-B, Coral Circle, El Segundo, CA

Abstract: An alternative remeshing methodology for 3D Abaqus explicit analyses which utilizes mesh mapping is presented. Excessive element distortion in 3D large deformation analyses necessitates remeshing, however in Abaqus explicit, the standard 3D remeshing techniques limits the analyst's choice of element type. The presented technique allows remeshing of 3D perfectly plastic fully integrated higher order elements. The remeshing occurs in regions of high equivalent plastic strain or high equivalent plastic strain rate. The remeshing procedure maps the old boundary conditions, stress, and inelastic strain from the old distorted mesh. A benchmark comparing the results of an analysis using the remeshing technique to a known solution showed less than 0.2% error as a result of the remeshing procedure.

Keywords: Finite Element Analysis, Metal Forming, Abaqus/Explicit, Adaptive Remeshing, Python Scripting, API, Hypermesh

1. Introduction

In metal forming analyses such as coining and shaving there are situations in which remeshing the model between iterations is required due to high elemental distortion. The ALE adaptive meshing capability in Abaqus is limited in its capabilities by requiring the analyst to use either linear or reduced integration elements. However, there are situations in which there is a need to use fully integrated quadratic elements, and in these scenarios the only available tool is to add artificial viscosity to the elements with the Distortion Controls in Abaqus. These limited controls lead to severe elemental distortions, often causing the analysis to end prematurely. Thus, there is need to use remeshing to improve the element quality during these large deformation simulations.

2. Process Layout

2.1 Algorithm

The remeshing algorithm is shown in Figure 1. Python is the chosen programing language due to its integration in Abaqus. The simulations start with the user providing the time step information and the number of remeshing's steps. The Python scripts generate the necessary input files, and the substep simulations are performed. The simulation's deformed mesh is exported at then end of each substep using API programming routines. The deformed mesh is imported to an external remesher, Hypermesh. Hypermesh performs a remeshing of the computational domain, and the new mesh is exported out of Hypermesh. The results from the old mesh are read from the odb file

and extrapolated to the new mesh. The individual elements of the algorithm are described in greater detail in the following sections.

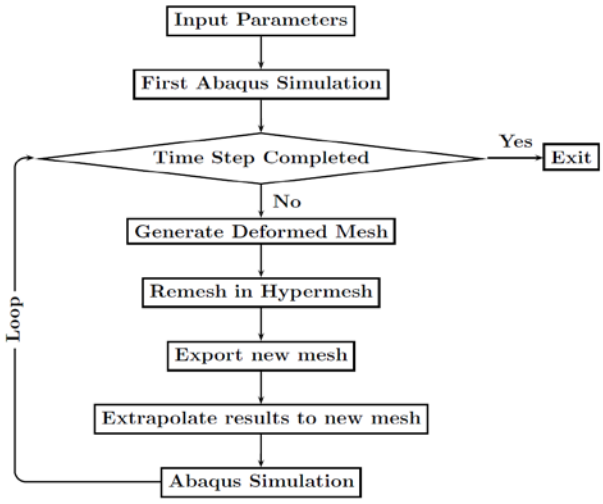


Figure 1. Adaptive Remeshing Algorithm

2.2 Setting up the problem

The Input Parameters are read into a python script reads from a text file. Here the user specifies: the number of remeshings, the total time step for the analysis, the target element size, and the remesh regions. The remesh regions are chosen based on the elemental Equivalent Plastic Strain or Equivalent Strain Rates. All intermediate files are generated automatically and are named as per the iteration number as *input_file_name_<it_number>*. *To* ease the generation of the input decks the input file is split into the deformable body's mesh and the rest of the analysis definition. The file containing the mesh definition is called the mesh file and the file containing the step information is called the step file. Using the **`*INCLUDE`** command the mesh file is included in the step file. The time interval between remeshing increments is

$$\Delta t = \frac{t_{\text{final}} - t_{\text{initial}}}{n} \qquad \textbf{Equation 1}$$

Where 'n' is the number of iterations. Thus, the first run is simulated for Δt duration, and then a remeshing is triggered.

2.3 Deformed Mesh Generation

A separate python script is executed at the end of the first iteration. This reads the odb file and calculates the deformed coordinates of all the deformable bodies in the model. The deformed coordinates are exported out as an Abaqus deck, and can be read into the meshing program. Element sets which have high equivalent strain or strain rate values are created in the model. The equivalent strain is calculated as

$$\varepsilon_{eq} = \sqrt{\frac{3}{2}\varepsilon_{ij} : \varepsilon_{ij}} \qquad \textbf{Equation 2}$$

An equivalent strain rate is calculated similarly. The user has an option of choosing which variable to use. The equivalent plastic strain or strain rate is used to capture the regions requiring remesh. The maximum and the minimum values of each of these variables are calculated. The user provides a limiting percentage value. The elements satisfying

$$\begin{aligned} \varepsilon &> \varepsilon_{\min} + f \times (\varepsilon_{\max} - \varepsilon_{\min}) \\ \dot{\varepsilon} &> \dot{\varepsilon}_{\min} + f \times (\dot{\varepsilon}_{\max} - \dot{\varepsilon}_{\min}) \end{aligned} \qquad \textbf{Equation 3}$$

are placed into the element set, the parameter f is a user input.

2.4 New Mesh Generation

The deformed mesh generated from the earlier step is imported into Hypermesh using Tcl routines. The elements satisfying Equation 3, which were placed in the element set in the previous step, are remeshed to match the target element size provided by the user. The remeshing technique uses an existing global and a local remeshing algorithm. The global algorithm refines certain regions and improves the quality of other elements, but at times may fail. The global algorithm usually fails when lapping or element folding occurs. The local remesher is called when the global remeshing algorithm fails; which instead of remeshing the element set satisfying Equation 3, remeshes the elements that have a Jacobian less than a critical value. The rest of the mesh is refined to improve element quality. The newly created mesh is exported in Abaqus format. All of these are completely automated needs no under intervention.

2.5 Results Interpolation

A python routine handles the interpolation of results from the old mesh to the new mesh. The old deformed mesh is read into the program. A bounding box is formed from the extremities of the old mesh. To give some leeway, the bounding box is increased in size by 1%. The bounding box is now divided into small partitions. The user provides the number of equisized partitions along each axis. The partition size dx_i is calculated as

$$dx_i = \frac{x_{\max(i)} - x_{\min(i)}}{n_i} \qquad \textbf{Equation 4}$$

where n_i is the number of partitions along the i^{th} axis. The newly created mesh is read into the program, and the element centroids are computed. Due to the geometric complexity of a 3D large deformation analysis, tetrahedral elements are used to simplify the mesh generation. The centroid of a tetrahedral element, C, is calculated as

$$\vec{C} = \frac{1}{n+1}\sum_{i=0}^{n}\vec{v}_i \qquad \textbf{Equation 5}$$

where $\mathbf{v_i}$ represents the vertices and n is the number of vertices. For each new element centroid, the corresponding bounding box partition is computed. The partition elements immediately

adjacent to the current partition box are also chosen. The old elements contained within the chosen bounding box partitions are located.

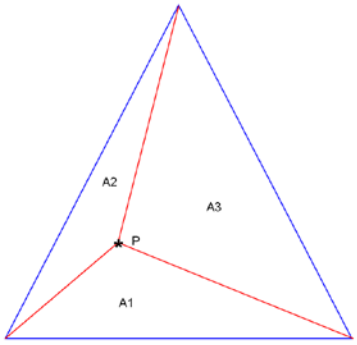


Figure 2. Point Lying Inside an Element

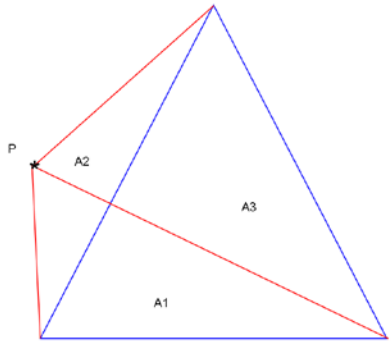


Figure 3. Point Lying Outside an Element

To located which old element contains the centroid of the new element, consider a point P shown in Figure 2 and Figure 3. If the point **P** lies inside the triangle then the sum of the area of the triangle created using the vertices and this new point is equal to the area of the original triangle. From Figure 2 it can be inferred that **A1+A2+A3=A**. If the point lies outside as shown in Figure 3 then the relationship does not hold true. The same analogy can be extended to a tetrahedron using volumes instead of areas.

Using this algorithm and the list of old elements in the partitioned boxes, the closest old element enclosing the new element centroid is computed. the nodal results from the old element are extrapolated to the new element's centroid location. The value at the centroid is computed as

$$\eta_{\text{cen}} = \sum_{i=0}^{4} \frac{V_i}{V} \eta_i \qquad \textbf{Equation 6}$$

where $\boldsymbol{\eta}$ is any field variable, $\boldsymbol{V_i}$ is the volume formed using the centroid and the 3 vertices of the old element, and $\boldsymbol{V}$ is the volume of the old element using only its nodes. This is the volume based shape function of a linear tetrahedron. Future work can expand this extrapolation to include the results at midside nodes using the shape functions of a quadratic tetrahedral element. The field variables calculated at the new element centroid are: plastic strain tensor, the stress tensor, and the equivalent strains. The same procedure is used for nodes, but only the nodal velocity is extrapolated.

2.6 Further Iterations

The initial conditions for the new mesh are created in Abaqus format and are included in the newly generated input files. These files are submitted for analysis, and this procedure continues until the requested time step is completed.

3. Validation Problem

To check the validity of the algorithm an analysis in which there is not significant element distortion was conducted. The problem was solved using both remeshing and no remeshing. Also the `*INITIAL CONDITIONS` does not allow the elastic strain component to be imported. Therefore, when the solution starts there is an initial imbalance in the new mesh due to the lack of elastic strains. The lack of elastic strains limits the current application to perfectly plastic materials. This can be circumvented by using `UMAT` routines in which elastic strain is one of the user field variables which can be extrapolated, but this work around is beyond the scope of the current effort. The model was meshed with 2nd order tetrahedral elements (`C3D10M`). The model consists of a cylinder placed between two rigid plates. The bottom plate is fixed in space and the top plate pressed down causing deformation in the cylinder. The top and bottom plates were modeled using analytical rigid surfaces.

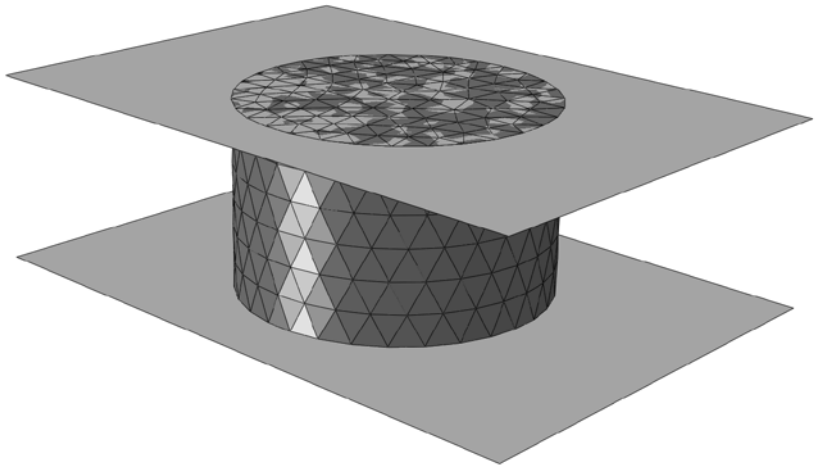


Figure 4. Cylinder Compression Problem

The punch moves down at a constant velocity of 1,000mm/s. Surfaces were defined for each component using the `*SURFACE` command in Abaqus. The `*CONTACT PAIR` option was used to define contact-as opposed to general contact-because of the ability to use the `*CONTACT CLEARANCE ASSIGNMENT` command; which is used because after remeshing some of the newly created nodes might end up penetrating the master surface. The model was analyzed for 1E-2 secs with no remeshing, and for two steps of 5E-3 with one remeshing in between the two steps. All the other parameters were the same between the models. Ths plastic strain criterion was used for remeshing purposes, and the *f* factor in Equation 3 was set to 0.6. The target element size was defined as 2.5mm. The initial coarse mesh had an average element size of 10mm. The final deformed cylinder shape from both runs are shown in Figure 5 and Figure 6. From Figure 6 it can be inferred that the new mesh is significantly finer than the initial coarse mesh. The mesh settings were deliberately set to create a finer mesh during the remeshing step.

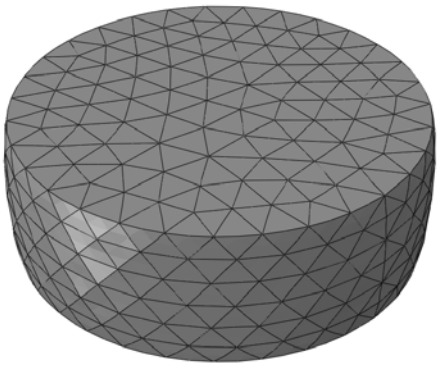


Figure 5. Deformed Shape w/o Remeshing

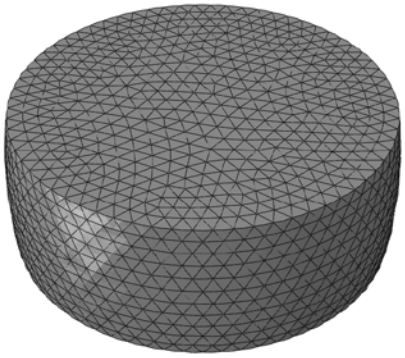


Figure 6. Deformed Shape with Remeshing

To compare the validity of the model the strain energy from both simulations was compared. The comparison is shown in Figure 7. Only the strain energy from the second step of the remeshed model is compared to the strain energy from the final 5E-3 seconds of the non-remeshed analysis. Therefore, the starting strain energies shown in Figure 7 are non-zero.

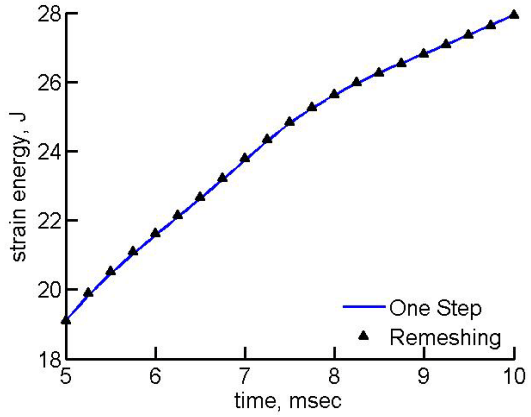


Figure 7. Comparison of Internal Energy

The results in Figure 7 show an excellent correlation between the computed strain energy of each model. The maximum difference in strain energy between the models is 0.2%.

4. Application to Metal Forming

After validation, the algorithm was used to simulate a typical metal forming application. The model consists of a deformable body known as the blank, placed on a die. A punch presses into the blank to generate deformation. All tooling components were assumed to rigid (analytical and

discrete rigid) in the model. In addition to the punch and die, rigid side walls were included to prevent sideward movement of the blank. The model set up is shown in Figure 8 and Figure 9. The blank was meshed with modified 2nd order tetrahedral elements (`C3D10M`). The model has 19734 nodes and 12389 elements. The region of the model close to the forming punch was meshed with an average element size of 1mm and the rest of the model was meshed with an average element size of 3mm.

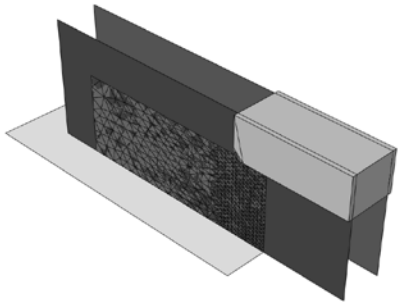


Figure 8. Model with Sidewalls

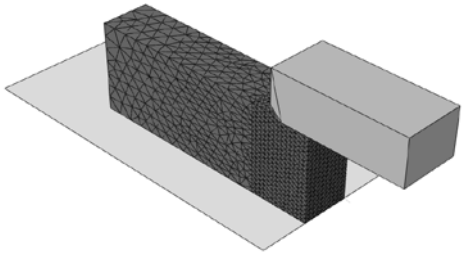
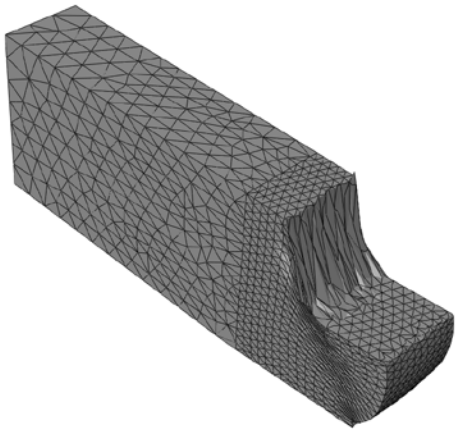
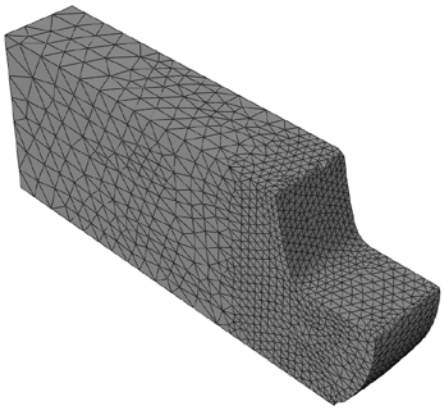


Figure 9. Model without Sidewalls

The punch moves down at a constant velocity of 500mm/s and the entire operation was simulated for 4E-3 seconds. The simulation was performed with and without remeshing. In the remeshing case the strain criterion was used with an *f* factor set to 0.4. The number of requested remeshing steps is 10. The target element size requested was 0.8mm



(a) Model with no Remeshing

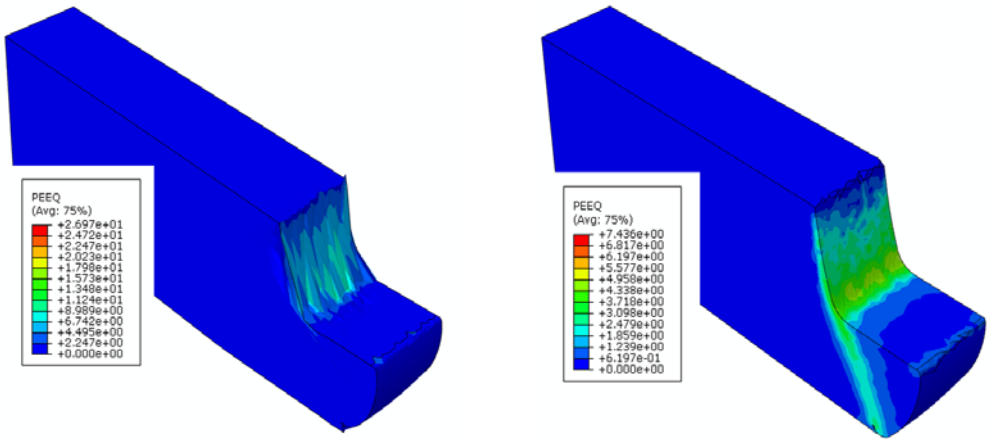


(b) Model with Remeshing

Figure 10. Deformed Blank Shape Comparison

The final deformed shapes from both runs are shown in Figure 10. Figure 10(a) is the deformed shape without remeshing and Figure 10(b) is the deformed shape with remeshing. Highly distorted

elements can be observed in the analysis without remeshing, while higher quality elements are present in the analysis with remeshing. The mesh shown in Figure 10(b) has 19181 nodes and 11579 elements. The remesher generated fine elements in regions of high equivalent plastic region. In the other regions remeshing was performed just to improve elemental quality. In a typical metal forming simulation there is an accumulation of plastic strain in the bottom of the deformed portion. This strain accumulation usually denotes a region where potential failure can occur. The equivalent plastic strain PEEQ from both the runs are shown in Figure 11.



(a) Model with no Remeshing (b) Model with Remeshing

Figure 11. Comparison of Equivalent Plastic Strains

The appearance of the strain band can be clearly seen in Figure 11(b), while the analysis without remeshing, Figure 11(a), does not show the strain band. Also the analysis without remeshing, Figure 11(a), shows spurious maximum values of PEEQ, which are not present in the remeshed analysis, Figure 11(b).

5. Summary

A 3D adaptive remeshing algorithm was developed for performing metal forming simulations using fully integrated quadratic tetrahedron elements. The remeshing technique assumes a perfectly plastic material. Validation of the technique shows good correlation, with only a 0.2% difference between the remeshed result and a known solution. The algorithm was then used to solve for typical metal forming simulation, showing a substantial improvement in strain prediction. The finite element discretization error output from Abaqus provides a metric of when remeshing to be done. However this leads to CAE remeshing the initial model and continuing with the analysis. There is still no remeshing performed in between increments. The future work would include: modifying the interpolation routine to use the shape functions of 2nd order tetrahedral elements; also using `UMAT` to include the elastic component of the strain allowing for elastic plastic materials to be analyzed; perform remeshing only when it is needed (based on the discretization

error out from `*ELEMENT OUTPUT`) and not on a fixed regular interval resulting in reduced remeshing calls

6. Acknowledgments

The authors would like to nanoPrecision Products for supporting this work. The authors would also like to thank Professor Tamer El Sayed, (King Abdullah University of Science and Technology) for his valuable contributions in developing the algorithm.

References

1. Klauss Jurgen Bathe, “Finite Element Procedures”, Prentice Hall, USA
2. Abaqus Users Manual, Version 6.9, Simulia USA
3. Hypermesh documentation, Altair Engineering, USA