

SANDIA REPORT

SAND2012-7892

Unlimited Release

Printed 09/2012

Clustered Void Growth in Ductile Metals: Final LDRD Report

Timothy D. Kostka

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.



Sandia National Laboratories



Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: <http://www.osti.gov/bridge>

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd.
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: <http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online>



SAND2012-7892
Unlimited Release
Printed 09/2012

Clustered Void Growth in Ductile Metals: Final LDRD Report

Timothy D. Kostka
Multi-physics Modeling and Simulation
Sandia National Laboratories
P.O. Box 969
Livermore, CA 94551-0969
tdkostk@sandia.gov

Abstract

This report is a compilation of projects completing during an investigation into void growth in ductile metals. Included, are the following results. (a) We have performed a suite of tests looking at the effect of porosity on the macroscopic yield stress of the material in a plane strain framework. Results have shown the orientation of voids to have a large effect on the yield stress. (b) Preliminary simulations of a periodic three dimensional void microstructure are given along with the meshing procedure. Results show less void interaction as compared to the two dimensional case. (c) Development and implementation of an anisotropic plasticity model is detailed. The model is used to replicate anisotropic necking seen in a tensile bar experimentally tested to failure. (d) We have shown efficiency gains of 84% of a 2D solution framework compared the standard 3D framework. (e) We investigated a large number of element formulations and have shown the q1p0 element (selectively integrated hex8) to outperform all others in the context of large deformation plasticity simulations. (f) The implementation of the q1p0 element into SIERRA is provided along with results of verification and performance investigations.

Contents

Contents	5
List of Figures	8
List of Tables	11
Introduction	15
Background	15
Archival	16
Effect of porosity on macroscopic yield stress—a two-dimensional study	17
Meshing Procedure	17
Overview	17
Details	19
Analysis procedure	20
Results	22
Effect of porosity on yield stress	22
Effect of void uniformity on yield stress	23
Effect of void aspect ratios on yield stress	24
Effect of void aspect ratios on yield stress	25
Results of two-dimensional bifurcation simulations	27
Procedure	27
Meshing	27
Results	28
Additional work	30
Results of three-dimensional void growth simulations	31
Procedure	31
Initial conditions	31
Results	31
Differences from two-dimensional results	32
Future work	32
Three-dimensional mesh generation	35
Void distribution in Mathematica	35
Microstructure parameters	36
Estimate of computational costs	36
Choosing the best unit cell	38
Problems at corners	38

Problems at edges.....	38
Problems at faces	39
Meshing procedure.....	40
Problems encountered while meshing	45
Cubit IDs change.....	45
Copy mesh produces unreliable results.....	46
Importing more than one surface at once problematic.....	46
Anisotropic plasticity model development	47
Procedure	47
Formulation.....	47
Elastic response.....	47
Infinitesimal elastic response	48
Yield surface	48
Parameter fitting.....	49
Inverse stress calculation	52
Implementation	53
Preliminary results	53
Mathematical enforcement of boundary conditions	55
Enforcement of periodic boundary condition	55
Simple linear constraint example.....	56
Efficiency of a plane strain formulation	57
Procedure	57
Planar plane strain.....	57
Meshing.....	58
Element formulation	59
Platform.....	59
Metrics	60
Results.....	60
Timing profile	61
Sample input deck.....	62
Best elements for large deformation plasticity	65
Background	66
Mathematical foundation	66
Standard displacement formulation	67
Theoretical foundation	67
Selective integration formulation.....	67
Selective deviatoric formulation.....	68
A note on selective techniques.....	69
Modeling procedure	69

Notched tensile specimen.....	70
Material models.....	70
Boundary conditions	71
Mesh refinement levels	71
Element formulations	73
Metrics of interest	74
Visualizing element fields.....	75
Results.....	77
Fields after infinitesimal loading	77
Fields after full loading.....	78
Selective formulations of the HEX20 element	81
Typical timing profile	82
Load at displacement error.....	83
Triaxiality field	84
Additional work	85
Q1P0 element implementation into SIERRA	87
Formulation.....	87
Usage syntax	87
Output.....	87
Additional options.....	88
Verification tests	89
Cantilever beam—Implicit.....	90
Constrained block compression—Implicit.....	92
Constrained block compression—Explicit.....	93
Distribution	94

List of Figures

Figure 1: At left, the typical microstructure of stainless steel 304L before a metal forming operation. Both voids (white) and inclusions (red) are present. At right, under loading, nearby voids grow and link up to form the beginning of the fracture surface.....	15
Figure 2: Fracture surface of stainless steel 304L. Inclusions can be seen at the center of the large voids which have coalesced.	16
Figure 3: Example meshes for the plane strain void geometry for a porosity of 10% (left) and 1% (right). Both meshes have voids with an aspect ratio of 2.	18
Figure 4: A mesh of a randomly generated microstructure in which the size, orientation and aspect ratio of voids were taken from a probability distribution function.	18
Figure 5: Normalized yield stress as a function of porosity for randomly distributed initially circular voids.	22
Figure 6: Normalized yield stress as a function of porosity for randomly distributed initially circular voids near the region of zero porosity. (This is a close-up of the previous figure.).....	23
Figure 7: The probability density function of the void area for this study.	23
Figure 8: This shows the effect of nonuniform void areas versus uniform void areas on the normalized yield stress.....	24
Figure 9: Normalized yield stress as a function of porosity for a variety of void aspect ratios.....	24
Figure 10: For a void aspect ratio of 4, the effect of orientation of the major axis with the uniaxial tension loading direction.	25
Figure 11: The meshing procedure started with a periodic array of voids (left), then placement of nodes on the exterior (middle), then generation of the final mesh using the paver algorithm (right).	27
Figure 12: Representative load versus displacement curve for a two dimensional array of voids. The onset of necking—or bifurcation point—is noted at the point of maximal load.....	28
Figure 13: The beginning of a fracture surface can be seen. Note the voids along the zone of high plastic strain have increased significantly in size compared to voids in other regions.....	28
Figure 14: Load curves up to the point of bifurcation are shown for each of the 32 microstructures.....	29
Figure 15: A histogram of the strain at which each microstructure sample necked is shown along with a best fit normal distribution curve.....	29
Figure 16: Equivalent plastic strain in one slice of the void microstructure.	32

Figure 17: A problematic unit cell choice because the corner is within a void.	38
Figure 18: A problematic unit cell choice because an edge is very close to a void surface.	39
Figure 19: A problematic unit cell choice because a face is very close to a void surface extent.	39
Figure 20: The generated geometry after subtracting the voids from the unit cell.	41
Figure 21: The placement of nodes along each curve.	42
Figure 22: Meshes of the six external faces of the unit cell. Note the interior and void surfaces are unmeshed at this point.	43
Figure 23: The final and truly periodic mesh in all its glory.	45
Figure 24: The yield surface as depicted as an ellipse in two dimensions. The return mapping algorithm finds σ_n as a function of σ_y and the trial σ	50
Figure 25: The tensile specimen tested at 300 °C. At right is a view of the asymmetric fracture surface.	53
Figure 26: Equivalent plastic strain in a tensile specimen with anisotropic properties.	54
Figure 27: The initial configuration (left) and one possible deformed configuration (right) of four material points.	55
Figure 28: The mesh used in the analysis (left) and a close-up of the mesh around a single void (right).	58
Figure 29: The equivalent plastic strain field.	60
Figure 30: The proposed selectively integrated HEX8 element formulation.	65
Figure 31: The proposed selectively integrated TET10 element formulation.	65
Figure 32: Geometry of the cylindrical notched tensile specimen. The specimen is axisymmetric about the left edge and symmetric across the bottom edge.	70
Figure 33: Yield stress curve fits to experimental stainless steel 304L data.	71
Figure 34: Successive refinements of the HEX8 notched tensile mesh. Refinement levels are “x2” (left), “x4” (middle), and “x8” (right). The “x16” and “x32” refinements are not pictured.	72
Figure 35: Successive refinements of the TET notched tensile mesh. Refinement levels are “x2” (left), “x4” (middle), and “x8” (right). The “x16” and “x32” refinements are not pictured.	73
Figure 36: Construction of a pointwise defined field (right) from discrete values at cubature points (left). The value at each cubature point is shown as a sphere on the left-hand picture.	76
Figure 37: A per-element construction of a pointwise defined field (right) from discrete values at cubature points (left). The construction was done through a least squares fit to a linear polynomial basis.	76

Figure 38: The normalized equivalent stress (left), normalized pressure (center), and triaxiality (right) fields for an infinitesimal loading of the notched geometry. The maximum triaxiality of approximately 0.94 occurs at the center of the geometry.	77
Figure 39: The triaxiality field under an infinitesimal loading for the HEX20 formulation under successive refinements, from “x2” (left) or “x32” (right).	77
Figure 40: The triaxiality field under an infinitesimal loading for the TET4 formulation under successive refinements, from “x2” (left) or “x16” (right).	78
Figure 41: The equivalent plastic strain (left), normalized pressure (center), and triaxiality (right) fields after a loading of 0.2 inches of the notched geometry for the HEX8SI element formulation.	78
Figure 42: The evolution of the triaxiality field from the beginning of plastic deformation (left) to the end of the simulation (right).	79
Figure 43: The pressure field for a variety of standard displacement formulations. Note that there is considerable clipping in the fields in all cases, especially for the HEX8 and TET4 cases.	79
Figure 44: The pressure field for three selective element formulations.	80
Figure 45: The triaxiality field obtained through averaging the integration point values in the HEX8 formulation (left), the HEX8SD formulation (middle), and the HEX8SI formulation (right).	81
Figure 46: Values of the triaxiality at bulk cubature points for two HEX20SI formulations. Note the formation of ridges along the notch surface.	82
Figure 47: Error in load versus element edge length. Light gray contour lines correspond to the expected linear convergence behavior.	83
Figure 48: Error in load versus simulation walltime.	84
Figure 49: Error in maximum triaxiality versus element edge length. Light gray contour lines correspond to linear convergence behavior.	84
Figure 50: Error in maximum triaxiality versus walltime.	85
Figure 51: The equivalent plastic strain field in the cantilever beam problem is shown for the x8 mesh.	90
Figure 52: The constrained block is shown being progressively deformed.	93

List of Tables

Table 1: Function to return the aspect ratio.	19
Table 2: Function to return the relative void area.	19
Table 3: Sample Ares input deck for running a 2D void simulation.	22
Table 4: Microstructure and meshing variables used in this analysis.	36
Table 5: Comparison of the estimated number of elements to the actual number for a given mesh for several sets of microstructure parameters.	37
Table 6: An example Cubit journal file to create and save the void geometry.	40
Table 7: An example Cubit journal file to mesh the boundary curves and faces.	43
Table 8: An example Cubit journal file to imprint the external faces and mesh the interior.	44
Table 9: Algorithm to find elastic stretch as a function of unrotated stress.	53
Table 10: Element interpolation and cubature types used in each framework.	59
Table 11: Statistics for each framework.	59
Table 12: Results for each framework.	60
Table 13: Timing profile for the “Standard 3D” framework.	61
Table 14: Timing profile for the “Reduced 3D” framework.	61
Table 15: Timing profile for the “True 2D” framework.	62
Table 16: Ares input deck for the “True 2D” framework.	63
Table 17: Material parameters for stainless steel 304L.	71
Table 18: Number of nodes and elements in each mesh.	73
Table 19: Summary of the interpolation schemes used.	74
Table 20: Summary of the cubature schemes used.	74
Table 21: Timing profile information for the TET10 x4 refinement run. Note that the vast majority of the time is spent within the linear solver.	82
Table 22: The resultant force is shown for each mesh and each element formulation for the implicit cantilever beam problem.	91
Table 23: The change in resultant force is shown for each successive mesh refinement and each element formulation for the implicit cantilever beam problem.	91
Table 24: The percentage error is shown for each mesh and each element formulation for the implicit cantilever beam problem.	92
Table 25: The total cpu-hours are shown for each mesh and each element formulation for the implicit cantilever beam problem.	92

Table 26: The scaled load at displacement is shown for each mesh and each element formulation for the implicit constrained block compression problem.....	93
Table 27: The scaled load at displacement is shown for each mesh and each element formulation for the implicit constrained block compression problem.....	93

Introduction

This report is a collection of documentation—some previously disseminated—that was done as part of an Early Career LDRD on clustered void growth in ductile metals.

Background

Fracture in ductile metals occurs through the processes of void formation, growth and coalescence. Ultimately, voids grow large enough to join together and fracture occurs.

An initial material microstructure before the forming operation is shown in Figure 1 (left). Within the matrix material, both second phase particles and voids are typically present. As the material is deformed by the forming process or by stresses and loads applied while the component is in use, nearby voids can grow and link up to form the beginning of the fracture surface (right). During this time, new voids can also be nucleated at inclusions.

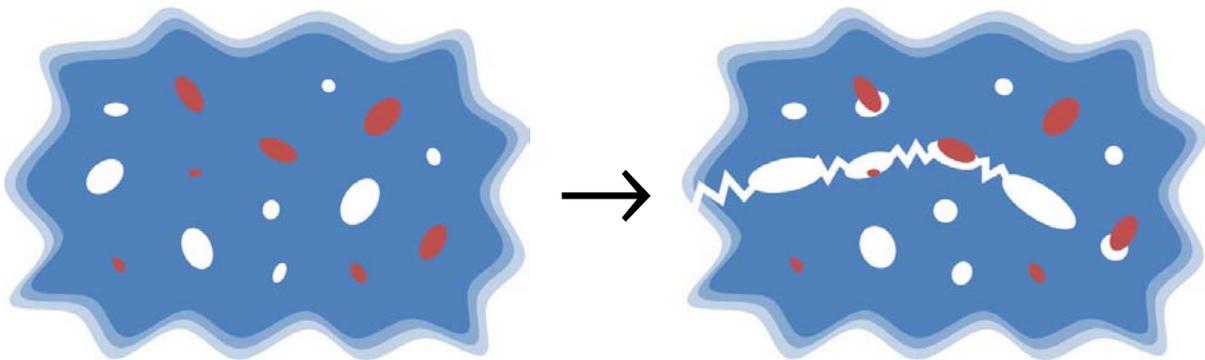


Figure 1: At left, the typical microstructure of stainless steel 304L before a metal forming operation. Both voids (white) and inclusions (red) are present. At right, under loading, nearby voids grow and link up to form the beginning of the fracture surface.

A typical fracture surface of stainless steel 304L is shown in Figure 2. At the magnification shown, this picture resembles a cratered surface. Each crater forms half of what was previously a void in the material which was separated when the fracture surfaces were formed.

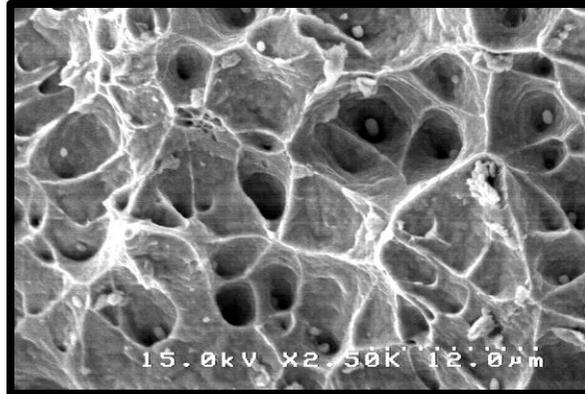


Figure 2: Fracture surface of stainless steel 304L. Inclusions can be seen at the center of the large voids which have coalesced.

In order to assess the safety and reliability of components made from stainless steel, it is critical that we understand this failure process.

Archival

In addition to being summarized within this document, all work on the project has been archived under the WorkBench project `clustered_void_growth`.

Effect of porosity on macroscopic yield stress—a two-dimensional study

Within our current computational tools, all damage models have implemented an isotropic void growth rule. These models assumed that damage affects the yield stress in a linear fashion and that the response is isotropic.

In this chapter, we examined the effect of porosity and anisotropic effects on the macroscopic response of the material to investigate the accuracy of these assumptions.

Meshing Procedure

The procedure for generating meshes will first be outlined with details given in later sections. All related files are archived at the following Workbench location:

- `workbench:/clustered_void_growth/2d_porosity_study/meshing/`

Scripts and source code files within this directory are generously commented.

Overview

In the meshing process, the microstructure is generated via the following procedure.

1. Create a “trial” void placed randomly within the unit cell
2. If void is sufficiently far away from all other voids, add it to the microstructure. If not, create a new trial void and repeat
3. Find the best “unit cell” for the given microstructure.
 - a. First, find a point which is far away from all voids and translate all voids so this point is at (0, 0).
 - b. Find a path from (0, 0) to (0, 1) keeping far enough away from each void surface.
 - c. Find a path from (0, 0) to (1, 0) keeping far enough away from each void surface.
4. Create the geometry in Cubit and generate a preliminary mesh.
5. Within ParaView, evaluate the element size function at each node based on the distance away from the nearest void surface.
6. Reload the geometry within Cubit and create the final mesh using the generated sizing function.

Two examples meshes can be seen in Figure 3 for a distribution of 64 voids—each with an aspect ratio of 2 and aligned in the same direction—for a porosity of 10% (left) and 1% (right).

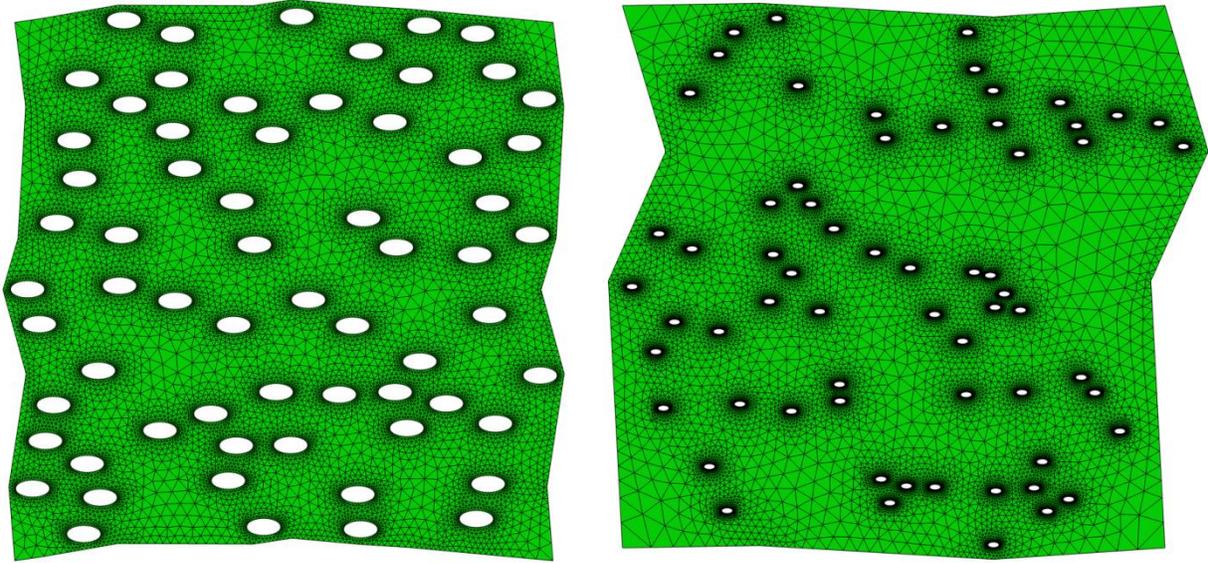


Figure 3: Example meshes for the plane strain void geometry for a porosity of 10% (left) and 1% (right). Both meshes have voids with an aspect ratio of 2.

It should be noted that the tool used to generate the microstructure and subsequent mesh is able to generate considerably more complex microstructures than those depicted in the previous figure. For example, Figure 4 shows a mesh which was generated using probability distribution functions for the void size, orientation and aspect ratio.

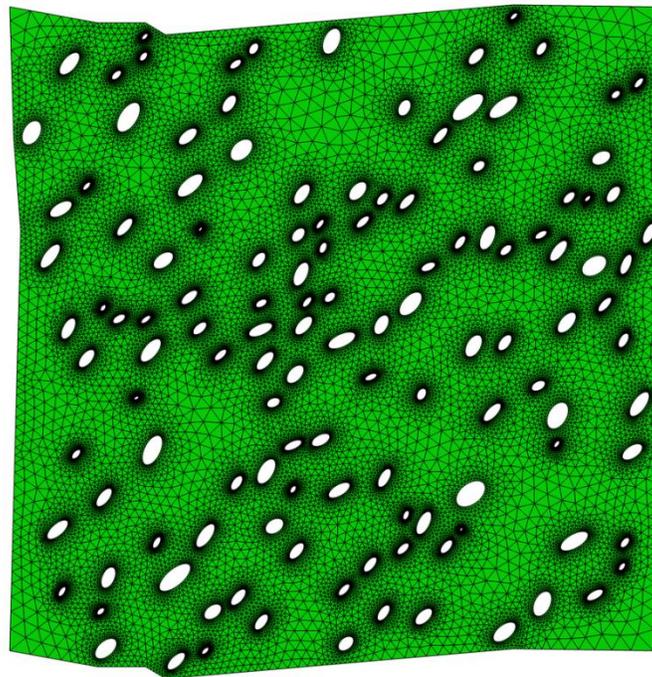


Figure 4: A mesh of a randomly generated microstructure in which the size, orientation and aspect ratio of voids were taken from a probability distribution function.

Details

Details for each of the steps are given below.

1. Create a “trial” void placed randomly within the unit cell

A trial void consists of a void area, location, major and minor radii, and orientation angle within the plane. Each of these variables can be taken from a distribution function. In the code, a uniformly distributed number between 0–1 is generated and passed to a routine to calculate the, for example, aspect ratio.

The routine to return the aspect ratio is shown in Table 1 and generates the aspect ratio according to a normal distribution with a mean 2 and a standard deviation of 0.3. This code can be changed to return any other distribution (no necessarily normal) and recompiled.

```
// return the aspect ratio of the phi percentile
double getAspectRatio(const double phi)
{
    return randNormal(2., 0.3);
}
```

Table 1: Function to return the aspect ratio.

The code to return the relative void area according to a Weibull distribution is shown in Table 2.

```
// return the unscaled void area of the phi percentile
double getArea(const double phi)
{
    // get the shape, scale parameters for the distribution
    static const std::pair<double, double> weibullParameter =
        getWeibullParameters(1., 0.1);
    // return a random point
    return randWeibull(weibullParameter.first, weibullParameter.second);
}
```

Table 2: Function to return the relative void area.

Routines for these and other parameters of the void distribution and documentation on their use can be found in the source files.

2. If void is sufficiently far away from all other voids, add it to the microstructure. If not, create a new trial void and repeat.

A void is sufficiently far away from all other voids if it is separated by at least a given distance. This distance is calculated based on the distribution functions given and can be modified.

If the trial void cannot be successfully placed, a new trial void is created by simply translating the old one into a new position. This was done to avoid the scenario where smaller voids had a relatively larger probability of being successfully placed which could alter the distribution.

3. Find the best “unit cell” for the given microstructure.

- a. First, find a point which is far away from all voids and translate all voids so this point is at (0, 0).
- b. Find a path from (0, 0) to (0, 1) keeping far enough away from each void surface.
- c. Find a path from (0, 0) to (1, 0) keeping far enough away from each void surface.

In this step, we wish to find a tessellation of the space such that the boundaries of the tessellation are as far away from a void as possible. This is in generally a computationally intensive process to perform for an arbitrary array of voids.

First, a single point is obtained through sampling which is the farthest away from a void. This point is chosen to be (0, 0).

We then attempt to find a tessellation with boundaries separated from voids by a distance of at least δ .

A uniform grid is used with points spaced in increments of $\frac{1}{2}\delta$. At each grid point, the distance to the nearest void is calculated. Then, a modified version of Dijkstra's algorithm is employed to find the shortest distance from point (0, 0) to point (1, 0) using grid points which are at least δ away from voids. If this algorithm fails, we reduce δ and try again.

After boundaries have been established, a minor smoothing algorithm is performed to improve mesh quality.

4. Create the geometry in Cubit and generate a preliminary mesh.

A preliminary mesh is created using Cubit's automated `tetmesh` algorithm. This preliminary mesh has widely varying element sizes and is not suitable for computation.

5. Within ParaView, evaluate the element size function at each node based on the distance away from the nearest void surface.

At each node, the distance to the nearest void surface is calculated and stored into a node-based field within the `ExodusII` file. This is performed using the `Calculator` filter within ParaView.

6. Reload the geometry within Cubit and create the final mesh using the generated sizing function.

The mesh and node-based field is read into Cubit. The `paver` algorithm is used with a skeleton sizing field obtained from the field. The resulting mesh has smoothly varying element sizes with smaller elements at the void surface and larger elements in places far from voids. This mesh is suitable for efficient computation.

Analysis procedure

A sample run can be found in the following Workbench directory

- `workbench:clustered_void_growth/2d_porosity_study/sample_run/`

The material model used in these runs is representative of a structural metal such as an aluminum alloy. There is a small amount of strain hardening present. In addition, a strain rate dependence was added to provide the analysis some numerical stability.

A sample Ares input deck is given in Table 3.

```
# set up the 2D plane strain framework
>framework
plane_strain_planar

# define the material model
>material
```

```

material, plastic_isotropic, 160e9, 78e9, 8e3, hardening_curve,
      strain_rate_dependence_curve
>data curve
hardening_curve, piecewise_linear
0, 200e6
1, 220e6
>data curve
strain_rate_dependence_curve, piecewise_linear
0, 1
1, 1.1

# include the mesh file
>include
"mesh_voids_planar.in"

# set termination time
>termination time
0.0034641

# set the initial timestep size
# (initial size) = (termination time) / (timesteps)
>timesteps
10

# adjust the timestep so that the max eqps change
# in any element per timestep is at most X
>control timestep via material state change
bulk, eqps, 0.03

# set the element formulation
>element block cubature
bulk, tri7p5o

# enforce periodic boundaries to unit cell
>bc periodic translational direction
x, 1
y, 1

# plane strain condition
>assign surface id analytic
bottom, "Y < 0.5"
left, "X < 0.5"

# displacement boundary conditions
>assign node id
-1, 0, 0, 0
-2, 1, 0, 0
-3, 0, 1, 0
-4, 1, 1, 0
>bc prescribe node displacement
-1, zero, x
-1, zero, y

# x displacement of top
>analytic curve
u, "t"
>bc prescribe node displacement
-4, u, x
-4, zero, y
-2, zero, y
-2, zero, x

# output surface forces
>output surface force
bottom, y, "load_yy.txt"
left, x, "load_xx.txt"
bottom, x, "load_yx.txt"
left, y, "load_xy.txt"

```

```

# output material maximuma
>output material state value
bulk, eqps, maximum, "eqps_max.txt"
bulk, eqpsdot, maximum, "eqpsdot_max.txt"

# output expansion of block
>output element block value
bulk, "v/V-1", "expansion.txt"
bulk, "v", "dense_area.txt"

```

Table 3: Sample Ares input deck for running a 2D void simulation.

Results

Compiled results can be found in the following Workbench directory

- `workbench:clustered_void_growth/2d_porosity_study/results/`

Effect of porosity on yield stress

In one set of simulations, we generated microstructures with uniform circular voids of varying porosities between 0.001 and about 0.2. After deforming the body, the macroscopic yield stress was postprocessed and compared to the yield stress of the unvoided material.

At each porosity, around 50 different microstructures were generated. The error bars in the plot refer to two standard deviations from the mean.

The overall effect of porosity on yield strength for an initially isotropic microstructure is shown in Figure 5. This trend was empirically fit to an curve which is given in the figure.

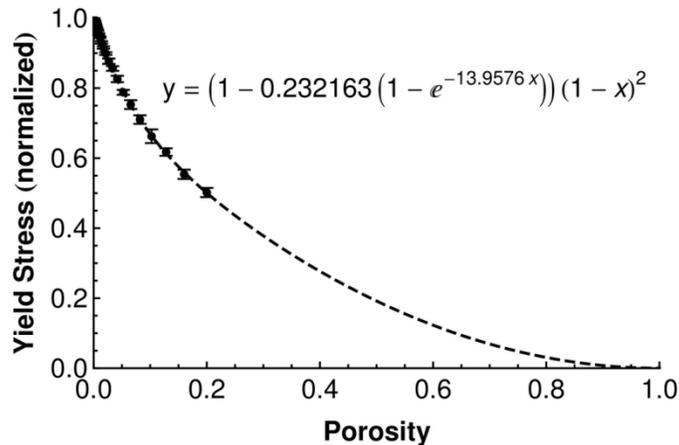


Figure 5: Normalized yield stress as a function of porosity for randomly distributed initially circular voids.

An inset of the previous figure around the neighborhood of zero porosity is shown in Figure 6. From the curve fit in this figure, we see that a 0.01 increase in porosity causes a 0.05 drop in yield stress. In other words, the effect of porosity on yield stress for low porosities is closer to

$$\sigma_y(\phi) \approx (1 - 5\phi)\sigma_y^0 \quad (0.1)$$

instead of the commonly assumed $(1 - \phi)$ dependence.

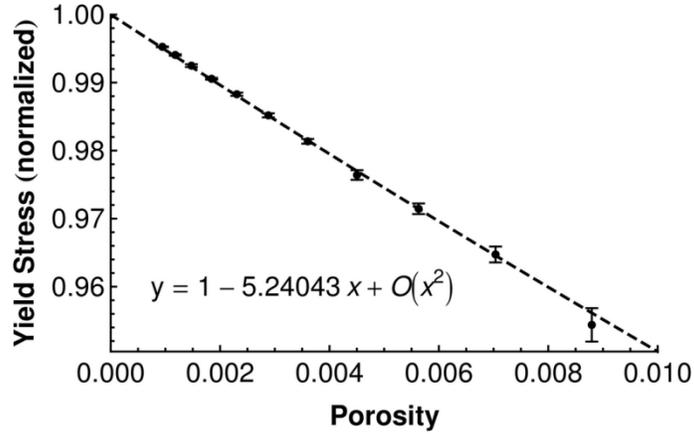


Figure 6: Normalized yield stress as a function of porosity for randomly distributed initially circular voids near the region of zero porosity. (This is a close-up of the previous figure.)

Effect of void uniformity on yield stress

In the previous results, we assumed all voids had the same radius. In these results, we have repeated the previous study but generated new microstructures and meshes with a distribution of void areas. These areas were given by a Weibull distribution with parameters $k = 1.72$ and $\lambda = 1.12$ which were chosen such that $\sigma / \mu = 0.6$ and is shown in Figure 7.

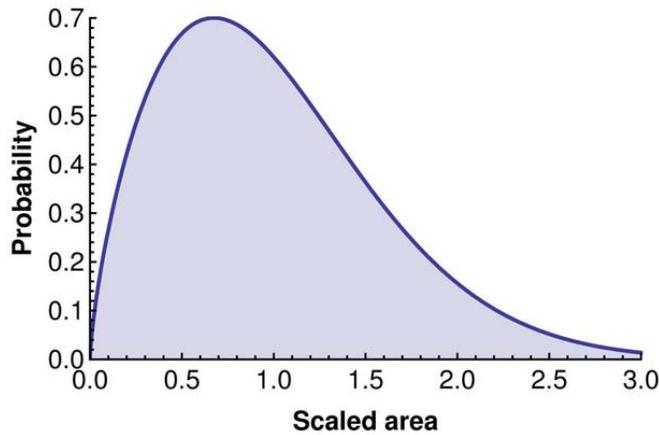


Figure 7: The probability density function of the void area for this study.

The results are shown in Figure 8 for the effect of uniform voids and voids with varying areas. As the figure clearly shows, the effect for all porosities is minimal. The mean of each curve lies well within the error bars of the other curve. This was a surprising result, but it does suggest that it may not be necessary to track the relative void sizes of the microstructure within an anisotropic void growth model.

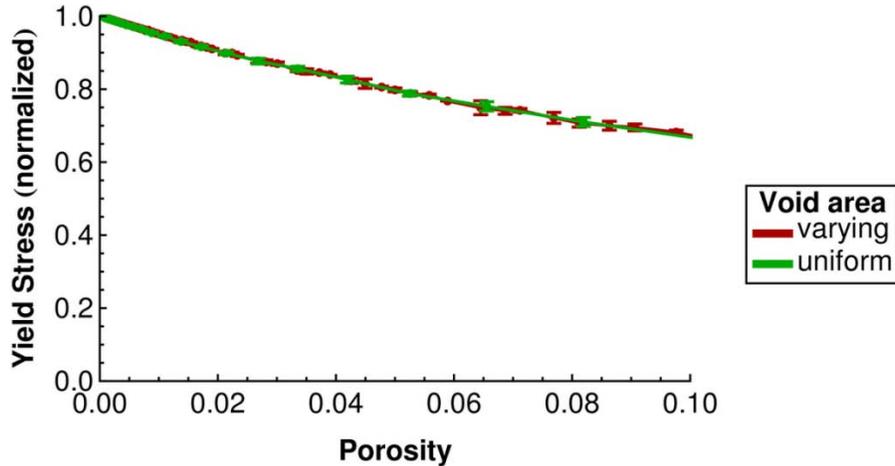


Figure 8: This shows the effect of nonuniform void areas versus uniform void areas on the normalized yield stress.

Additional void area distributions were tested and none were found to have an appreciable effect on the yield stress versus porosity curve.

Effect of void aspect ratios on yield stress

In this set of runs, the void aspect ratio was varied. Voids were aligned with either their major or minor axis pointing towards the loading direction. An aspect ratio of 4 indicates voids with the minor axis aligned perpendicular to the loading direction while an aspect ratio of $\frac{1}{4}$ indicates the same void shape but with the major axis in the loading direction.

The results are shown in Figure 9. From this figure, it is clear that void orientation plays a large role in the macroscopic response of the material. For a given porosity, voids aligned with the loading direction have a relatively minor effect on the yield stress. Voids oriented perpendicular to this direction have a much larger effect. The nominal case of initially circular voids produces results in between these two extremes.

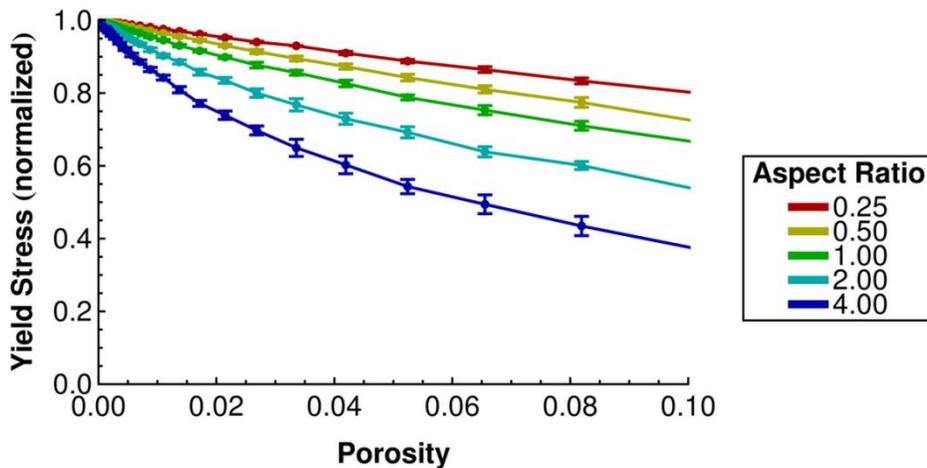


Figure 9: Normalized yield stress as a function of porosity for a variety of void aspect ratios.

Effect of void aspect ratios on yield stress

In this set of runs, the aspect ratio of the voids was fixed at 4. The orientation with respect to the loading direction was varied from 0° (aligned), to 90° (perpendicular). The results are shown in Figure 10.

The results from this experiment are similar to those of the previous one. Voids aligned with the loading direction have less of an effect on the yield stress. However, we also see the effect at angles between these extremes. The difference between 0° and 30° is much larger than that between 90° and 60° .

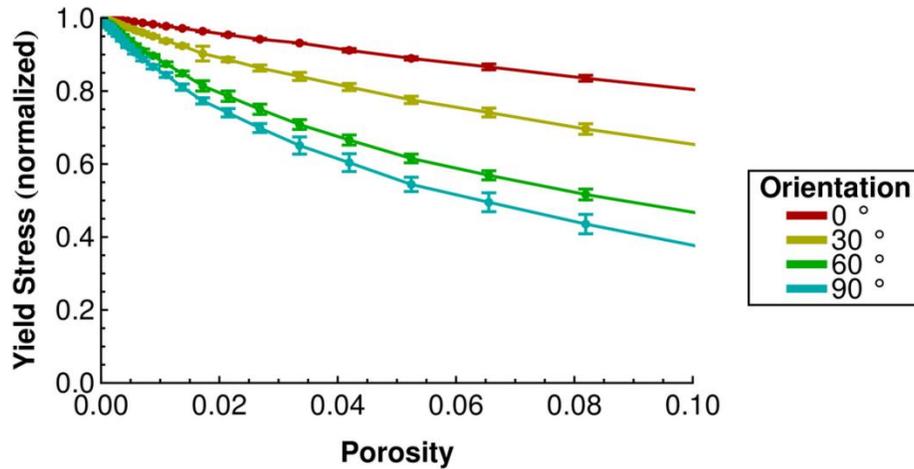


Figure 10: For a void aspect ratio of 4, the effect of orientation of the major axis with the uniaxial tension loading direction.

Results of two-dimensional bifurcation simulations

This chapter includes the results of a study which looked at the onset of coalescence in a randomized microstructure.

Procedure

Starting from a randomized microstructure of initially circular voids, the body was loaded under uniaxial tension in a plane strain framework. This loading condition induces an averaged triaxiality of 0.5 which varies locally due to the presence of voids.

We used a basic power law hardening material model representative of stainless steel 304L.

For each microstructure, the load and displacement were tracked throughout the simulation. The point of maximal load was noted as the onset of coalescence. This point is annotated in Figure 12. This point corresponds to the onset of necking in a tensile bar.

A set of 32 randomly generated microstructures were generated. This represents the natural variation which would occur in a material.

Meshing

For this analysis, the procedure for generating meshes was a primitive version of the version outlined in a previous chapter. Because the new version is superior, this version will not be heavily documented.

The basic meshing procedure was to (a) generate a random microstructure, (b) choose a unit cell, (c) place nodes everywhere on the boundary, and (d) mesh the interior. These stages are shown in Figure 11.

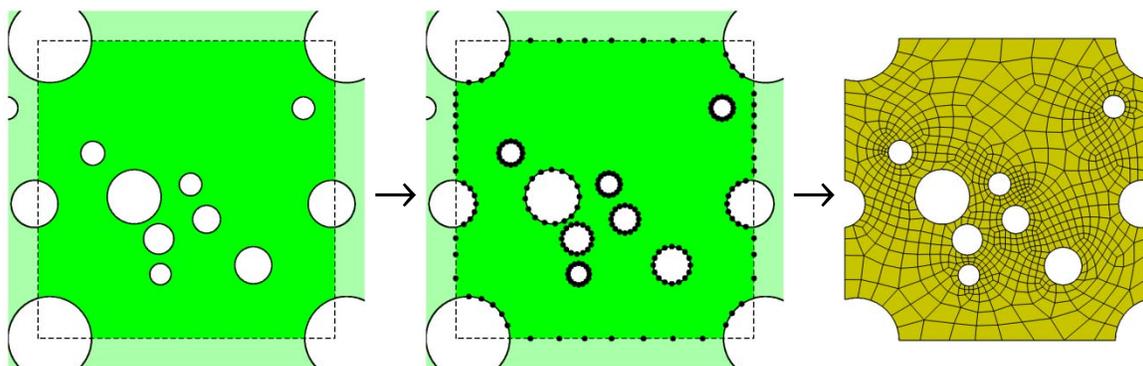


Figure 11: The meshing procedure started with a periodic array of voids (left), then placement of nodes on the exterior (middle), then generation of the final mesh using the paver algorithm (right).

Although this meshing procedure worked well enough for this analysis, it had several drawbacks keeping it from being more applicable in general. With void volume fractions under about 1%,

the meshing procedure often failed to work correctly or produced elements too distorted to be reliable. Additionally, the having voids of different sizes often caused the meshing to fail. These drawbacks were improved in the updated meshing procedure.

Results

A representative load versus displacement curve is shown in Figure 12 with the bifurcation point annotated. This point represents the point of material instability in which the deformation begins to localize.

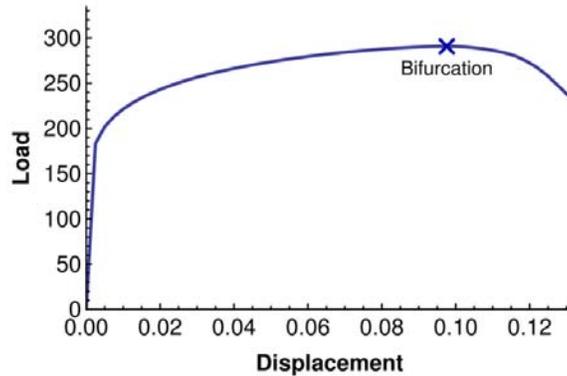


Figure 12: Representative load versus displacement curve for a two dimensional array of voids. The onset of necking—or bifurcation point—is noted at the point of maximal load.

In Figure 13, the equivalent plastic strain is shown at a point in the simulation shortly after bifurcation. The beginning of the fracture surface is clearly evident along the region of increased plastic strain. Further loading of the material will result in nearly all deformation taking place in this zone. Although each void started out identical in size, the voids along the fracture surface have grown significantly in size while others have grown only a small amount.

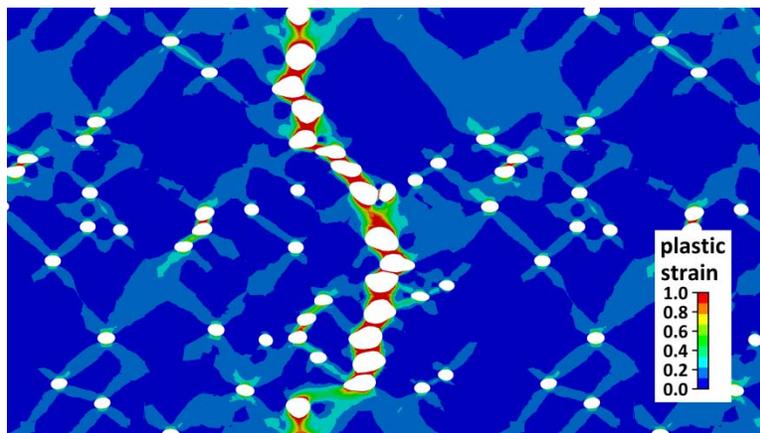


Figure 13: The beginning of a fracture surface can be seen. Note the voids along the zone of high plastic strain have increased significantly in size compared to voids in other regions.

By plotting each load versus displacement curve up until the point on fracture, we can see the spread in the data. This is shown in Figure 14 with an “X” marking the bifurcation point. Data past this point is not plotted for clarity. Even though the microstructures were all seeded with the same randomized generation algorithm, there is a significant spread in the displacement at which bifurcation takes places.

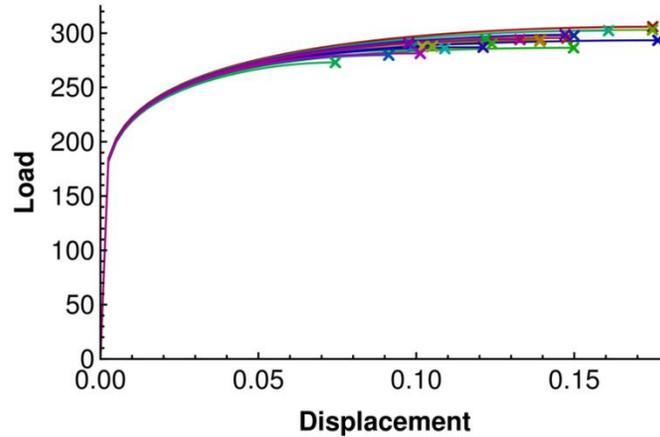


Figure 14: Load curves up to the point of bifurcation are shown for each of the 32 microstructures.

In order to assess material integrity at the macro scale, we have homogenized these results in the following manner. We have assumed a normal distribution for the strain at bifurcation. When the material is strained past this point, even though it may not be fractured, the material has lost its structural integrity. In this sense, we regard the point of bifurcation (or necking) as the point of failure. These points were plotted and a best fit normal distribution was used to approximate the data.

We can see this procedure in Figure 15. A histogram of the necking strains is shown in green, while the best fit normal distribution curve is shown as a black line. In this example, the strain at failure for a 95% confidence level is shown in red. For a material which has been strained more than this level, we can regard that material as failed.

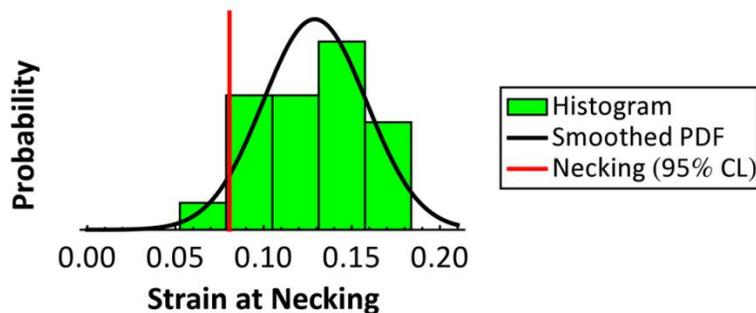


Figure 15: A histogram of the strain at which each microstructure sample necked is shown along with a best fit normal distribution curve.

Additional work

The results shown in this chapter are applicable to a specific material and microstructure. Because each material is unique, these results are not applicable to ductile metals in a general sense.

Although the results given were using a material model representative of stainless steel 304L, the initial void volume fraction of 3% is considerably higher than the $< 0.1\%$ found in the actual material. This discrepancy was a result of technical hurdles which could not be overcome in the time allotted. Along with the meshing difficulties discussed earlier, the time required for each simulation was significant. Reducing the void volume fraction by a factor of 10 would increase the time required by about a factor of 10. Instead of each simulation taking a few days, they might take a month or more to finish. Just due purely to the instability of the computational platforms, this approach would not have been feasible.

Results of three-dimensional void growth simulations

Similar to the previous study of voids in a two dimensional plane strain framework, we ran a series of simulations on a fully three dimensional microstructure.

Procedure

Using the meshing procedure described in a later chapter, a set of 64 meshes was created to represent a random distribution of initial void microstructures. All voids were initially spherical in shape and had a uniform diameter.

To efficiently capture the large deformation response, a selectively integrated TET10 element was used. This element was shown to be the most efficient in the element study detailed in a later chapter.

The periodic boundary conditions were enforced in a manner identical to the 2D plane strain simulations. Opposite faces were constrained to have the same shape. Since the choice of the boundary of the unit cell is arbitrary, no artificial constraints were placed to keep faces planar. This allows for the most general periodic solution.

Loading was applied to represent a uniaxial tension condition. This results in an effective triaxiality of 0.33. Since this is less than the triaxiality of 0.5 seen in the plane strain simulations, one would expect voids to grow slightly slower.

Initial conditions

In an ideal world, we would model the initial conditions of the microstructure—including void distribution, shape and orientation—to closely resemble that of the actual material. While some techniques are available to determine some characteristics of the initial microstructure, these techniques are not yet developed enough to paint a detailed three dimensional picture. We therefore rely on assumptions to seed the initial microstructure.

In this analysis, our assumptions of initially spherical voids distributed randomly was done both for computational advantages and to model an isotropic material.

Results

Some typical results can be seen in Figure 16 where the equivalent plastic strain in the microstructure is plotted at a macroscopic plastic strain of 10%. The body is being loading in the vertical direction with a loading condition representative of uniaxial tension. Interaction effects between neighboring voids can be seen, especially with the two voids on the left-hand axis linked by a band of increased plastic strain. Areas of low plastic strain can also be seen on the top and bottom of each void.

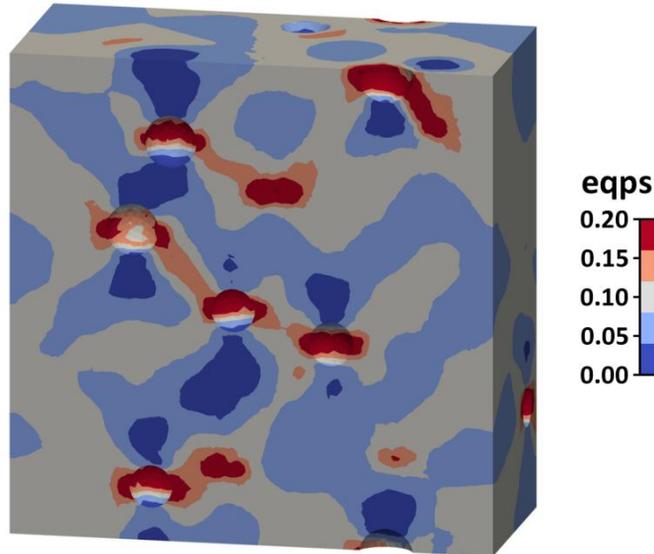


Figure 16: Equivalent plastic strain in one slice of the void microstructure.

While most simulations were able to be run to 20% macroscopic strain, we ran into computational limits preventing us from going much further. The walltime of 4 days on a single processor, coupled with no mechanism for simulations restarts, limited the run.

Differences from two-dimensional results

In these simulations, we did not see noticeable buckling behavior as we did in the two-dimensional runs. This is likely on account of several issues. The loading condition here had a triaxiality of 0.33 compared to 0.5 in the 2D runs. Because of this, we expect voids to grow and link up more slowly. The other issue is likely on account of the natural differences between 2D and 3D simulations. In three dimensions, the link between voids is not nearly as strong even if they are separated by the same distance. This causes less of an interaction and the mechanism for enabling coalescence is not as strong.

Future work

There is some question as to whether or not the selectively integrated TET10 element is behaving well in this simulation. Although the element performed well in the notched tensile study, it may be producing an overly compliant response in this simulation. In some simulations, deformation at a particular cubature point would grow much more than expected. Rather than attributing this as a result of the deformation, we believe it may be pointing to a deficiency in the element. This remains an open question.

Apart from the element issue, the mathematical framework used in this analysis is sound—including the meshing procedure, the enforcement of periodic boundary conditions and the loading conditions. The computational cost of the simulation is simply too much for our current capabilities.

The effective applied triaxiality of 0.33 is a bit low compared to that seen in material regions expected to fail. For example, the triaxiality ahead of a crack tip can reach 2.0. Due to the

nature of the loading conditions, we cannot easily modify the applied triaxiality. It may be possible to prescribe tangential displacements (in addition to the axial one) in order to effect this response. However, this would need to be done assuming some sort of macroscopic response. At the point in the simulation where necking begins to happen, the macroscopic response no longer behaves the same as that of the dense material. Future work could be done to perform this modification and mitigate any problems which may arise.

Three-dimensional mesh generation

As a continuation of the FY 2011 work that was done looking at periodic two-dimensional arrays of initially circular voids in a plane strain framework, we wanted to extend this method to three dimensions.

The computational techniques for modeling a periodic boundary condition in three dimensions are identical to those used for the previous study in two dimensions, so no additional modification of the code was required.

The process of generating and meshing a randomly generated microstructure in an automated process turned out to be a considerable challenge. It was immediately decided to use selectively integrated TET10 elements due to their ease of meshing—compared to hexahedral elements—and their accuracy in modeling large deformation plasticity. The “trimesh” and “tetmesh” schemes worked well for meshing arbitrary surfaces and volumes, respectively.

While a number of issues were encountered along the way, ultimately we performed this task by generating the microstructure within Mathematica along with the corresponding Cubit journal file. A Bash script was used to run the various components, process data, and generate additional Cubit journal files to complete the task.

If desired, this procedure can be readily adapted to model more complex initial microstructures such as those found in anisotropic materials.

The Mathematica notebook and Bash script files used in this procedure are archived in the DART Workbench project “clustered_void_growth” within the “threedee/meshing” folder.

The overall procedure to generate a randomized three dimensional periodic microstructure of voids is:

- Using Mathematica, place voids randomly within the unit cell
- Using Cubit, replicate geometry and mesh in a periodic manner

The details of each of the steps are given below.

Void distribution in Mathematica

The structure of the Mathematica file to generate voids can be summarized as follows:

1. Establish microstructure parameters
 - Unit cell dimensions = {1, 1, 1}
 - Void volume fraction = 0.03
 - Void diameter distribution = uniform
 - Number of voids = 32
 - Mesh quality = 12
 - Minimum void spacing = 0.5 * diameter
2. Generate a trial void and see if it works
 - Random position within unit cell
 - Random diameter based on sizing function

- Ensure minimum void spacing is satisfied
 - If it fits, add it to the list
 - If it doesn't fit, discard it
3. Repeat until the number of voids is reached
 4. Find the best unit cell to mesh
 5. Create Cubit journal file to recreate and mesh geometry

This script is located in the DART Workbench at the following location:

- `workbench:clustered_void_growth/threedee/meshing/generation.nb`

Microstructure parameters

The microstructure parameters are what allow us to tune the microstructure. For the initial set of runs, we have chosen each void to be a spherical void with all voids having a uniform radius.

Ideally, this step would be set up to generate a microstructure closely resembling those found in actual materials, however we have had to make considerable allowances on a count of computational costs.

Estimate of computational costs

We can perform a simple calculation to determine the range of microstructure variables we can use on account of computational costs. A list of the microstructure and meshing variables is shown in Table 4. Note that the majority of these are calculated from the parameter set $\{n, q, \phi\}$.

Variable	Symbol	Type
number of voids	n	chosen
mesh quality	q	chosen
initial void volume fraction	ϕ	chosen
unit cell volume	V_{total}	chosen (but arbitrary)
dense volume	V_{dense}	calculated
void radius	r	calculated
element edge length	ℓ	calculated
element volume	V_{element}	calculated
number of elements	e	calculated

Table 4: Microstructure and meshing variables used in this analysis.

We choose a cube to be the unit cell of the periodic microstructure. For ease, we choose the edge length to be 1 meter, although since there is no length scale type parameter associated with this problem, it has no bearing on the results.

We can easily estimate the number of elements in a given mesh for a given set of parameters $\{n, q, \phi\}$. Given the total volume of the unit cell, V_{total} , the volume occupied by elements is $V_{\text{dense}} = (1 - \phi)V_{\text{total}}$. Given then, the number of elements is easily approximated as

$$e \approx V_{\text{dense}} / V_{\text{element}} \quad (1.1)$$

where V_{element} is the typical element volume.

For n spherical voids of uniform radius r , the total void volume is given by $\frac{4}{3}\pi r^3 n$. Given that this is the void volume fraction ϕV_{total} , we can solve for the radius:

$$r = \sqrt[3]{\frac{3\phi V_{\text{total}}}{4\pi n}} \quad (1.2)$$

The mesh quality parameter q defines how many element edge lengths fit around the circumference of the void at its largest diameter, so we may calculate this as

$$\ell = \frac{2\pi r}{q} \quad (1.3)$$

A standard tetrahedron of element edge length ℓ has volume $\frac{\sqrt{2}}{12}\ell^3$. Although standard tetrahedrons are not space-filling, we use this approximation to calculate the average element volume:

$$V_{\text{element}} \approx \frac{\sqrt{2}}{12}\ell^3 \quad (1.4)$$

Putting everything together leaves us with an approximation of the number of elements in a mesh for a given set of input parameters

$$e \approx \frac{\sqrt{2}}{\pi^2} \frac{nq^3}{\phi} \quad (1.5)$$

On a 12 GB machine (e.g. a redsky node), we can have about 250,000 selectively integrated TET10 elements without running out of memory. The following choices have been made with this in mind.

A number of previous studies have been done which have concluded that for these types of simulations, a mesh quality of around 16 is required for accurate results. In the first set of three dimensional simulations, we relax this requirement to 12 to allow for more voids and a smaller void volume fraction.

The physical void volume fraction is around $3 \cdot 10^{-4}$, however due to computational considerations we relax this to be $3 \cdot 10^{-2}$ for the first set of runs.

The appropriate number of voids in a unit cell is somewhat up for discussion. This value is where a length scale parameter could be given to the simulation. For the first set of runs, we have chosen $n = 32$.

In practice, the equation (1.5) is fairly accurate and typically overpredicts the actual value by 5–25%. A comparison of calculated values versus the actual values is given in Table 5.

n	q	ϕ	calculated	actual
32	16	0.03	626,042	497,675
32	12	0.03	264,111	249,493
16	12	0.03	132,055	127,974
16	16	0.10	93,906	85,655

Table 5: Comparison of the estimated number of elements to the actual number for a given mesh for several sets of microstructure parameters.

Choosing the best unit cell

Although the microstructure we are representing is periodic—and therefore infinite in dimensions—we must choose a particular unit cell to explicitly mesh and model to perform the simulation. The particular choice is arbitrary from a geometric standpoint. However, if poorly formed elements such as slivers are present, they could corrupt the system of equations and cause the solution to diverge. Therefore, we choose the unit cell to model based on meshability.

To choose the unit cell which is most easily meshed, we create an objective function Φ within Mathematica to avoid problem elements and find the unit cell which maximizes this objective. This global objective function is the maximum of each sub-objective function ϕ as given below.

Problems at corners

In order to track unit cell dimensions (length, width, height), we need a node to be at each corner of the unit cell. Therefore, the origin cannot be within a void. We create n sub-objective functions to ensure this, with each objective function being the distance of the origin away from the void.

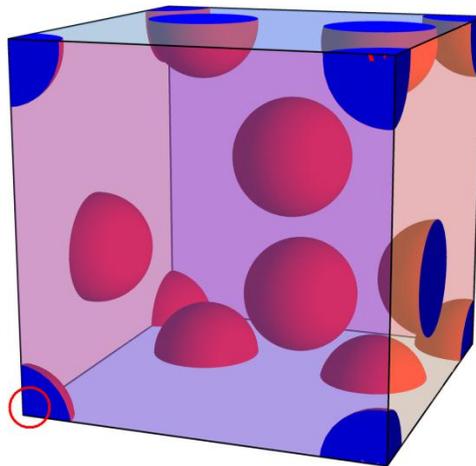


Figure 17: A problematic unit cell choice because the corner is within a void.

Problems at edges

To avoid cases where an edge of the unit cell intersects a void, we create a sub-objective function for each case with the objective function being the shortest distance between the void surface and the edge. Since there are 12 edges to a cube, this results in $12n$ sub-objective functions.

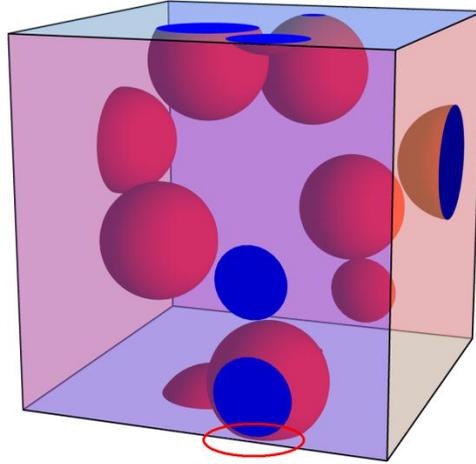


Figure 18: A problematic unit cell choice because an edge is very close to a void surface.

Problems at faces

To avoid cases where a face of the unit cell is very close to the surface of a void, we create a sub-objective function with the objective function being the distance of the unit cell face towards the corresponding void extent. Since there are 6 faces, and 2 corresponding extents per void, this creates $12n$ sub-objective functions.

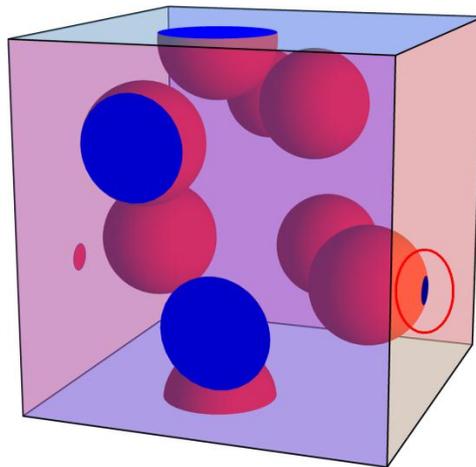


Figure 19: A problematic unit cell choice because a face is very close to a void surface extent.

This results in an objective function

$$\Phi(\Delta x, \Delta y, \Delta z) = \max_{\text{each } \phi} \phi \quad (1.6)$$

which we maximize to find the offset $(\Delta x, \Delta y, \Delta z)$ which we then apply to each void.

The frequency with which at least one of these problematic issues occurs in the original microstructure increases with the number of voids in the unit cell. In practice, one or more of

these problems occurred virtually all of the time with the chosen set of microstructure parameters and so this scheme was adopted to avoid these issues.

Meshing procedure

Once the microstructure and unit cell are chosen, we can begin the meshing procedure.

Due to the periodic requirement, the meshing procedure is long and tortuous, and the procedure given here was found by trial and error to be the most reliable. The general procedure is summarized as follows:

1. Create the geometry within Cubit
2. Set up mesh intervals along external curves
3. Mesh surfaces lying on the negative faces of the unit cell
4. Replicate these faces onto the positive faces
5. Imprint these surface meshes into the 3D geometry
6. Mesh the other surfaces
7. Mesh the interior
8. Perform node matching on corresponding surfaces

The details of each step are as follows.

1. Create the geometry within Cubit

To do this, we create the unit cell, then find each void which intersects the unit cell and create the sphere. We then perform a boolean operation to subtract each void sphere from the unit cell and output that as a cub file. An example journal file for this operation is shown in Table 6.

```
# reset everything
reset

# create the unit cell solid and move it into place
brick x 1 y 1 z 1
move vertex 7 location 0 0 0 nomerge

# create each void and move it into place
create sphere radius 0.0607148
move volume 2 location x -0.00777978 y -0.0264078 z 0.716514
...

# perform the boolean operation to subtract the voids
subtract volume 2 to 47 from volume 1

# save the result
save as "cube_geometry.cub" overwrite
```

Table 6: An example Cubit journal file to create and save the void geometry.

An example of the geometry generated after this procedure is shown in Figure 20.

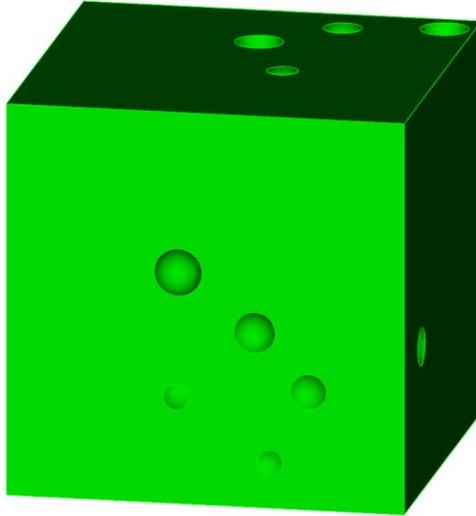


Figure 20: The generated geometry after subtracting the voids from the unit cell.

2. Set up mesh intervals along external curves

Surfaces must be meshed before volumes, and curves must be meshed before surfaces, and vertices must be meshed before curves.

As it turns out, the circular curves seen in Figure 20 have a vertex at the start/end point. For a given circular curve, the location of this vertex is not necessarily the same on the front face as it is on the back face. However, in cases where it is not the same, it varies by 180° . Therefore, if we mesh each curve with an interval divisible by 2, the nodes will match.

To determine the appropriate amount of nodes per curve, we first find the curve length which can be output by Cubit using the “list curve all” command. The number of intervals along a given curve is then

$$n_i = 2 \cdot \left\lfloor \frac{1}{2} \ell_c / \ell \right\rfloor + 2 \quad (1.7)$$

where ℓ_c is the curve length and ℓ is the target element edge length. This function simply rounds ℓ_c / ℓ to a nearby multiple of 2. This procedure was done for all curves in which the end vertex was the same as the start vertex.

The placement of nodes can be seen in Figure 21.

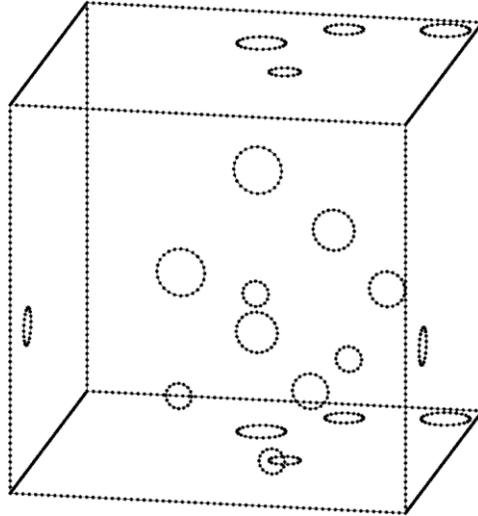


Figure 21: The placement of nodes along each curve.

3. Mesh surfaces lying on the negative faces of the unit cell

After nodes were placed, the negative faces (surfaces with normal $-x$, $-y$, and $-z$) were meshed. Since the ID of these is not known, they were specified by location of the centroid.

4. Replicate these faces onto the positive faces

The mesh at $x=0$ was simply translated by $(1,0,0)$ to create the desired mesh at $x=1$. A sample Cubit journal file for the previous three steps is shown in Table 7.

```
# reset everything
reset

# open the geometry file
open "cube_geometry.cub"

# redefine block numbers and element types
reset block
block 1 volume all
block 1 element type tetra10
block 2 surface all
block 2 element type tri6

# set surface and volume target element edge length
surface all size 0.0317902
volume all size 0.0317902

# set up surface/volume meshing schemes
surface all scheme trimesh
volume all scheme tetmesh

# mesh boundary curves
curve 13 scheme equal
curve 13 interval 10
mesh curve 13
...

# mesh and save the negative x surface
mesh surface with x_coord = 0
export mesh "mesh_periodic_1.g" dimension 3 overwrite

# translate the surface and save it at the positive face location
move volume all x 1 include_merged
```

```

export mesh "mesh_periodic_2.g" dimension 3 overwrite
move volume all x -1 include_merged
delete mesh surface all

# repeat for y and z sides
...

```

Table 7: An example Cubit journal file to mesh the boundary curves and faces.

The resulting six surface meshes (each is a separate file) can be seen in Figure 22.

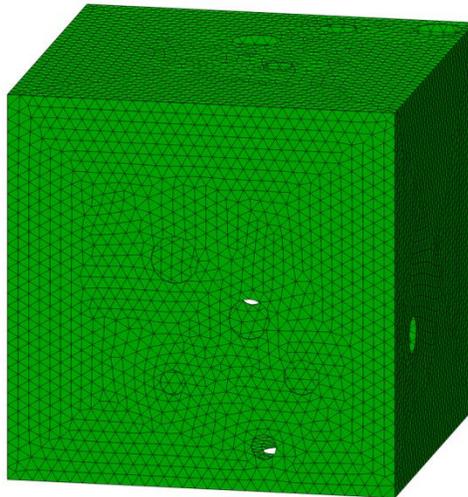


Figure 22: Meshes of the six external faces of the unit cell. Note the interior and void surfaces are unmeshed at this point.

5. *Imprint these surface meshes into the 3D geometry*

At this point, we have a geometry file along with six surface mesh files for each of the six faces of the unit cell. We can “imprint” these meshes onto the geometry by loading the geometry file and then importing each mesh.

This step works most of the time with Cubit successfully associating the mesh in the file with the appropriate surface. However, it doesn’t always work and sometimes an error is thrown. Therefore, the success of this step must be checked.

6. *Mesh the other surfaces*

Since the mesh is now guaranteed to be periodic, we can mesh the rest of the surfaces using a simple “trimesh” scheme.

7. *Mesh the interior*

With all surfaces meshed, the interior is meshed using the “tetmesh” scheme.

A sample script for the last three steps is shown in Table 8.

```

# reset everything
reset

# open the geometry file
open "cube_geometry.cub"

```

```

# redefine block numbers and element types
reset block
block 1 volume all
block 1 element type tetra10
block 2 surface all
block 2 element type tri6

# set surface and volume target element edge length
surface all size 0.0317902
volume all size 0.0317902

# set up surface/volume meshing schemes
surface all scheme trimesh
volume all scheme tetmesh

# attempt to import and imprint the mesh for
# each external face
import mesh "mesh_periodic_1.g"
import mesh "mesh_periodic_2.g"
import mesh "mesh_periodic_3.g"
import mesh "mesh_periodic_4.g"
import mesh "mesh_periodic_5.g"
import mesh "mesh_periodic_6.g"

# mesh the rest of the surfaces
mesh surface all

# mesh the volume
mesh volume all

# save it
export mesh "mesh_cube.g" dimension 3 block 1 overwrite

```

Table 8: An example Cubit journal file to imprint the external faces and mesh the interior.

8. *Perform node matching on corresponding surfaces*

Rather unfortunately, even after the precautions taken in the procedure to ensure nodes on corresponding faces match, nodal positions can still be off by a small amount. To correct this final issue, we perform the following face-matching routine:

- For each node on the $x = 1$ face with coordinates $(1, y, z)$
 - Find the closest node to $(0, y, z)$ with coordinates $(0, y', z')$
 - Move the original node to $(1, y', z')$

The resulting mesh then exactly matches on the $x = 0$ and $x = 1$ faces. That is, for every node with coordinates $(0, y, z)$ there is a corresponding node with exactly the coordinates $(1, y, z)$, and vice-versa.

This procedure is also repeated on the other two pairs of surfaces.

The final mesh can be seen in Figure 23.

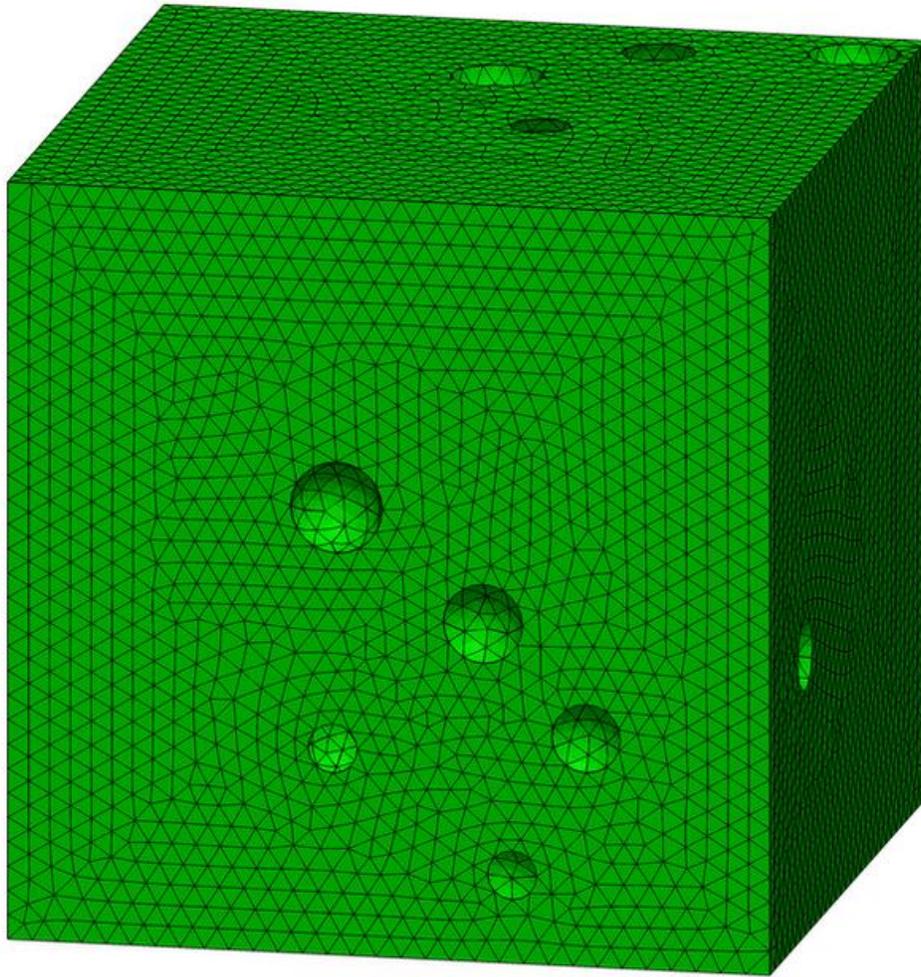


Figure 23: The final and truly periodic mesh in all its glory.

Problems encountered while meshing

At first glance, the meshing procedure outline previously may seem overly complicated. However, there numerous bugs and issues encountered which influenced this path.

Cubit IDs change

In general, there is no guarantee that “surface 1” after the boolean subtraction operation is the same surface as “surface 1” before the operation. It may have disappeared or changed values. This is the reason we refer to surfaces by coordinate rather than ID.

In addition, it is well known that, for the same journal file, IDs frequently change between code versions. Thus, referring to entities by ID in a particular journal file will likely cause the journal file to not function correctly in future code versions.

Copy mesh produces unreliable results

There is a `copy mesh` method in Cubit which purportedly creates a mesh on the target surface from the mesh on the source surface. If the two surfaces are geometrically identical (within a rigid body motion), this command should reproduce the mesh identically (within the same rigid body motion). However, in practice, this was not the case.

For a well-formed source mesh, the created target mesh often contained inverted elements. Other times, the operation would simply fail. Even in cases where the method appeared to have worked (by visual inspection), the nodal coordinates on the target mesh would significantly differ from those on the source mesh, making enforcement of the periodic condition impossible.

For these reasons, we abandoned use of the `copy mesh` method.

Importing more than one surface at once problematic

During the step where we import surface meshes, we found that trying to import the entire surface mesh in a single file often failed. By trial and error, we found that individually importing surfaces worked much more reliably. This is the reason for 6 separate surface mesh files.

Anisotropic plasticity model development

This chapter contains the formulation and implementation details of an anisotropic plasticity material model.

After investigating the effect of void texture on the macroscopic response, it became clear that an isotropic material model would be unable to accurately capture the response of a material with an anisotropic void distribution (this includes most ductile metals). Because no anisotropic plasticity models are available in SIERRA, we decided to formulate and implement one such formulation into the Area code.

Procedure

The model is relatively easy to formulate, but difficult to implement in a robust fashion. In particular, this formulation is complicated by not using the typical decomposition of the deformation tensor into elastic and plastic parts—also known as hypoelasticity.

This derivation will be given for the case in which the material axes of symmetry are aligned with the coordinate axes in the reference configuration.

Formulation

The model is effectively a linearly hardening model with an anisotropic yield function. We break up the deformation gradient into an elastic and plastic part

$$\mathbf{F} = \mathbf{F}^e \cdot \mathbf{F}^p \quad (8)$$

with the further restriction that $\mathbf{F}^p = \mathbf{U}^p$ is symmetric by definition. We can find the polar decomposition of \mathbf{F} to calculate

$$\mathbf{F} = \mathbf{R} \cdot \mathbf{U} \quad (9)$$

where \mathbf{R} is a rotation tensor with the properties $\det \mathbf{R} = 1$ and $\mathbf{R}^T = \mathbf{R}^{-1}$ and \mathbf{U} is the right stretch tensor which is symmetric. We can also apply the polar decomposition to \mathbf{F}^e to get

$$\mathbf{F}^e = \mathbf{R}^e \cdot \mathbf{U}^e \quad (10)$$

where $\mathbf{R}^e = \mathbf{R}$ and \mathbf{U}^e is symmetric.

Elastic response

In this section, to simplify the presentation, we will assume $\mathbf{U}^p = \mathbf{I}$ and therefore $\mathbf{F} = \mathbf{F}^e$.

In plasticity, the elastic deformation typically has a magnitude of less than 1%. Because of this, the particular elasticity law one chooses is nearly insignificant as they all behave similarly in the regime of interest. For computational convenience, we choose the elastic response to be

$$\mathbf{R}^T \cdot \boldsymbol{\sigma} \cdot \mathbf{R} = \frac{1}{2} \kappa (J - J^{-1}) \mathbf{I} + 2\mu \operatorname{dev} \bar{\mathbf{U}} \quad (11)$$

which satisfies the usual requirements on objectivity where

$$\bar{\mathbf{U}} \stackrel{\text{def}}{=} \mathcal{J}^{-1/3} \mathbf{U} \quad (12)$$

is the isochoric stretch tensor. Note that the volumetric response is decoupled from the deviatoric response.

Infinitesimal elastic response

As an exercise, we can linearize this about a state of zero deformation by making the approximation

$$\mathbf{F} = \mathbf{I} + \varepsilon \mathbf{H} \quad (13)$$

for $|\varepsilon| \ll 1$. The polar decomposition then becomes

$$\begin{aligned} \mathbf{R} &= \mathbf{I} + \frac{1}{2} \varepsilon (\mathbf{H} - \mathbf{H}^T) + O(\varepsilon^2) \\ \mathbf{U} &= \mathbf{I} + \frac{1}{2} \varepsilon (\mathbf{H} + \mathbf{H}^T) + O(\varepsilon^2) \end{aligned} \quad (14)$$

and the stress becomes

$$\boldsymbol{\sigma} = \varepsilon \left[\mu \operatorname{dev}(\mathbf{H} + \mathbf{H}^T) + \frac{1}{3} \kappa (\operatorname{tr} \mathbf{H}) \mathbf{I} \right] + O(\varepsilon^2) \quad (15)$$

which, if we use the infinitesimal small strain tensor

$$\boldsymbol{\varepsilon} \stackrel{\text{def}}{=} \varepsilon \frac{1}{2} (\mathbf{H} + \mathbf{H}^T) \quad (16)$$

the stress can be written as

$$\boldsymbol{\sigma} = \mu \operatorname{dev} \boldsymbol{\varepsilon} + \frac{1}{3} \kappa (\operatorname{tr} \boldsymbol{\varepsilon}) \mathbf{I} + O(\|\boldsymbol{\varepsilon}\|^2) \quad (17)$$

which is linear elasticity.

Thus, the elasticity law we use here collapses to linear elasticity in the small strain limit and the constants μ and κ have their usual meanings of the shear modulus and the bulk modulus, respectively.

Yield surface

To form the yield surface, we use the Hill yield criterion with a slight modification to allow for strain hardening. We form the equivalent stress

$$\sigma_{\text{eq}} \stackrel{\text{def}}{=} c_1 (\sigma_{22} - \sigma_{33})^2 + c_2 (\sigma_{33} - \sigma_{11})^2 + c_3 (\sigma_{11} - \sigma_{22})^2 + 2c_4 \sigma_{23}^2 + 2c_5 \sigma_{31}^2 + 2c_6 \sigma_{12}^2 \quad (18)$$

which comes from a simple extension of the Von-Mises equivalent stress. The parameters $\{c_1, c_2, \dots, c_6\}$ must be fit from experimental data. The yield surface is then defined by $\sigma_{\text{eq}} = \sigma_y$ with a purely elastic response taking place when $\sigma_{\text{eq}} < \sigma_y$.

The yield stress σ_y is also fit from experimental data. Note that this system is non-unique. This is deliberate to allow for the constants to have physical meanings while retaining the traditional definition of σ_y as the yield stress in a particular direction—typically the longitudinal direction.

The yield stress evolves linearly as the material hardens due to plastic strain according to the law

$$\sigma_y^{\text{def}} = \sigma_y^0 + K \varepsilon_{p,\text{eq}} \quad (19)$$

where σ_y^0 is the initial yield stress (in the 1–1 direction), K is the hardening modulus, and $\varepsilon_{p,\text{eq}}$ is the equivalent plastic strain which evolves according to

$$\dot{\varepsilon}_{p,\text{eq}}^{\text{def}} = \sqrt{\frac{2}{3}} \|\mathbf{d}^p\| \quad (20)$$

where

$$\mathbf{d}^p = \dot{\mathbf{U}}^p \cdot (\mathbf{U}^p)^{-1} \quad (21)$$

is the rate of plastic deformation tensor.

Parameter fitting

The six constants $\{c_1, c_2, \dots, c_6\}$ are fit using the following procedure.

The yield stress in each direction must be experimentally obtained by performing tensile tests and shear failure tests. The yield stress of each of these must be obtained. The constants may be fit with the equations

$$\begin{aligned} c_1 &= \frac{1}{2} \left[-1 + \frac{1}{(\sigma_{22}^y / \sigma_{11}^y)^2} + \frac{1}{(\sigma_{33}^y / \sigma_{11}^y)^2} \right] \\ c_2 &= \frac{1}{2} \left[1 - \frac{1}{(\sigma_{22}^y / \sigma_{11}^y)^2} + \frac{1}{(\sigma_{33}^y / \sigma_{11}^y)^2} \right] \\ c_3 &= \frac{1}{2} \left[1 + \frac{1}{(\sigma_{22}^y / \sigma_{11}^y)^2} - \frac{1}{(\sigma_{33}^y / \sigma_{11}^y)^2} \right] \end{aligned} \quad (22)$$

where σ_{11}^y is the yield stress obtained from a simple tension tests along the 1–1 direction and

$$c_4 = \frac{1}{2(\tau_{23}^y / \sigma_{11}^y)^2} \quad c_5 = \frac{1}{2(\tau_{31}^y / \sigma_{11}^y)^2} \quad c_6 = \frac{1}{2(\tau_{12}^y / \sigma_{11}^y)^2} \quad (23)$$

where τ_{12}^y is the yield stress (in shear) of a simple shear tests in the 1–2 direction.

Note that for the typical J-2 plasticity isotropic yield surface, we have

$$\begin{aligned} \sigma_{11}^y &= \sigma_{22}^y = \sigma_{33}^y = \sigma_y \\ \tau_{12}^y &= \tau_{23}^y = \tau_{31}^y = \frac{1}{\sqrt{3}} \sigma_y \end{aligned} \quad (24)$$

and therefore

$$\begin{aligned} c_1 = c_2 = c_3 &= \frac{1}{2} \\ c_4 = c_5 = c_6 &= \frac{3}{2} \end{aligned} \tag{25}$$

In this case, the equivalent stress is equal to the Von-Mises stress.

Flow rule

The principle of maximum plastic dissipation effectively says the following: During yielding, the plastic strain—however you wish to define it—evolves in such a way that the Cauchy stress evolves in a direction normal to the yield surface.

This rule is good for two reasons: (1) it makes physical sense based on thermodynamic principles and (2) it preserves the symmetry of the stiffness matrix. For these reasons, and for lack of experimental data suggesting otherwise, we adopt this flow rule for our model.

In isotropic plasticity, for a point on the yield surface, the Cauchy stress is always normal to the yield surface. Thus, this is a trivial exercise. The radial return algorithm literally describes this approach as the trial stress outside the yield surface returns radially. This is further made clear in that the isotropic yield surface can be visualized in three dimensional space as a sphere. Upon yielding, the stress evolves literally in the radial direction.

For our anisotropic case, the yield surface is a 5-dimensional manifold in 6-dimensional space and the task of finding the appropriate path

Return mapping algorithm

The most expensive task of the return mapping algorithm is finding the point on the yield surface whose normal points in the direction of the trial stress. This is depicted for a two-dimensional system in Figure 24.

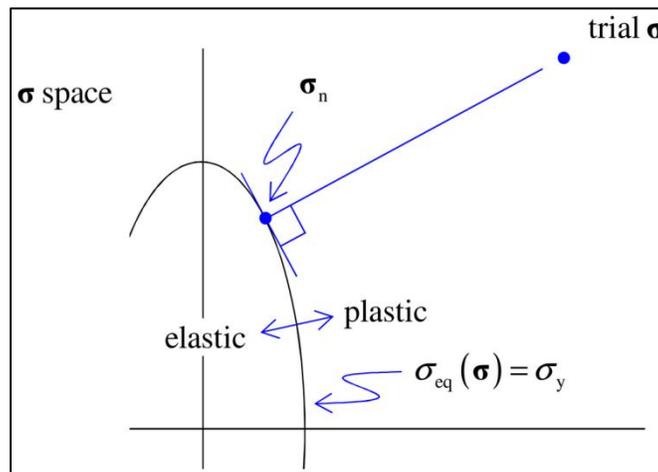


Figure 24: The yield surface as depicted as an ellipse in two dimensions. The return mapping algorithm finds σ_n as a function of σ_y and the trial σ .

To find this point, an iterative Newton’s method approach is used. Since the equivalent stress function is linear, the first trial σ_n is chosen by radially scaling the trial σ back to the yield

surface. The residual is then calculated by projecting the difference onto a plane tangent to the yield surface. The system is then linearized about this point and an update is found using Newton's method.

Starting with the definition of the residual

$$\mathbf{r} \stackrel{\text{def}}{=} \Delta \boldsymbol{\tau} - \frac{\nabla f}{\|\nabla f\|} \left(\frac{\nabla f}{\|\nabla f\|} \cdot \Delta \boldsymbol{\tau} \right) \quad (26)$$

where

$$\Delta \boldsymbol{\tau} \stackrel{\text{def}}{=} \boldsymbol{\tau} - \boldsymbol{\tau}^n \quad (27)$$

and f is the equivalent stress function (i.e. Hill's yield criteria) and ∇f is a function of $\boldsymbol{\tau}^n$. We can rewrite the residual in index notation as

$$r_i = \Delta \tau_i - \frac{\nabla f_i}{\|\nabla f\|^2} (\nabla f_k \Delta \tau_k) \quad (28)$$

which we take the gradient of to give us

$$\frac{dr_i}{d\tau_j^n} = \frac{d\Delta \tau_i}{d\tau_j^n} - \frac{d\nabla f_i}{d\tau_j^n} \frac{1}{\|\nabla f\|^2} (\nabla f_k \Delta \tau_k) + \frac{2\nabla f_i}{\|\nabla f\|^3} \frac{d\|\nabla f\|}{d\tau_j^n} (\nabla f_k \Delta \tau_k) - \frac{\nabla f_i}{\|\nabla f\|^2} \left(\frac{d\nabla f_k}{d\tau_j^n} \Delta \tau_k + \nabla f_k \frac{d\Delta \tau_k}{d\tau_j^n} \right) \quad (29)$$

To simplify this, we can calculate

$$\frac{d\|\nabla f\|}{d\tau_j^n} = \frac{\nabla f_i}{\|\nabla f\|} \frac{d\nabla f_i}{d\tau_j^n} \quad (30)$$

and

$$\frac{d\Delta \tau_i}{d\tau_j^n} = \frac{d}{d\tau_j^n} (\tau_i - \tau_i^n) = -\delta_{ij} \quad (31)$$

and use these in (29) to find

$$\frac{dr_i}{d\tau_j^n} = -\delta_{ij} - \frac{d\nabla f_i}{d\tau_j^n} \frac{\nabla f \cdot \Delta \boldsymbol{\tau}}{\|\nabla f\|^2} + \frac{\nabla f_i}{\|\nabla f\|^2} \frac{d\nabla f_k}{d\tau_j^n} \left(2\nabla f_k \frac{\nabla f \cdot \Delta \boldsymbol{\tau}}{\|\nabla f\|^2} - \Delta \tau_k \right) + \frac{\nabla f_i \nabla f_j}{\|\nabla f\|^2} \quad (32)$$

The update to the system is found by taking

$$\boldsymbol{\tau}_n^{I+1} = \boldsymbol{\tau}_n^I - \left(\frac{d\mathbf{r}}{d\boldsymbol{\tau}^n} \right)^{-1} \cdot \mathbf{r}^I \quad (33)$$

In practice, with moderately anisotropic yield functions, this converges in 3–4 iterations to a tolerance near machine precision (10^{-16}).

Inverse stress calculation

Due to the formulation of the flow rule, an implicit equation must be solved to find the elastic stretch \mathbf{U}^e in terms of the un-rotated stress $\boldsymbol{\tau}$. We can take the determinant of (11) to find

$$\text{tr } \boldsymbol{\tau} = \frac{3}{2} \kappa (J - J^{-1}) \quad (34)$$

which we can solve to find

$$J = \frac{\text{tr } \boldsymbol{\tau}}{3\kappa} + \sqrt{1 + \left(\frac{\text{tr } \boldsymbol{\tau}}{3\kappa}\right)^2} \quad (35)$$

Taking the deviatoric part of (11) yields

$$\text{dev } \boldsymbol{\tau} = 2\mu \text{dev } \bar{\mathbf{U}} = J^{-1/3} 2\mu \text{dev } \mathbf{U} \quad (36)$$

From here, we know

$$\det \mathbf{U} = J \quad (37)$$

and

$$\mathbf{U} = \text{dev } \mathbf{U} + \alpha \mathbf{I} \quad (38)$$

for some real number α . Since the elastic part of the deformation is typically close to \mathbf{I} , we can define our residual to be

$$r^I \stackrel{\text{def}}{=} \det(\text{dev } \mathbf{U} + \alpha^I \mathbf{I}) - J \quad (39)$$

which has a gradient of

$$\frac{dr^I}{d\alpha^I} = \det(\text{dev } \mathbf{U} + \alpha^I \mathbf{I}) \text{tr} \left[(\text{dev } \mathbf{U} + \alpha^I \mathbf{I})^{-1} \right] \quad (40)$$

To converge, we choose our initial point $\alpha^0 = 1$ and use Newton's method to find the exact solution. This scheme is summarized in Table 9.

1. Calculate $J = \frac{\text{tr } \boldsymbol{\tau}}{3\kappa} + \sqrt{1 + \left(\frac{\text{tr } \boldsymbol{\tau}}{3\kappa}\right)^2}$
2. Calculate $\text{dev } \mathbf{U} = \frac{J^{1/3}}{2\mu} \text{dev } \boldsymbol{\tau}$
3. Set $I = 0$, choose $\alpha^I = 1$
 - a. Calculate $\mathbf{U}^I = \text{dev } \mathbf{U} + \alpha^I \mathbf{I}$
 - b. Calculate $r^I = \det \mathbf{U}^I - J$
 - c. If $\|r^I\| \ll 1$, exit
 - d. Calculate $\alpha^{I+1} = \alpha^I - \frac{r^I}{\det(\mathbf{U}^I) \text{tr}[(\mathbf{U}^I)^{-1}]}$
 - e. Set $I = I + 1$ and iterate

Table 9: Algorithm to find elastic stretch as a function of unrotated stress.

In practice, for plastic materials, this algorithm converges to within machine precision after 1–4 iterations. Also, note that the quantity $\det(\mathbf{U}^I) \text{tr}[(\mathbf{U}^I)^{-1}]$ simplifies considerably and the calculation the calculation of the full inverse is not necessary.

Implementation

A C++ implementation of this algorithm is available within the finite element code Ares¹ under the `anisotropic_void_plasticity` material model.

Preliminary results

In formed materials, results of anisotropy can often be seen in the asymmetric response of a specimen. In Figure 25, the results of a tensile specimen of aluminum 7050-T74 tested to failure at 300 °C can be seen. Although the initial geometry is axisymmetric, the cross section of the necked region is noticeably elliptical.

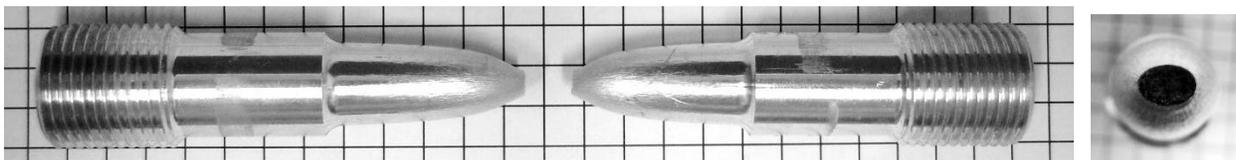


Figure 25: The tensile specimen tested at 300 °C. At right is a view of the asymmetric fracture surface.

¹ <http://www.sourceforge.net/projects/aresfea/>

Using the anisotropic plasticity model, we ran a simulation of a tensile specimen with the yield stress in the long transverse direction reduced to 90% and the yield stress in the short transverse direction reduced to 80% compared to the longitudinal (axial) direction. These values were chosen to show a proof of concept and do not necessarily represent the actual material properties. A more rigorous testing procedure would be necessary to characterize the anisotropy of the material.

The results of this are shown in Figure 26. On the left, one can clearly see the anisotropy of the necked region. The two middle pictures show the profile of the specimen rotated 90 degrees relative to one another.

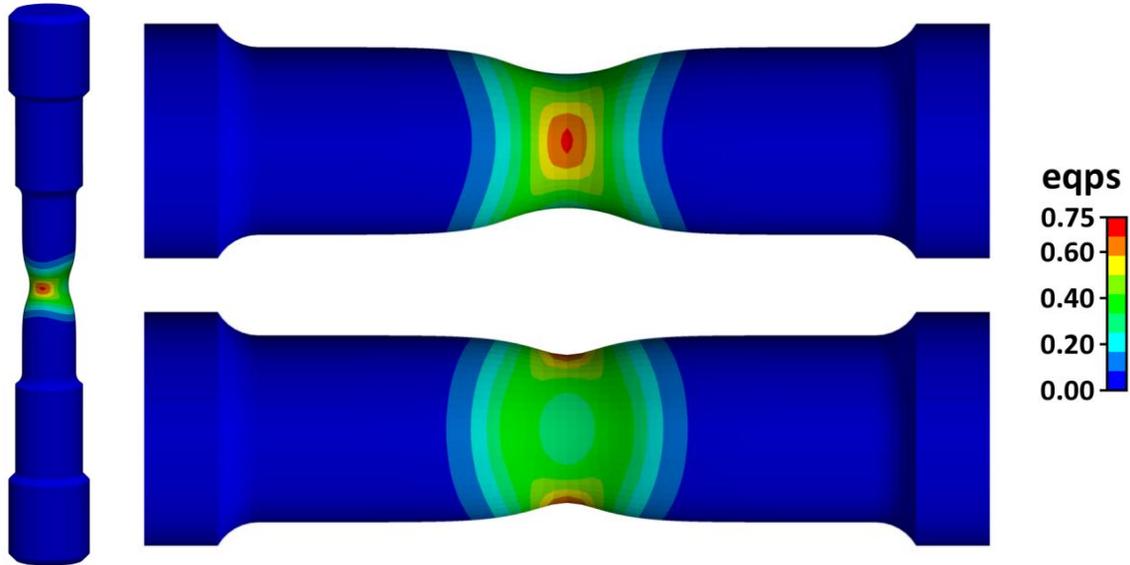


Figure 26: Equivalent plastic strain in a tensile specimen with anisotropic properties.

The results of this study show even a modest amount of material anisotropy can affect the shape of the necked region.

Mathematical enforcement of boundary conditions

The mathematical framework necessary for enforcing a periodic boundary condition is given below.

Enforcement of periodic boundary condition

In our framework, we have a unit cell which is repeated according to

$$\psi\left(\mathbf{X} + \sum_{i=1}^3 n_A \mathbf{R}_A\right) = \psi(\mathbf{X}) \quad (1.41)$$

where ψ is the material function (0 within a void, 1 within the matrix), \mathbf{R}_A are the unit cell directions, and n_A are any integer. In short, this equation says that if there is a void at \mathbf{X} , there is also a void at $\mathbf{X} + \mathbf{R}_1$, $\mathbf{X} + 2\mathbf{R}_1$, etc... This framework is identical to that of crystal lattice, except that the geometry is random and not ordered.

A periodic boundary condition requires the displacement field to satisfy

$$\mathbf{u}(\mathbf{X} + d\mathbf{X}) - \mathbf{u}(\mathbf{X}) = \mathbf{u}\left(\mathbf{X} + d\mathbf{X} + \sum_{i=1}^3 n_A \mathbf{R}_A\right) - \mathbf{u}\left(\mathbf{X} + \sum_{i=1}^3 n_A \mathbf{R}_A\right) \quad (1.42)$$

for an arbitrary position \mathbf{X} and vector $d\mathbf{X}$. Note that this is not the same as enforcing the condition $\mathbf{u}(\mathbf{X} + n_A \mathbf{R}_A) = \mathbf{u}(\mathbf{X})$ which is overly restrictive.

This boundary condition enforcement is illustrated in Figure 27. On the left, the unit cell is shown as the gray dashed square. The locations of four material points are also shown. Equation (1.42) forces the relative positions given by the two red arrows to be identical, as the shape of the left and right boundaries of the unit cell must match. No restriction is required on the relative position of the left and right boundary, as shown by the green arrow. On the right, a possible deformed configuration is given.

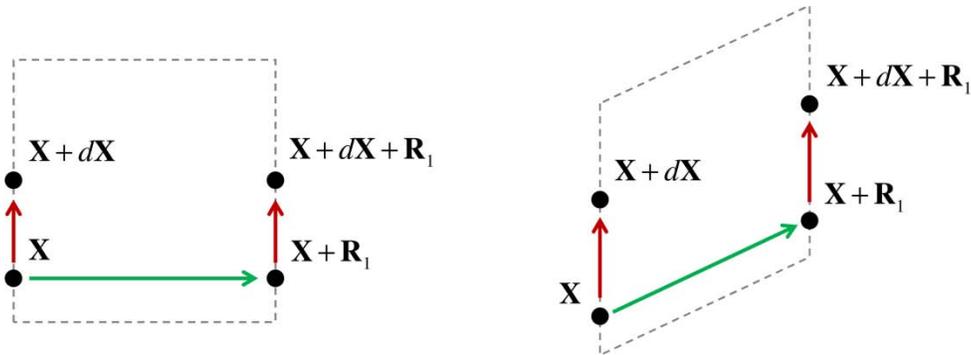


Figure 27: The initial configuration (left) and one possible deformed configuration (right) of four material points.

Since we have taken great pains to ensure that nodes on the $x=1$ face have a corresponding node on the $x=0$ face, ensuring that corresponding nodal displacements are periodic also ensures the point-wise displacement on the face is periodic.

Simple linear constraint example

Consider the convex system energy function

$$\phi = 3x^2 + y^2 + 2xy + 8y \quad (1.43)$$

where our two unknowns are (x, y) . Minimizing this is a trivial exercise in which we find $\phi = -24$ at $(x, y) = (2, -6)$.

Consider now the case in which we constrain the solution to satisfy $x = y$ due to some kinematic constraint. The system then becomes $\phi = 8x + 6x^2$ and the minimum $\phi = \frac{-8}{3}$ is obtained at $(x, y) = (\frac{-2}{3}, \frac{-2}{3})$.

This simple 2-variable example is exactly analogous to the procedure done on the full system of equations in our actual problem. To satisfy the periodic boundary conditions, we form linear constraints of the form

$$a_i = a_i^0 + \sum_{J,j} c_{Jj} a_{Jj} \quad (1.44)$$

where each a_{Jj} is a component of a nodal displacement and the c_{Jj} are constants found in our application of the constraint.

A more complete discussion of this can be found in previous documentation².

² T. D. Kostka, "On better understanding dilute void growth in ductile metals", Ph.D. dissertation, University of California, Berkeley, 2010.

Efficiency of a plane strain formulation

Three different computational frameworks were investigated for solving the same plane strain solid mechanics boundary value problem—Standard 3D, Reduced 3D, and True 2D. The first of these denotes what is currently available in three-dimensional codes by having a single layer of elements and constraining the vertical displacement to be zero at each node. The “True 2D” framework denotes a framework in which elements are two-dimensional and where each node only has 2 displacement unknowns. An intermediate “Reduced 3D” framework uses analytical constraints to produce a truly 2D solution within the three-dimensional framework.

Since the boundary value problem is identical in all three cases, the results match to within the specified numerical tolerance and the primary difference between the cases is the computational effort (time, memory) required to solve the problem.

Compared to the “Standard 3D” framework, the “True 2D” framework reduced the analysis time by 84%—a factor of 6.25. In addition, the RAM memory footprint was reduced by 81% and output file size by 51%.

An intermediate framework in which corresponding node pairs are constrained to have the same displacement reduced the analysis time by 65% and the memory footprint by 32%.

These results show a vast performance increase obtained by the two-dimensional framework. The reduction in computational time—a factor of 6.25—is above what one would expect based purely on $O(N)$ type approximations. Although this test was performed on a single computational architecture (a dual Intel Xeon X5570 setup), the performance increase is expected to be consistent across other modern processor configurations.

Procedure

In this section, we outline the procedure used to obtain the results.

Planar plane strain

In the most general sense, the “plane strain” condition simply means that the out-of-plane strain is zero at all points and that the displacement is not a function of the out-of-plane coordinate. Mathematically, this constraint yields

$$\mathbf{u}(X, Y, Z) = u_1(X, Y)\mathbf{e}_1 + u_2(X, Y)\mathbf{e}_2 + u_3(X, Y)\mathbf{e}_3 \quad (1.45)$$

where we take \mathbf{e}_3 as the out-of-plane direction.

A subset of this case is planar deformation—often simply called “plane strain” itself—in which the out of plane deformation is zero. In this case, we have $u_3 = 0$ and are left with only two displacement unknowns at each point.

In this analysis, we enforce $u_3 = 0$ and call the framework *planar* plane strain.

The three plane strain frameworks we examine are as follows.

- **Standard 3D**—a standard 3D framework (3 unknowns per node) is used and the vertical displacement of all nodes is prescribed to be zero.
- **Reduced 3D**—a standard 3D framework is used. In addition to setting the vertical displacement of all nodes to be zero, for each pair of nodes that share the same in-plane coordinates, the displacement of the top node is set to be equal to the displacement of the bottom node. This results in a reduction of the number of effective unknowns when solving the linear system.
- **True 2D**—the underlying framework is 2D (2 unknowns per node). The third unknown is implicitly assumed to be zero to accommodate the plane strain boundary condition.

Although the system of equations for the “Reduced 3D” and “True 2D” cases are equivalent, the major differences lie in the amount of memory space required for the stiffness matrix and the vector list of unknowns. This memory requirement between these two is approximately a factor of 3.

Meshing

The mesh was created using Cubit and represents a randomized microstructure of elliptical voids. The three-dimensional meshes were formed by extruding the elements a small distance out of plane. The mesh can be seen in Figure 28.

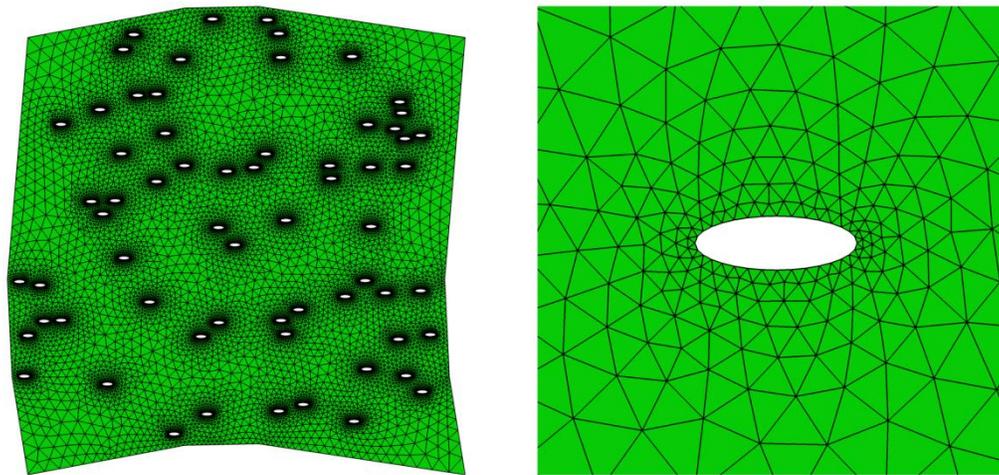


Figure 28: The mesh used in the analysis (left) and a close-up of the mesh around a single void (right).

The mesh has been intentionally created such that elements near the void surface are relatively smaller compared to those in the middle of the body.

Element formulation

The Ares³ finite element code was used to perform these simulations. Ares is an open source nonlinear implicit finite element software with multithreaded capabilities built on a shared memory architecture.

The interpolation and cubature type for each framework are shown in Table 10.

Framework	Element	In-plane cubature	Out-of-plane cubature
Standard 3D	WEDGE15	TRI7P5O	GAUSS2P3O
Reduced 3D	WEDGE12	TRI7P5O	GAUSS1P1O
True 2D	TRI6	TRI7P5O	n/a

Table 10: Element interpolation and cubature types used in each framework.

For all frameworks, the in-plane element interpolation is quadratic and the integration is done using a 7 point, 5th order scheme. In the out-of-plane direction, the interpolation order is quadratic (WEDGE15) or linear (WEDGE12). Although the integration order may seem excessive, it is important to realize that the bulk of the computational time is spent in the linear solver which is not affected by the integration scheme.

The number of nodes, element, unknowns, and cubature point in each framework is shown in Table 11.

Framework	Nodes	Elements	Actual unknowns	Effective unknowns	Cubature points
Standard 3D	133,892	28,553	401,676	281,832	399,742
Reduced 3D	118,560	28,553	355,680	118,093	199,871
True 2D	59,280	28,553	118,560	118,093	199,871

Table 11: Statistics for each framework.

The *actual unknowns* are the number of nodes multiplied by the number of unknowns per node (3 for 3D, 2 for 2D).

The *effective unknowns* take into consideration the constraints placed on the system. For example, in the “Standard 3D” framework the vertical displacement of all nodes is fixed at zero. Therefore, the number of effective unknowns is about two thirds that of the actual unknowns.

Platform

All analyses were run on the redsky cluster using the same code version and on the same day. Disk access time in each run was negligible compared to the total run time. The hardware setup is a dual Intel Xeon X5570 board. The code is multithreaded and makes use of all of the 8 available cores.

³ <http://sourceforge.net/projects/aresfea/>

Metrics

The *analysis time* was computed by the walltime from the start of the simulation to the end of the 7th step.

The *memory footprint* was computed via the “top” command in Linux and represents actual memory usage—not an estimate.

The *output file size* is the file size required for each step of the analysis.

Results

The results were obtained after 7 nonlinear steps were performed which corresponded to a bulk plastic strain of about 3%. The equivalent plastic strain field can be seen in Figure 29. The results of all three simulations were the same to within a relative tolerance of 10^{-6} . One can see that the solution field is highly nonlinear.

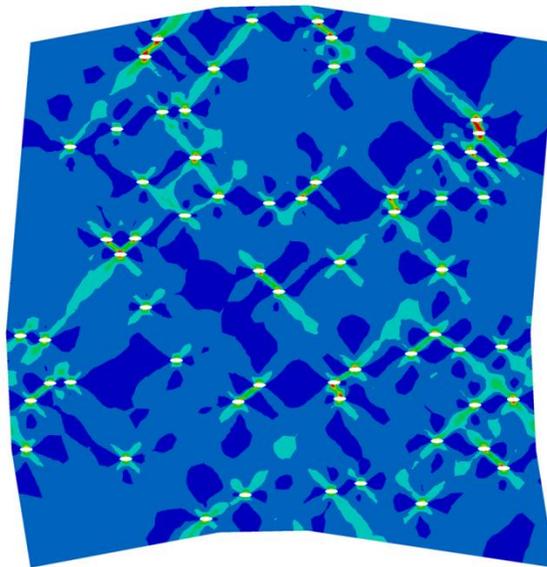


Figure 29: The equivalent plastic strain field.

The walltime and memory footprint were obtained after 7 nonlinear steps. Along with the output file size, these metrics are shown in Table 12.

Framework	Time (min)	Memory (MB)	File size (MB)
Standard 3D	145	1,260	47
Reduced 3D	51	780	47
True 2D	23	264	23

Table 12: Results for each framework.

The increase in computational efficiency between the “Standard 3D” framework and the “True 2D” framework is a factor of 6.3. In other words, a simulation which takes one week to

complete could be reduced to one day if the framework was changed. This is an incredible speed up in efficiency given that the results from the two simulations are equivalent.

The memory footprint also sees a considerably decrease by a factor of 4.7. While memory is typically not the barrier to higher resolution implicit simulations, this is still an impressive result.

The intermediate “Reduced 3D” framework delivered results somewhere between the two extremes. It should be noted that this framework and the “True 2D” framework solved the *exact* same set of equations and the differences were that the “Reduced 3D” framework holds a considerable number of “dummy” equations resulting from the reduction of unknowns due to analytical constraints. This increases the memory requirement for all components and so slows down the speed of the computation.

Timing profile

The walltime spent in each part of the code was reported in Table 13, Table 14, and Table 15. It is clear that in all three cases, the majority of time was spent within the linear solver. By comparison, formation of the stiffness matrix and load vector—reported within “System generation”—was relatively minor. Reduction of the system of equations by enforcing the analytical constraints—reported within “Constraints”—was relatively minor as well, even for the “Reduced 3D” framework which reduced the number of unknowns by more than two thirds.

=== Timing profile information ===			
Item	Hits	Percent	Absolute
Setup.....	3	0.27%	24.07 s
Constraints.....	133	1.15%	1.66 m
System generation.....	176	2.95%	4.27 m
Linear solver.....	37	94.67%	2.28 hr
Output.....	8	0.26%	22.95 s
Other.....	177	0.66%	58.05 s
Total.....	534	100.00%	2.41 hr

Table 13: Timing profile for the “Standard 3D” framework.

=== Timing profile information ===			
Item	Hits	Percent	Absolute
Setup.....	3	0.80%	24.79 s
Constraints.....	133	4.67%	2.39 m
System generation.....	176	4.09%	2.10 m
Linear solver.....	37	89.15%	45.63 m
Output.....	8	0.73%	22.52 s
Other.....	177	0.53%	16.32 s
Total.....	534	100.00%	51.18 m

Table 14: Timing profile for the “Reduced 3D” framework.

=== Timing profile information =====			
Item	Hits	Percent	Absolute
Setup.....	3	0.15%	2.22 s
Constraints.....	145	0.35%	5.05 s
System generation.....	192	8.62%	2.02 m
Linear solver.....	41	89.27%	20.89 m
Output.....	8	1.06%	15.00 s
Other.....	193	0.50%	7.16 s

Total.....	582	100.00%	23.40 m

Table 15: Timing profile for the “True 2D” framework.

Sample input deck

A sample Ares input deck for the “True 2D” framework is given in Table 16. The mesh file is not included.

```
# set up the 2D framework
>framework
plane_strain_planar

# define the material
>material
material, plasticisotropic, 160e9, 78e9, 8e3, hardening_curve,
strain_rate_dependence_curve
>data curve
hardening_curve, piecewise_linear
0, 200e6
1, 250e6
>data curve
strain_rate_dependence_curve, piecewise_linear
0, 1
1, 1.1

# include the mesh file
>include
"mesh_voids_planar.in"

# solution settings
>developer initial motion limit
0.10
>developer newton solver target tolerance
1e-6
>developer linear solver target tolerance
1e-7

# set termination time
>termination time
1

# set the initial timestep size
# (initial size) = (termination time) / (timesteps)
>timesteps
4000

# adjust the timestep so that the max eqps change
# in any element per timestep is at most X
>control timestep via material state change
bulk, "eqps", 0.05

# set the cubature type
>element block cubature
bulk, tri7p5o
```

```

# set minimal output
>output iteration steps
none
>output integration points
none
>output nodal forces
none

# enforce periodic boundaries to unit cell
>bc periodic translational direction
x, 1
y, 1

# define surfaces for load calculation
>assign surface id analytic
bottom, "X+Y<1 && X-Y<0"
bottom2, "X<0.5"

# assign node numbers to corner nodes
>assign node id
-1, 0, 0, 0
-2, 1, 0, 0
-3, 0, 1, 0

# set the unit cell displacements
>bc prescribe node displacement
-1, zero, x
-1, zero, y
-2, zero, y
-3, zero, x

# displace the right side
>analytic curve
displacement, "exp(sqrt(3)/2*t)-1"
>bc prescribe node displacement
-2, displacement, x

# output surface forces
>output surface force
bottom, x, "load.txt"
bottom2, x, "load2.txt"

# set output element type
>element block output
bulk, tri3, 1, 0

# output displacement
>output node value
-3, "x-X", "displacement.txt"

# output material maximums
>output material state value
bulk, eqps, maximum, "eqps_max.txt"
bulk, eqpsdot, maximum, "eqpsdot_max.txt"

```

Table 16: Ares input deck for the “True 2D” framework.

Best elements for large deformation plasticity

In this analysis, a number of element formulations were tested on a large deformation notched tensile specimen. The element formulations included the most basic standard displacement formulations as well as two nontrivial formulations—selective integration and selective deviatoric. The formulations have been ranked by accuracy and efficiency for both macroscopic and local quantities of interest.

All element formulations were shown to have convergent behavior, although the standard displacement formulations converged so slowly as to be of no practical use. For macroscopic variables—such as the load at a given displacement—typical errors ranged from around 10% for standard displacement formulations to around 1% for the more advanced formulations. For local variables—such as the triaxiality in the center at a given displacement—standard displacement formulations were off by a factor of 2 or more, while other formulations were off by approximately 1–10%.

The clear winners in terms of efficiency were the selectively integrated HEX8 and the selectively integrated TET10 formulations. Both of these elements have proven to be remarkably more accurate than standard formulations. In addition, neither requires the application of hourglass restoring forces. We propose implementing these two new selective integration element formulations into the SIERRA framework of codes.

The two proposed schemes are shown in Figure 30 and Figure 31, with the interpolation type shown on the (left), the shear cubature scheme shown in the (middle), and the volumetric cubature scheme shown on the (right).

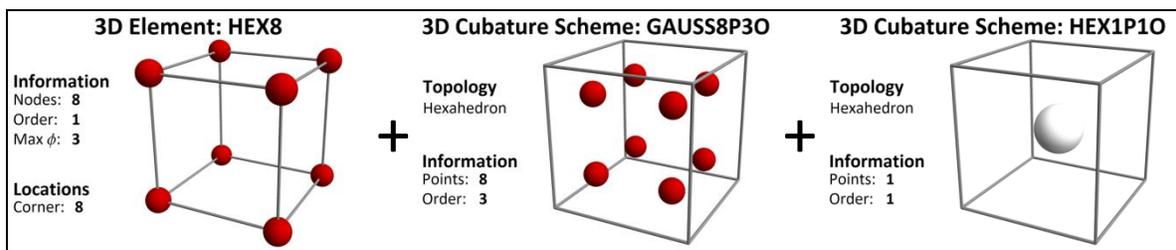


Figure 30: The proposed selectively integrated HEX8 element formulation.

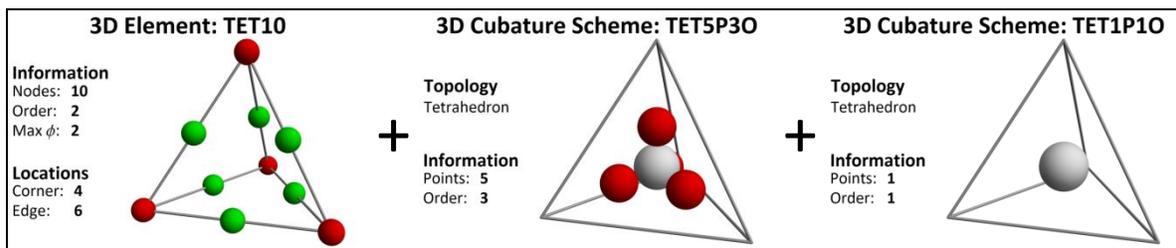


Figure 31: The proposed selectively integrated TET10 element formulation.

Background

Understanding the mechanical behavior of structural metals is of fundamental importance in the mission of Sandia to ensure the safety and reliability of the U.S. nuclear weapon stockpile. The capability to model the fracture of ductile metals is of vital importance to allow us to support this mission.

The finite element framework allows us to model a complex three-dimensional system by breaking it up into a number of elements. Mathematically, the laws of momentum balance are solved on these elements and the system of equations is assembled. Of critical importance in this procedure is the element formulation—both the interpolation scheme and integration scheme.

It is well known that the triaxiality field is the primary variable driving void growth and coalescence⁴. Therefore, it is important to accurately resolve the triaxiality field if one is interested in predicting failure of this type. Traditional standard displacement elements have been shown to do a poor job of resolving the pressure field in the context of large deformation plasticity. The popular underintegrated elements have a fundamental drawback in that the application of hourglass restoring forces, necessary to eliminate zero energy modes, is not uniquely defined. The hourglass force application varies between codes and can influence the results by several percent or more.

Since the mathematical formulation of the finite element method is a well-defined problem, a unique solution—given an appropriate material model—exists. The variation in the solution as a result of the hourglass formulation is a non-physical and unwelcome artifact.

Due to these drawbacks of the current element implementations, we wish to look at other formulations to more accurately capture the dynamics of the system and help us predict failure in ductile metals.

Mathematical foundation

Following from Newton’s second law of motion, the local balance of linear momentum in continuum mechanical is simply written as

$$\sigma_{ij,j} = \rho \ddot{u}_i \quad (46)$$

where σ is the Cauchy stress tensor, ρ is the density, and \mathbf{u} is the displacement. To transform this equation into a form useful for finite element calculations, we integrate over the body and introduce shape functions to arrive at the following form for the nodal force vector. The result of this procedure is

$$\underbrace{\int_{\Omega} \rho \phi_I b_i d\Omega}_{\text{body forces}} + \underbrace{\int_{\partial\Omega} \phi_I t_i da}_{\text{surface forces}} - \underbrace{\int_{\Omega} \phi_{I,j} \sigma_{ij} d\Omega}_{\text{internal forces}} = \ddot{u}_{jI} \underbrace{\int_{\Omega} \rho \phi_I \phi_J d\Omega}_{\text{momentum change}} \quad (47)$$

where Ω is the body, ϕ_I is the shape function of node I , b_i is the body force, $\partial\Omega$ is the surface of the body, t_i is the applied traction on that surface, $\phi_{I,j} = d\phi_I / dx_j$, and \ddot{u}_{jI} is the acceleration

⁴ F. McClintock, “A criterion for ductile fracture by the growth of holes,” *Journal of Applied Mechanics*, 1968.

of node J . Here, lower-case indices sum over the 3 Cartesian coordinates, and upper-case indices sum over each node.

The most significant of these terms is typically the contribution due to internal forces,

$$f_{ii} \stackrel{\text{def}}{=} \int_{\Omega} \phi_{I,j} \sigma_{ij} d\Omega \quad (48)$$

which must be evaluated using some approximation. The most basic of these is the standard displacement formulation, which approximates the integral by evaluation it at discrete integration points within the domain and weighting each contribution according to a given scheme—such as Gaussian quadrature.

We will look at the shortcomings of this approach and evaluate other methods.

Standard displacement formulation

In the standard displacement model, the displacement within the element is interpolated from the nodal displacements and the shape functions. Within each element, it is formed by the sum

$$u_i = \sum_{I=1}^N \phi_I a_{ii} \quad (49)$$

where N is the number of nodes, ϕ_I is the shape function corresponding to node I , and a_{ii} is the displacement of node I in the direction of \mathbf{e}_i . This allows us to evaluate the integrand of equation (48) at any point within the element.

We then approximate the integral by evaluating it at a number of integration points. These are often locations obtained from extending Gaussian quadrature into multiple dimensions, but this does not have to be the case. For this reason, these points are often called “Gauss points”. Equation (48) is then approximated as

$$\int_{\Omega} \phi_{I,j} \sigma_{ij} d\Omega \approx \sum_{c=1}^C w_c \phi_{I,j} \sigma_{ij} d\Omega \Big|_{\mathbf{x}_c} \quad (50)$$

where C is the number of integration points and w_c is the weight associated with the integration point located at \mathbf{x}_c .

For the naming convention, we simply list the interpolation and integration schemes. For example, the HEX8+GAUSS8P3O formulation uses the 8 node hexahedron for the shape functions and evaluates the integral using an 8 point, 3rd order Gaussian scheme.

Theoretical foundation

In this section we discuss the mathematics of the selective integration and the selective deviatoric formulations.

Selective integration formulation

In the selective integration formulation, the nodal force contributions from internal forces are separated into deviatoric and volumetric parts by using the decomposition of the stress

$$\sigma_{ij} = \sigma'_{ij} - p\delta_{ij} \quad (51)$$

$$p \stackrel{\text{def}}{=} -\frac{1}{3} \sigma_{kk} \quad (52)$$

where p is the pressure and σ'_{ij} is the deviatoric stress. Accordingly, we break the internal force integral into

$$\int_{\Omega} \phi_{I,j} \sigma_{ij} d\Omega = \underbrace{\int_{\Omega} \phi_{I,j} \sigma'_{ij} d\Omega}_{\text{deviatoric part}} + \underbrace{\int_{\Omega} \phi_{I,i} \frac{1}{3} \sigma_{kk} d\Omega}_{\text{volumetric part}} \quad (53)$$

where the two integrals are evaluated using separate integration schemes similar to equation (50). The volumetric integration scheme is typically of lower order than the deviatoric scheme.

Complications of this formulation exist in post-processing field data. Since we have two types of integration points, the analyst must be aware of this and choose the appropriate set. For example, the equivalent stress field is (typically) a function primarily of the deviatoric deformation, so the deviatoric cubature points would be an appropriate choice. The volumetric cubature points could also be used, although the resolution would not be as high since there are less of them. Similarly, the pressure is (typically) a function primarily of the volumetric deformation, so the volumetric cubature points would be the appropriate choice. We make no attempts at combining information from the two sets of fields.

Computationally, this formulation preserves the symmetry of the stiffness matrix. It has a cost only slightly higher than the standard displacement scheme due to the addition of some volumetric cubature points. Since the decomposition of the stress can be done after a call to the material routine, implementation of this formulation requires no modification to the material subroutine calls which calculate the stress.

For the naming convention, the scheme used to evaluate the deviatoric part is listed first. For example, a HEX8SI+GAUSS8P3O_HEX1P1O formulation uses the 8 node hexahedron element for the shape functions, and evaluates the deviatoric part using an 8 point, 3rd order Gaussian scheme, and the volumetric part using a single point scheme.

Selective deviatoric formulation

In the selective deviatoric formulation, we have a similar change to the internal force calculation. Before the evaluation of the stress takes place in the material subroutine, we pass

$$\mathbf{F}^{\text{sd}} \stackrel{\text{def}}{=} J_{\text{avg}}^{1/3} J^{-1/3} \mathbf{F} \quad (54)$$

instead of the local \mathbf{F} , where J_{avg} is the average Jacobian of all cubature points within the element.

This formulation can be extended to, for example, linearly interpolate J within the element, however these extended formulations have not proven as effective as the simple averaging operation.

Since

Computationally, the cost can be significantly higher than the standard displacement formulation. Since the stress at a given cubature point is a function of the deformation at *all*

cubature points (on account of averaging J) and not just the local deformation, calculating the stiffness matrix requires significantly more effort. This increase in cost grows as the number of cubature points increases. Additionally, the symmetry of the stiffness matrix is not preserved in this formulation. Although this effect is small, this requires a linear solver capable of solving non-symmetric equations. These solvers are considerably slower than their counterparts for symmetric systems.

Unlike the selective integration technique, the selective deviatoric technique has only a single set of cubature points and so there is no ambiguity associated with post-processing a given field.

For the naming convention, we simply append SD to the interpolation scheme to denote the use of the selective deviatoric technique—e.g. HEX8SD+GAUSS8P3O.

A note on selective techniques

The use of selective techniques to evaluate the internal forces is not a new concept and has been widely used. However, typical application has been limited to small strains and/or the assumption of hypoelasticity. The research in this report does not use either of these simplifying assumptions.

The 8 noded underintegrated hexahedron

The underintegrated hex has seen widespread use due to its computational efficiency. In this formulation, the internal force is approximated as

$$\int_{\Omega} \phi_{i,j} \sigma_{ij} d\Omega \approx \left(|\Omega| \phi_{i,j} \sigma_{ij} d\Omega \right)_{\text{element center}} + (\text{hourglass restoring forces}) \quad (55)$$

where the integrand is evaluated at a single central cubature point, and hourglass restoring forces are prescribed to prevent zero-energy hourglass modes from becoming significant.

This scheme gains some benefits of the selective integration technique as the integral is only evaluated at a central cubature point. This also allows the element to be several times faster than its fully integrated counterpart—although this computational savings is typically negligible in implicit simulations.

The difficulty in this scheme is the prescription of the hourglass restoring forces. A large amount of effort has been done to tune the hourglass settings based on the type of analysis. This is still an open question as evidenced by the proprietary nature of how each of the different commercial codes handles this situation.

The selective integration technique avoids the application of hourglass forces entirely as there are no spurious zero energy modes to correct.

Modeling procedure

This analysis was performed using Ares⁵—a three dimensional, implicit, mechanical finite element software. Meshes were created using Cubit⁶, and the results were viewed using ParaView⁷.

⁵ <http://sourceforge.net/projects/aresfea/>

Notched tensile specimen

We chose the notched tensile geometry as the model problem of interest as it possesses many features present in larger scale analyses, yet is small enough to allow for a high fidelity mesh convergence study. Among these features are:

- a simple geometry suitable for both HEX and TET meshing
- a geometry free of cracks and other sources of infinite stress concentrations
- under loading, a nontrivial transition from an elastic region to a large deformation plasticity region
- under loading, smooth stress and triaxiality fields
- suitable global and local metrics to assess convergence

The geometry of the notched tensile specimen is shown in Figure 32.

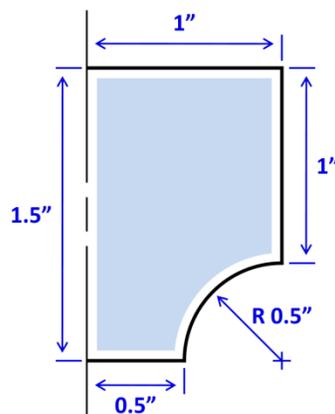


Figure 32: Geometry of the cylindrical notched tensile specimen. The specimen is axisymmetric about the left edge and symmetric across the bottom edge.

Material models

Our material model allows a yield stress of the form

$$\sigma_y = \sigma_y^0 + K \left(\varepsilon_{p,eq}^0 + \varepsilon_{p,eq} \right)^n \quad (56)$$

which contains linear hardening and power law hardening as specific cases. We have obtained compression data from B. Antoun (8246) for a compression specimen of stainless steel 304L up to a logarithmic compressive strain of around 0.7. These data points have been used to fit three curves conforming to (56). The material parameters resulting from this fit are given in Table 17. Since this specimen was annealed, we have chosen to fix the initial plastic strain at zero while other parameters have been fit using a least squares approach. The strain at necking parameter is both approximate and derived from the other parameters and is therefore shown in italics.

⁶ <http://cubit.sandia.gov/>

⁷ <http://www.paraview.org/>

Parameter	Variable	Units	Combined	Power Law	Linear
initial yield stress	σ_y^0	MPa	132.157	0	269.407
hardening modulus	K	MPa	1494.71	1547.92	1541.74
hardening exponent	n	(none)	0.719760	0.575326	1
initial plastic strain	$\epsilon_{p,eq}$	(none)	0	0	0
<i>strain at necking</i>	ϵ_{neck}	(none)	<i>0.641680</i>	<i>0.575326</i>	<i>0.916438</i>

Table 17: Material parameters for stainless steel 304L.

The fit of each of these hardening models compared to the experimental data is shown in Figure 33. Over the given range, all three models behave similarly, with variations of approximately 5% on most data points. At the first data point, the curves vary more due to the difference in initial yield stress and initial hardening slope between each model.

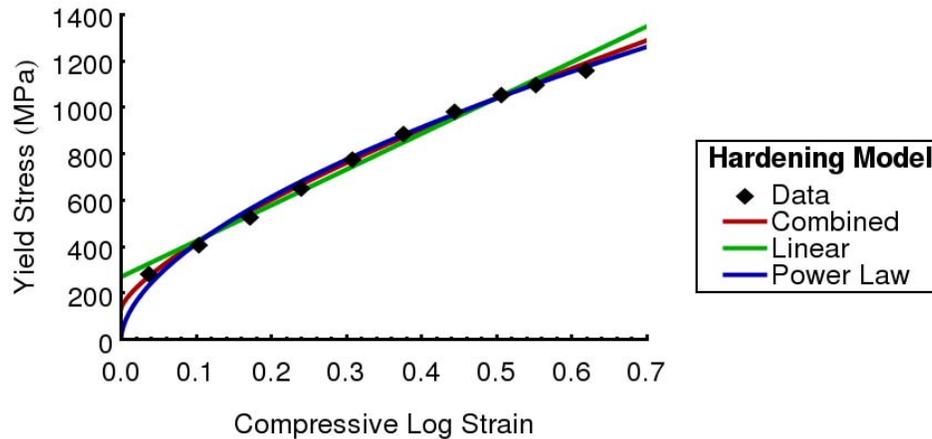


Figure 33: Yield stress curve fits to experimental stainless steel 304L data.

For results in this report, we will use the “Combined” model since it provides the best fit to experimental data. The other two fits are given as they are useful for comparing with published data.

Boundary conditions

On symmetry planes, the out-of-plane displacement is constrained to be zero.

To load the specimen, the vertical displacement of all nodes on the outer surface is ramped linearly from zero to 0.2” over 100 equal steps.

Mesh refinement levels

The notched tensile specimen geometry has been meshed using a parameterized Cubit file to generate both HEX and TET meshes at a variety of refinement levels. Taking advantage of symmetry, only one eighth of the geometry is modeled, and appropriate boundary conditions are applied along the symmetry planes.

Successive refinement levels for the HEX8 mesh can be seen in Figure 34, with the “x2” refinement on the left, the “x4” refinement (middle), and the “x8” refinement (right). Finer meshes are not pictured. The corresponding HEX20 meshes (not pictured) look similar to the HEX8 meshes, but have curved faces which better represent the specimen geometry.

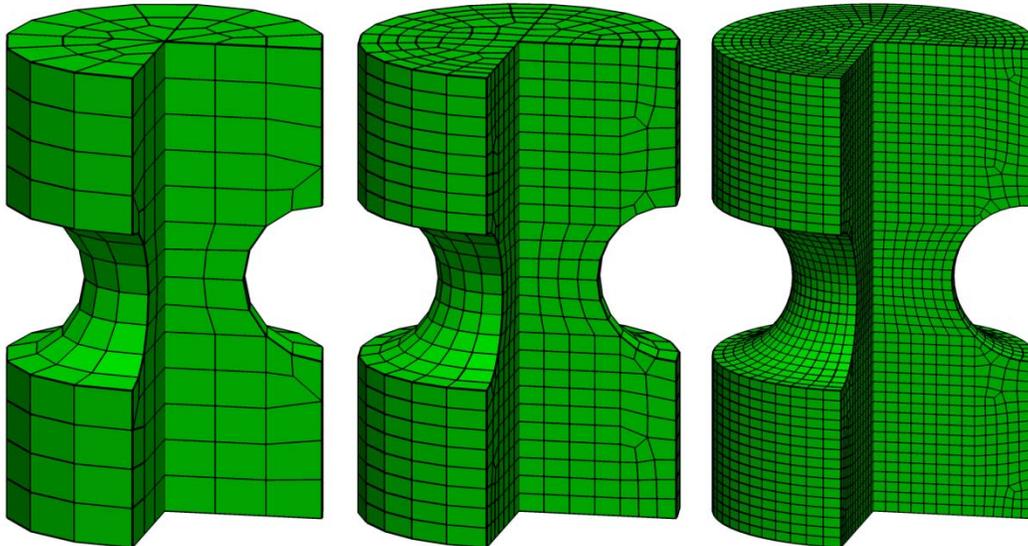


Figure 34: Successive refinements of the HEX8 notched tensile mesh. Refinement levels are “x2” (left), “x4” (middle), and “x8” (right). The “x16” and “x32” refinements are not pictured.

Successive refinement levels for the TET mesh can be seen in Figure 34, with the “x2” refinement on the left, the x4 refinement (middle), and the “x8” refinement (right). Finer meshes are not pictured. The corresponding TET10 meshes (not pictured) look similar to the TET4 meshes, but have curved faces which better represent the specimen geometry.

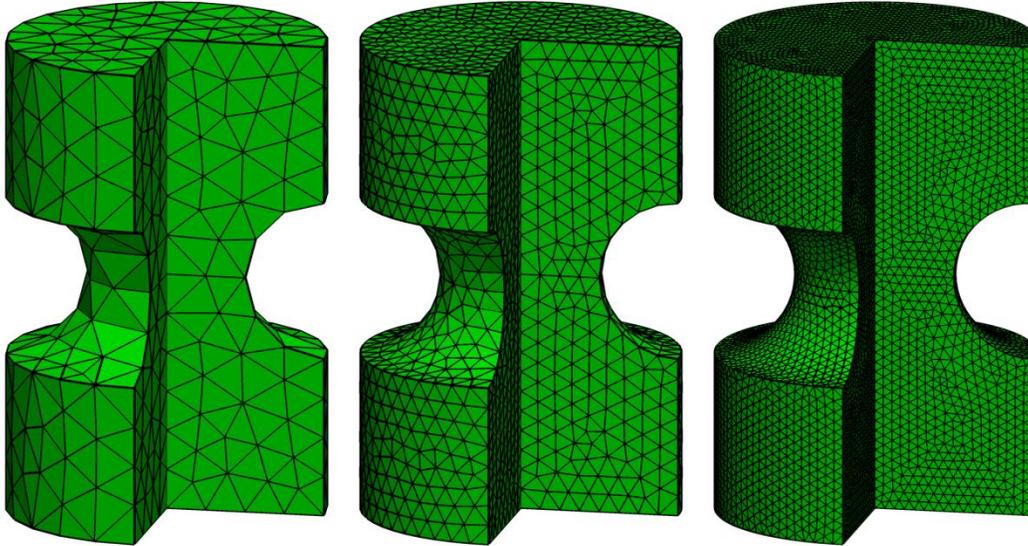


Figure 35: Successive refinements of the TET notched tensile mesh. Refinement levels are “x2” (left), “x4” (middle), and “x8” (right). The “x16” and “x32” refinements are not pictured.

The number of nodes and elements for each mesh discretization is given in Table 18.

Refinement Level	Nodes				Elements	
	HEX8	HEX20	TET4	TET10	HEX8/HEX20	TET4/TET10
x2	123	421	148	908	66	513
x4	469	1,690	922	3,288	307	4,120
x8	3,220	12,216	5,528	40,636	2,602	27,869
x16	23,115	90,003	34,504	262,468	20,748	186,580
x32	162,930	642,536	190,927	1,475,821	153,922	1,066,362

Table 18: Number of nodes and elements in each mesh.

Element formulations

While there are an incredibly large number of element formulations available within Ares, we have chosen to focus on the standard first and second order HEX and TET elements, including some variations. As mentioned earlier, we denote each scheme with the naming convention INTERPOLATION+CUBATURE.

Interpolation schemes

The list of interpolation schemes used in this analysis is given in Table 19.

Interpolation	Order	Nodes	Notes
HEX8	1	8	standard 8 node linear hexahedron element
HEX8SI	1	8	hex8 with selective integration of the bulk and deviatoric parts
HEX8SD	1	8	hex8 with averaging of the bulk response among all integration points
TET4	1	4	standard 4 node linear tetrahedron element
HEX20	2	20	quadratic “serendipity” hex element
TET10	2	10	standard quadratic tetrahedron element

Table 19: Summary of the interpolation schemes used.

Cubature schemes

The list of cubature schemes used in this analysis is given in Table 20. Note that only some combinations of interpolation and cubature schemes make sense.

Cubature	Order	Points	Notes
GAUSS8P3O	3	8	extended 2 point Gaussian quadrature scheme
GAUSS8P3O_GAUSS1P1O	3 / 1	8 + 1	8 point deviatoric scheme + 1 point bulk scheme
HEX14P5O	5	14	5 th order hex scheme with 14 points
TET4P2O	2	4	2 nd order tet scheme with 4 points
TET11P4O	4	11	4 th order tet scheme with 11 points

Table 20: Summary of the cubature schemes used.

A cubature scheme with an order twice the order of the element is typically required to “fully” integrate the element. In a hyperelastic material, very little effect is seen from increasing the accuracy of the scheme beyond this point. In a plastic material, since the material response is non-smooth, the cubature scheme has a much bigger effect on the results. In some scenarios, a more accurate response can be obtained by under integrating (e.g. HEX20+GAUSS8P3O), although this technique is often not without drawbacks (e.g. hourglass control requirement on HEX8+GAUSS1P1O elements).

Metrics of interest

In order to assess the performance of each element formulation, we look at two metrics of interest. For both of these, the error is calculated as

$$error = \ln \left(\frac{h^{approx}}{h^{exact}} \right) \quad (57)$$

where h^{approx} is the approximate solution, and h^{exact} is the “exact” solution as obtained from a Richardson extrapolation technique.

Load at displacement

For one metric of interest, we have measured the load after a displacement of 0.2 inches. The “exact” solution was obtained using Richardson extrapolation on the two finest HEX8SI

simulations and is expected to be accurate to over 3 decimal places. We have obtained this by summing the residual forces in the axial direction on the top face of the specimen.

This metric is a “global” quantity in the sense that local element variations should not significantly affect the result. Therefore, we expect convergence behavior to be well behaved.

Maximum triaxiality

The second metric of interest is the maximum triaxiality in the specimen after a displacement of 0.2 inches. The triaxiality is defined as

$$T = \frac{\overset{\text{def}}{\sigma_m}}{\sigma_{\text{eq}}} = \frac{\frac{1}{3} \sigma_{kk}}{\sqrt{\frac{3}{2} \sigma_{ij} \sigma_{ij}}} \quad (58)$$

where σ_m is the mean stress, and σ_{eq} is the equivalent (i.e. von Mises) stress. Since the triaxiality field is not defined pointwise within the specimen, we approximate this metric as the maximum value of all cubature points.

This metric is a “local” quantity in the sense that local element variations can greatly affect the results. The convergence behavior is expected to be

Visualizing element fields

In order to visualize an element field, we must process information at the integration points in some manner to produce a piecewise defined field. This is not a trivial task, in general. For some element formulations, the process is straightforward. For example, for elements with a single integration point one may either assume the field is constant within each element, or one may obtain nodal values through by taking a simple average of the neighboring elements and then use the shape functions to interpolate the field.

In general, there is not a straightforward manner to do this construction. In Figure 36, we show the results of such a transformation. On the left picture, the value of the triaxiality field at each cubature point is represented with a sphere. The construction of the pointwise defined field is shown on the right.

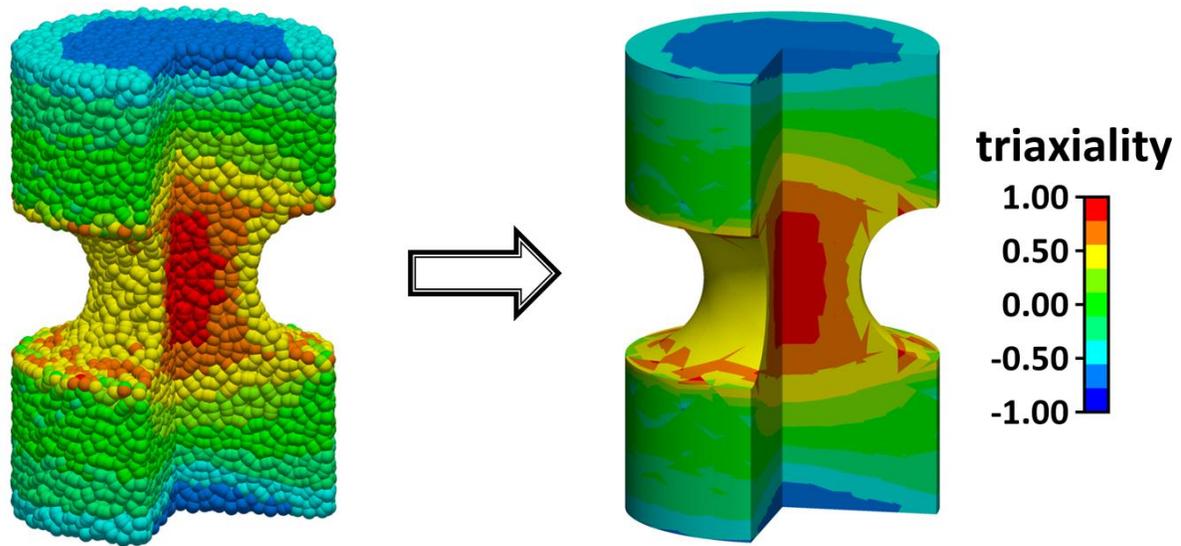


Figure 36: Construction of a pointwise defined field (right) from discrete values at cubature points (left). The value at each cubature point is shown as a sphere on the left-hand picture.

The construction shown in Figure 36 was done on an element-by-element basis. Within each element, a least-squares fit was done to a first order polynomial basis (4 shape functions). Since continuity of the field is not enforced across element boundaries, the resulting field can be discontinuous. In Figure 37, we have shown the construction of the field on a single element.

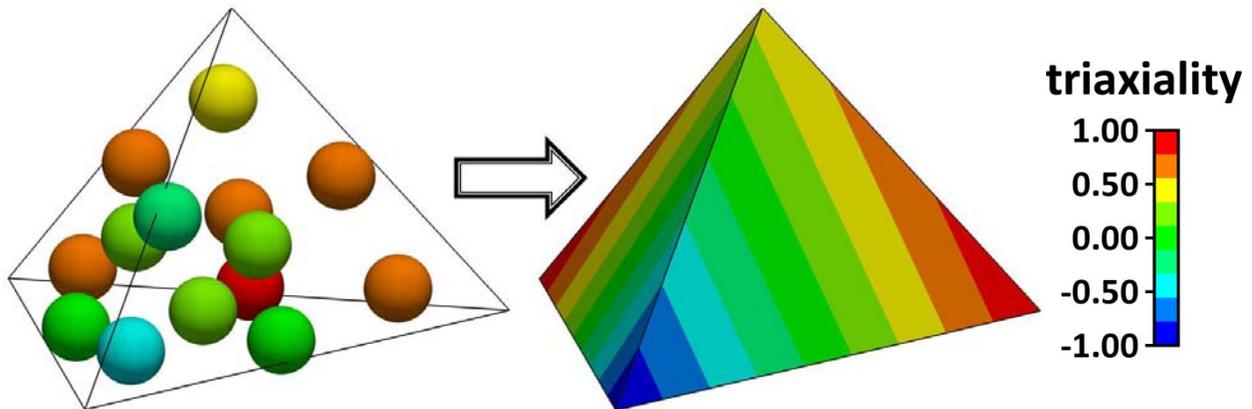


Figure 37: A per-element construction of a pointwise defined field (right) from discrete values at cubature points (left). The construction was done through a least squares fit to a linear polynomial basis.

This construction is not without drawbacks. The constructed field can have a maximum/minimum which is more extreme than the maximum/minimum of all cubature points. For fields which have physical bounds on their range (e.g. absolute temperature must be positive), this can lead to inconsistencies the analyst must address.

Results

We have broken up the results into the response under infinitesimal loading, and the response under the large deformation plastic loading.

Fields after infinitesimal loading

In a separate simulation, we have loaded the specimen infinitesimally to obtain the shape of each field. Three field of interest are shown in Figure 38—equivalent stress, pressure, and triaxiality. The first two fields have been normalized by their maximum value. The triaxiality field is unitless and therefore not normalized (although the maximum value of 0.94 is so close to 1 that it may look this way).

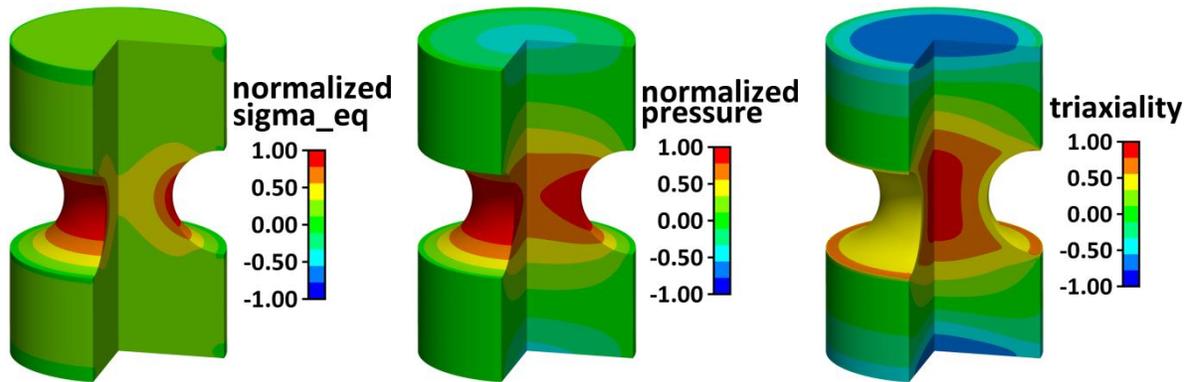


Figure 38: The normalized equivalent stress (left), normalized pressure (center), and triaxiality (right) fields for an infinitesimal loading of the notched geometry. The maximum triaxiality of approximately 0.94 occurs at the center of the geometry.

Under an infinitesimal loading, all element formulations performed relatively well. In Figure 39 and Figure 40 the triaxiality field is shown for a series of refinement from coarsest (left) to finest (right) for the standard HEX20 and TET4 element formulations, respectively. As can be seen, both formulations appear to converge to the same solution, although the HEX20 formulation converges significantly faster.

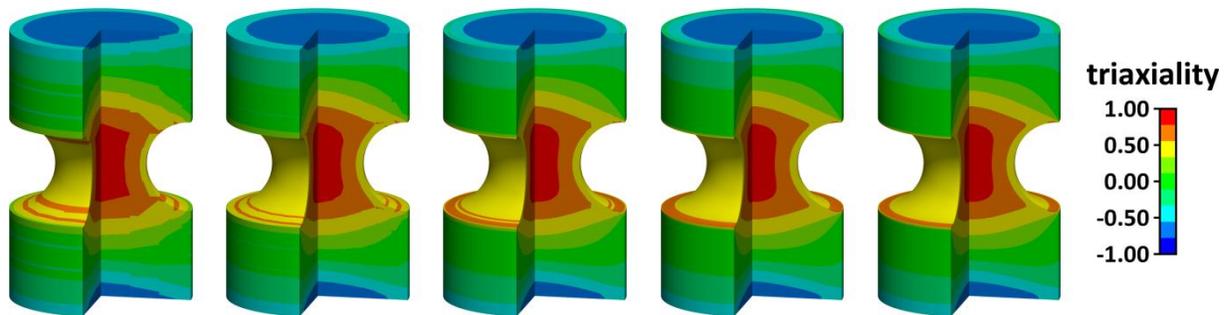


Figure 39: The triaxiality field under an infinitesimal loading for the HEX20 formulation under successive refinements, from “x2” (left) or “x32” (right).

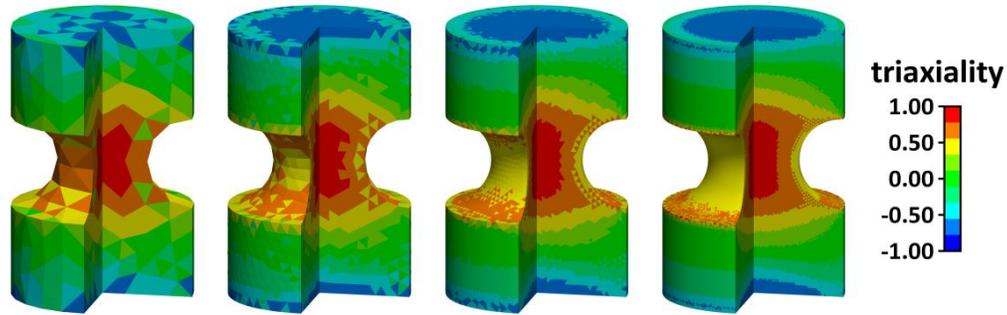


Figure 40: The triaxiality field under an infinitesimal loading for the TET4 formulation under successive refinements, from “x2” (left) or “x16” (right).

In these figures, the triaxiality field was chosen to plot because it behaves the worst out of the fields of interest. The other fields (equivalent stress, pressure) are much better behaved in general.

Fields after full loading

In Figure 41, the equivalent plastic strain, normalized pressure, and triaxiality fields are shown after the material has undergone the full deformation for the HEX8SI formulation. The plastic strain in the center of the specimen is seen to be about 0.45 and reaches a maximum of around 0.65 on the inside of the notch. Both the pressure and triaxiality fields have changed significantly from their relative values under infinitesimal loading. Among other things, this suggests that one cannot assume the fields under elastic loading are representative of the same fields after plastic deformation, even though the loading is monotonic.

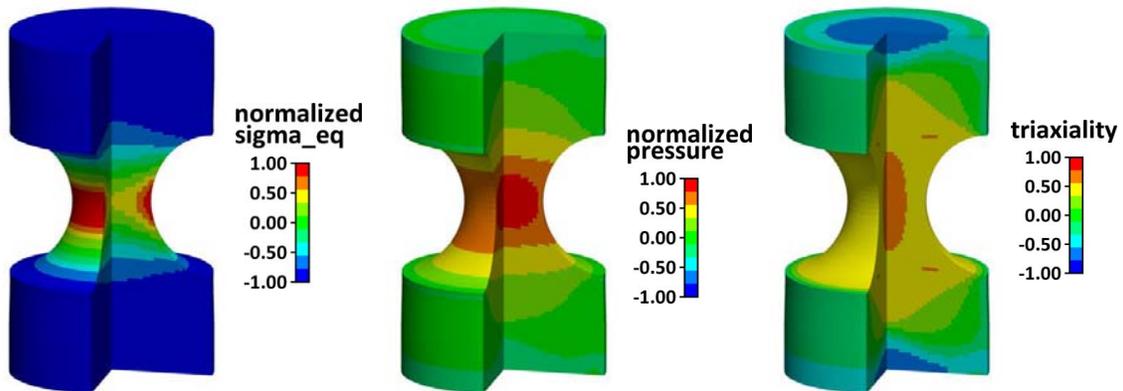


Figure 41: The equivalent plastic strain (left), normalized pressure (center), and triaxiality (right) fields after a loading of 0.2 inches of the notched geometry for the HEX8SI element formulation.

In general, fields from other element formulations did not look nearly as nice as the pictures in Figure 41. We will explore these artifacts and the reasons behind them.

Evolution of the triaxiality field

The evolution of the triaxiality field throughout the simulation follows a nontrivial path. In Figure 42 we have plotted the field at five equally spaced intervals along the load path. Its state at a small plastic deformation (left) closely resembles that of the elastic solution from Figure 38. After this, the point of maximum triaxiality initially at the center of the specimen separates axially into two local maxima, after which the field again evolves to have a single global maximum at the center of the specimen.

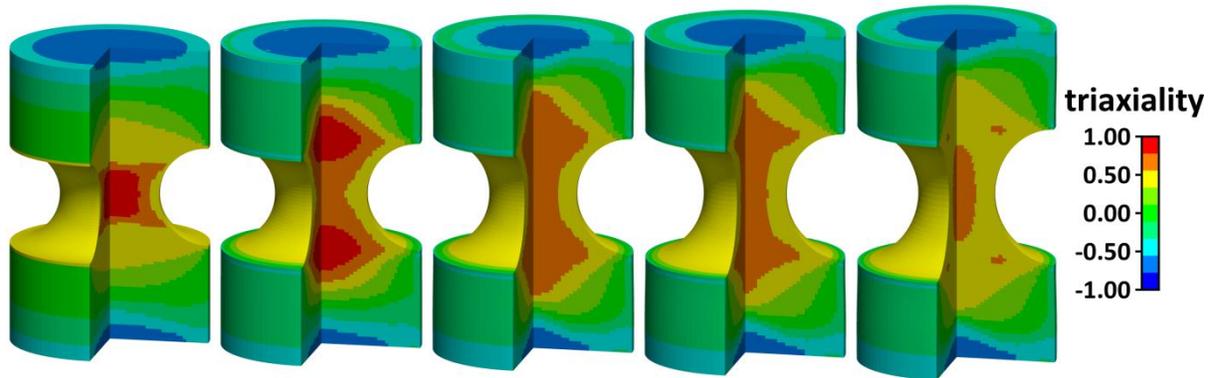


Figure 42: The evolution of the triaxiality field from the beginning of plastic deformation (left) to the end of the simulation (right).

Oscillations in pressure field

In all standard element formulations, the pressure field was found to oscillate significantly. We have plotted the value of this field at all cubature points in Figure 43. One can see significant clipping of the field in all subfigures, especially for HEX8 and TET4. By comparison, the quadratic formulations did not suffer nearly as much, although checkerboard-type oscillations can still be seen.

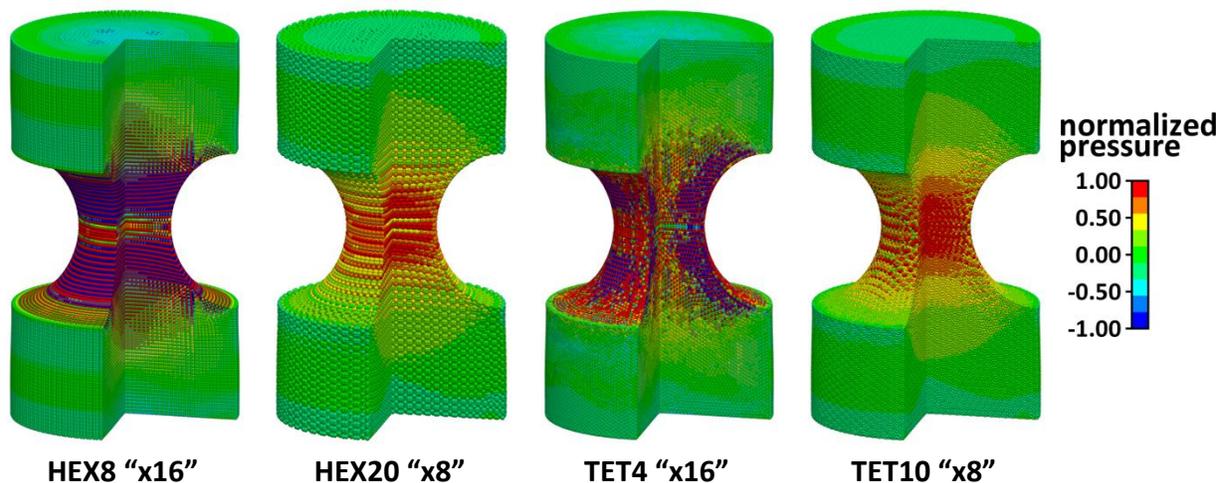


Figure 43: The pressure field for a variety of standard displacement formulations. Note that there is considerable clipping in the fields in all cases, especially for the HEX8 and TET4 cases.

For comparison, the pressure field for the selective integration and selective deviatoric schemes are plotted in Figure 44. None of these figures show oscillations.

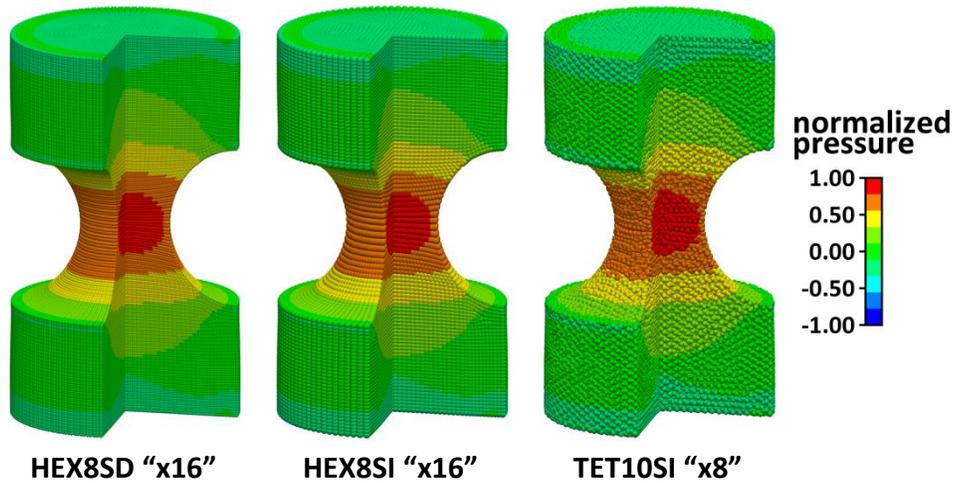


Figure 44: The pressure field for three selective element formulations.

Mitigation by averaging

In transferring cubature point information into element-based data, one can mitigate the appearance of these oscillations by averaging the integration point values in the element. However, doing so does not always produce accurate or good looking results. We note that such an averaging scheme is not equivalent to the HEX8SI formulation, which takes its triaxiality value from the central integration point. The difference between these two is shown in Figure 45. Even though the meshes are identical, the resulting fields have significant differences. The most apparent of these differences are the checkerboard-type oscillations in the averaged HEX8 field. One can see significant variations in the field in adjacent elements. This behavior is not expected in the true solution, and it is not seen in the HEX8SI field. These differences are not apparent under elastic loading, but become increasingly significant as the body is loaded plastically.

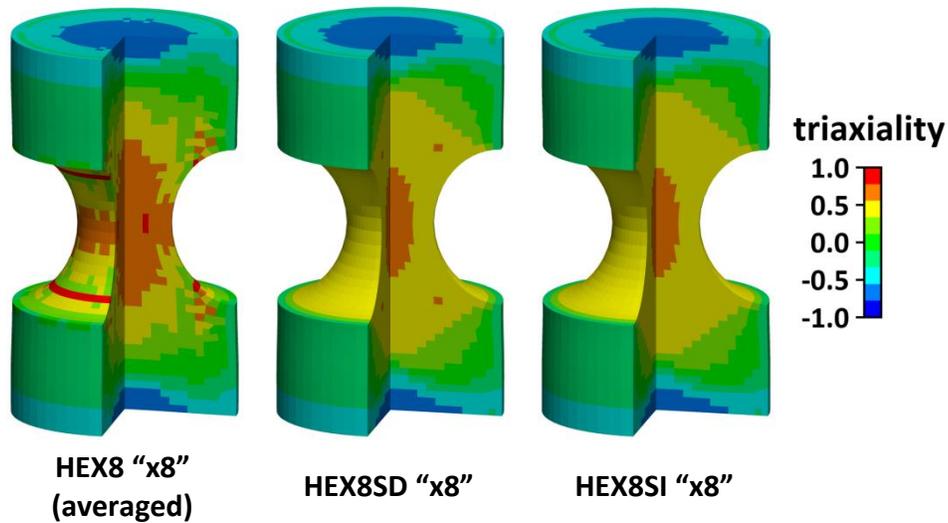


Figure 45: The triaxiality field obtained through averaging the integration point values in the HEX8 formulation (left), the HEX8SD formulation (middle), and the HEX8SI formulation (right).

Selective formulations of the HEX20 element

Although it is possible to run the HEX20SD+HEX14P5O formulation, the associated stiffness matrix was so poorly conditioned that the code could make little if any progress on convergence. Whereas most simulations converged within about 15 Newton-like iterations, this formulation took over 100 iterations and still did not meet the target 10^{-10} relative residual. The HEX20SI+HEX14P5O_HEX1P1O formulation suffered similar problems.

To see if any successful formulations could be made from the HEX20 element, we tried both the HEX20SI+RIEMANN27P1O_HEX1P1O and the HEX20SI+RIEMANN27P1O_RIEMANN8P formulations. Cubature schemes named RIEMANN have points spaced evenly across the domain with equal weights. Although these ran, the results show significant element-based artifacts. As shown in Figure 46, ridges formed at element boundaries on the external surface. These artifacts were reduced significantly with element refinement. However, their presence at any level of mesh discretization does not inspire confidence in the results.

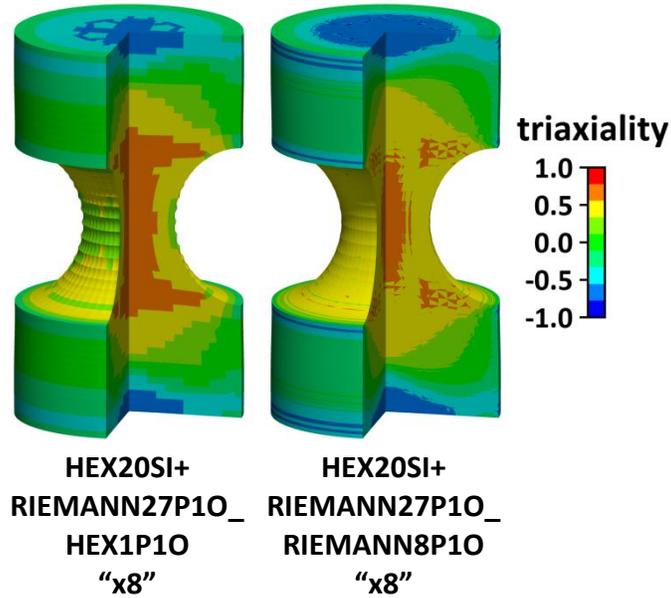


Figure 46: Values of the triaxiality at bulk cubature points for two HEX20SI formulations. Note the formation of ridges along the notch surface.

Typical timing profile

To gain some understanding of the computational cost of an implicit simulation, it is helpful to look at the timing profile. After each timestep, Ares outputs this information separated into the various parts of the calculation. The timing profile for the “x4” mesh of the TET10 mesh is shown in Table 21.

=== Timing profile information =====			
Item	Hits	Percent	Absolute
Setup.....	3	0.00%	8.22 s
Reading keywords	1	0.00%	1.61 s
Processing	1	0.00%	1.96 s
Initializing	1	0.00%	4.65 s
Constraints.....	1504	0.69%	1.60 hr
Initializing	502	0.36%	49.92 m
Static	1	0.00%	1.67 s
Dynamic	501	0.36%	49.89 m
Application.	501	0.33%	45.79 m
Evaluation	501	0.00%	400.00 ms
System generation.....	1002	4.85%	11.14 hr
Internal forces	501	4.85%	11.14 hr
Tractions	501	0.00%	0 s
Linear solver.....	421	93.74%	8.97 dy
DPCG	421	93.74%	8.97 dy
Output.....	501	0.32%	45.24 m
Other.....	2425	0.37%	52.28 m
Total.....	5856	100.00%	9.57 dy

Table 21: Timing profile information for the TET10 x4 refinement run. Note that the vast majority of the time is spent within the linear solver.

As can be seen, over 90% of the computational time is spent in the linear solver, in this case a diagonally preconditioned conjugate gradient (DPCG) solver. The next biggest contribution comes from generation of the system of linear equations (stiffness matrix, load vector) from internal element force calculations.

It is reassuring to note that the computation of the analytical constraints placed on the system of unknowns, including the generation of each linear equation, the Gaussian elimination to reduce redundant constraints, and the modification of the stiffness matrix based on the resulting system occupied less than 0.7% of the total calculation. If we were to use a penalty formulation or a Lagrange multiplier approach, this calculation would have been much more costly.

All simulations were performed on the same machine with the same version of the code and the associated runtimes are accurate to the extent possible. Of course, due to the complexity of today's CPUs, the runtime is by no means constant for a given simulation and has been seen to vary by 20% or so for no transparent reasons. In the results communicated, a difference of 20% in runtime has a negligible effect.

Load at displacement error

In Figure 47, the error in the load is plotted versus the element edge. Light gray contour lines are drawn to show the expected linear convergence behavior. It is reassuring that all element formulations converge at approximately a linear rate. Typical errors are around 1% for most formulations, but considerably worse (10–100%) for the standard fully integrated TET4 and HEX8 formulations. Since the cost of using an element is a function of more things than simply the element edge length, we need to look at another plot to examine the relative efficiency of the different formulations.

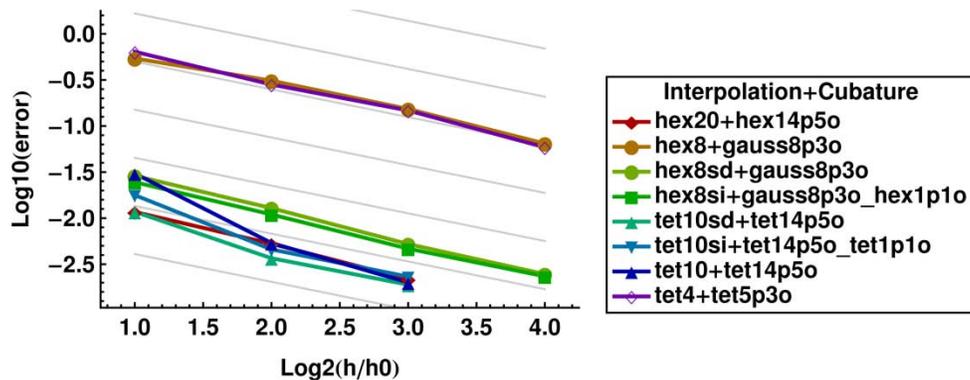


Figure 47: Error in load versus element edge length. Light gray contour lines correspond to the expected linear convergence behavior.

In Figure 48, the error is plotted versus the simulation walltime. Since Ares is a serial (i.e. single thread) code, this is simply the physical time between the start and end of the simulation. This plot allows us to effectively rank the element formulations at solving this particular problem by ranking curves from the bottom left (best) to the top right (worst). We notice the standard fully integrated HEX8 and TET4 elements are significantly worse than all other formulations. The HEX20+HEX14P50 formulation performs the best, with the HEX8SI formulation a close second.

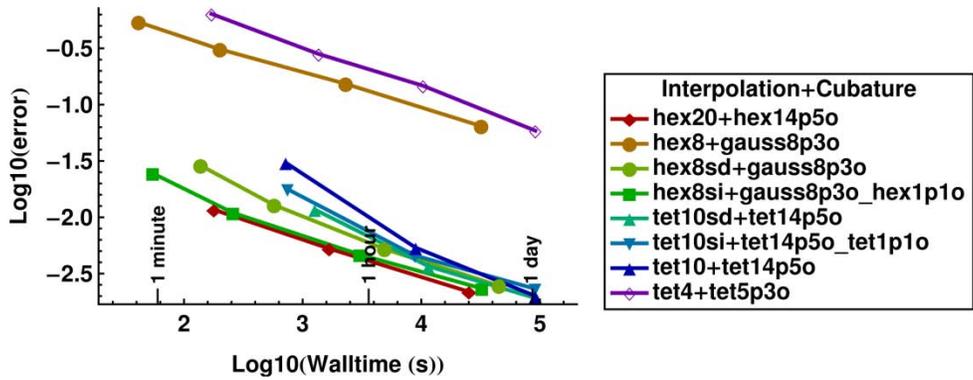


Figure 48: Error in load versus simulation walltime.

Triaxiality field

In Figure 49, the error in the maximum triaxiality metric is plotted versus element edge length. Unlike the results for the forging load, these results do not converge linearly in general. Five of the eight element formulations had an unacceptably high error. The three which had errors under 1% were the HEX8SD, HEX8SI, and TET10SI formulations.

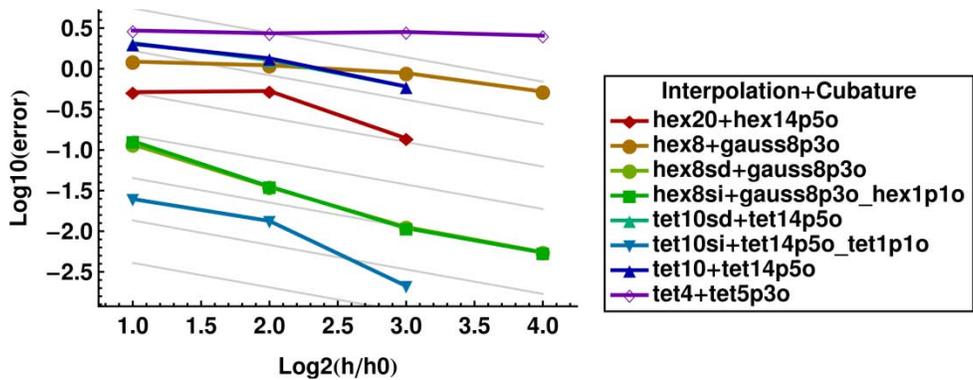


Figure 49: Error in maximum triaxiality versus element edge length. Light gray contour lines correspond to linear convergence behavior.

In Figure 50, the error is plotted versus the simulation walltime. One can see three best formulations are approximately equal in efficiency, with the HEX8SI formulation slightly better than the others.

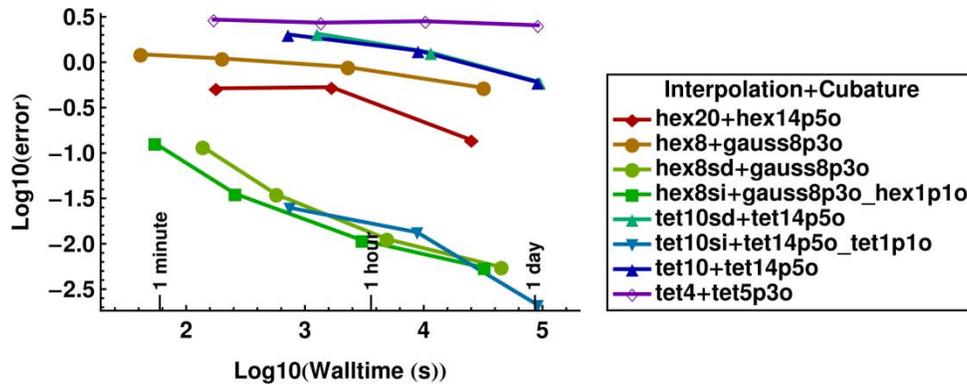


Figure 50: Error in maximum triaxiality versus walltime.

Selective integration vs. selective deviatoric

In the previous figures, some interesting trends can be seen between the selective deviatoric HEX8SD and the selective integration HEX8SI formulations.

For the load at displacement metric, we can see the accuracy for a given mesh is approximately the same. However, the selective deviatoric formulation takes more computational time. As noted earlier, this is due to the non-symmetric stiffness matrix it generates. The solver for such a matrix requires in general twice as much time to solve as a similar symmetric matrix.

For the maximum triaxiality metric, the same trends are seen.

Additional work

It is possible that other selective integration formulations also perform well. For example, a HEX20SI+HEX14O5O_HEX1P1O and TET10SI+TET14P5O_TET1P1O could be investigated. These would benefit from being able to represent curved surfaces while still retaining the advantages of the selective integration technique.

Q1P0 element implementation into SIERRA

As part of this project, the q1p0 formulation for the eight-node hexahedral element has been implemented into the SIERRA codes. This element is available for both explicit and implicit analysis.

This element has been used extensively in literature. In materials which exhibit near-incompressible behavior—where the effective Poisson’s ratio approaches 0.5—the element performs very well. This list includes structural metals undergoing large plastic deformations where the plastic deformation is assumed to be isochoric.

Formulation

In the q1p0 formulation, the internal forces due to material stress are selectively integrated. We break up the nodal force integral into two parts

$$\int_{\Omega^e} \frac{d\phi_I}{dx_j} \sigma_{ij} d\Omega^e = \underbrace{\int_{\Omega^e} \frac{d\phi_I}{dx_j} \left(\sigma_{ij} - \frac{1}{3} \sigma_{kk} \delta_{ij} \right) d\Omega^e}_{\text{deviatoric response}} + \underbrace{\int_{\Omega^e} \frac{d\phi_I}{dx_j} \left(\frac{1}{3} \sigma_{kk} \delta_{ij} \right) d\Omega^e}_{\text{volumetric response}} \quad (1.59)$$

and evaluate the deviatoric response using an 8 point Gauss rule and the volumetric response using a single integration point.

Usage syntax

This element can be used by including the following block at the SIERRA scope:

```
# define the q1p0 section
Begin Solid Section section_q1p0
  Formulation = q1p0
  Strain Incrementation = strongly_objective
End
```

and by calling out this section within the `Finite Element Model` block using this syntax:

```
# define material parameters
Begin Parameters For Block ...
  Material ...
  Section = section_q1p0
End
```

Output

When post processing results, one should generally use the values of internal state variables at the volumetric integration point. This is the first integration point output to the ExodusII file. For example, one should look at `eqps_1` to examine the plastic strain field instead of looking at all 9 values or using an averaging procedure.

For stress based values, such as `stress` and `von_mises`, these are not output at each integration point. Instead, these are calculated by using a stress tensor which combines the pressure at the volumetric integration point with the average of the deviatoric stress in the other 8 integration points.

Additional options

While the syntax given above should be sufficient for nearly all analyses, there are some additional options available within the `Solid Section` block to alter the performance of this element. These options, along with their defaults, are given in the following table.

```
# define the q1p0 section
Begin Solid Section section_q1p0
  Formulation = q1p0
  Strain Incrementation = strongly_objective
  Q1P0 Stabilization Threshold = <real> (0.0)
  Q1P0 Timestep Scale Factor = <real> (0.95)
  Q1P0 Timestep Wave Speed = <string> VOLUMETRIC|SHEAR|AUTOMATIC(AUTOMATIC)
  Q1P0 Timestep Length Scale = <string> DEFORMED_NODAL_DISTANCE|
  MINIMUM_MAPPING_STRETCH|INSCRIBED_SPHERE_DIAMETER
  (MINIMUM_MAPPING_STRETCH)
End
```

Strain Incrementation

The only strain incrementation available for this element is `strongly_objective`. Because this is not the default, it must be called out explicitly to avoid a warning message being generated.

Q1P0 Stabilization Threshold

In some scenarios, elements may invert under a significant and localized loading condition. This option is provided as a way to avoid these inversions. In these cases, a value of 0.25 might be preferable. As the element becomes significantly distorted, the formulation is smoothly reverted back to a fully integrated formulation which this option is active.

For each element, the minimum stretch λ_{\min} out of all integration points is obtained. If this value is above the threshold $\lambda_{\text{threshold}}$ specified, no modification is done. If it is below, the internal forces are calculated as

$$\mathbf{f}_{\text{int}} = (1 - \phi) \mathbf{f}_{\text{int}}^{\text{q1p0}} + \phi \mathbf{f}_{\text{int}}^{\text{full}} \quad (1.60)$$

where $\mathbf{f}_{\text{int}}^{\text{q1p0}}$ are the forces from a q1p0 formulation, and $\mathbf{f}_{\text{int}}^{\text{full}}$ are the forces from a fully integration formulation, and

$$\phi^{\text{def}} = \begin{cases} 0 & , \lambda_{\min} \geq \lambda_{\text{threshold}} \\ \frac{1}{2} + \frac{1}{2} \cos(\pi \lambda_{\min} / \lambda_{\text{threshold}}) & , \lambda_{\min} < \lambda_{\text{threshold}} \end{cases} \quad (1.61)$$

The value of ϕ is available in the analysis by requesting the element variable `stabilization_factor`.

Keep in mind that any value other than zero causes this formulation to differ from that of a selectively integrated q1p0 formulation. It may also cause the stiffness matrix to be non-symmetric, which may cause issues in converging to a solution, depending on the solver used.

By default, this option is set to zero, or off.

Q1P0 Timestep Scale Factor

In the critical timestep calculation, this is a scale factor applied to the calculated critical timestep. The default of 0.95 should be sufficient for most analyses.

If another timestep scale factor is prescribed within the `Parameters for Presto Region` block, the two factors have a multiplicative effect.

Q1P0 Timestep Wave Speed

The wave speed used to calculate the critical timestep can be selected from the following options:

- If set to `VOLUMETRIC`, this will be calculated as $\sqrt{3 * \text{bulk_modulus} / \text{density}}$.
- If set to `SHEAR`, this will be calculated as $\sqrt{2 * \text{shear_modulus} / \text{density}}$.
- If set to `AUTOMATIC`, the maximum of the other two options will be used. This option is the default.

The default should be sufficient for all analyses.

Q1P0 Timestep Length Scale

The method of computing the element length scale used in the critical timestep calculation can be selected from the following options:

- With the `DEFORMED_NODAL_DISTANCE` option, the length is calculated as the minimum nonzero distance between two nodes. This is the fastest option and works for most analyses.
- With the `MINIMUM_MAPPING_STRETCH` option, the length is calculated as the minimum stretch out of all integration points for the mapping from a unit cube to the current configuration of the element. While computationally intensive, this formulation is very robust. For this reason, this option is chosen as the default.
- With the `INSCRIBED_SPHERE_DIAMETER` option, the length is calculated as the diameter of the largest sphere which can be inscribed within the element. For this calculation, each side of the element is approximated by a flat plane which is tangent to the true—possible curved—surface at the midpoint.

Verification tests

A number of verification tests have been performed comparing this element formulation to others available within SIERRA. For reference, these formulations are listed below:

- The `ugelastic` formulation refers to the following block. This is the default in Adagio.

```
Begin Solid Section ...
  Formulation = mean_quadrature
  Strain Incrementation = midpoint_increment
End
...
Effective Moduli Model = elastic
```

- The `ugpronto` formulation refers to the following block. This is the default in Presto.

```
Begin Solid Section ...
  Formulation = mean_quadrature
  Strain Incrementation = midpoint_increment
End
...
Effective Moduli Model = pronto
```

- The `sdmi` formulation refers to the following block.

```
Begin Solid Section ...
  Formulation = selective_deviatoric
  Deviatoric Parameter = 1
  Strain Incrementation = midpoint_increment
End
```

- The `q1p0` formulation refers to the following block.

```
Begin Solid Section ...
  Formulation = q1p0
  Strain Incrementation = strongly_objective
End
```

- The `f11` formulation refers to the following block.

```
Begin Solid Section ...
  Formulation = fully_integrated
  Strain Incrementation = strongly_objective
End
```

Note that the `Effective Moduli Model` line is used in the calculation of hourglass restoring forces which are only applied for the `mean_quadrature` element. For other elements, this setting has no effect.

Cantilever beam—Implicit

In this problem, a simple cantilever beam composed of an elastic-plastic material is subjected to a vertical tip displacement. The mesh and equivalent plastic strain field at the end of the simulation are shown in Figure 51.

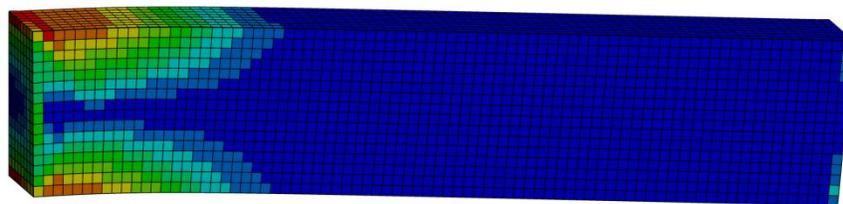


Figure 51: The equivalent plastic strain field in the cantilever beam problem is shown for the `x8` mesh.

The boundary conditions in this problem are as follows.

- A symmetry plane is enforced on the left side.
- A symmetry plane is enforced on the near side.
- The vertical displacement of all nodes on the left side is fixed.
- The vertical displacement of all nodes on the right side is progressively displaced over 50 timesteps.

To assess convergence behavior, we measured the vertical resultant load at the last timestep. A series of mesh refinements was done, with each successive refinement cutting the element edge length in half and increasing the total number of elements by a factor of 8. The results are shown in Table 22. Additionally and for convenience, the change in load as a result of successive mesh refinements is shown in Table 23.

Mesh	full	q1p0	sd	ugelastic	ugpronto
x1	23,029.81	19,648.52	19,649.13	29,703.02	20,225.83
x2	20,588.82	19,053.47	19,053.69	22,103.71	crash
x4	19,469.69	18,861.55	18,861.67	19,751.51	crash
x8	19,026.14	18,803.41	18,803.49	19,069.32	18,815.00
x16	18,866.25	18,783.17	18,783.24	18,868.56	crash

Table 22: The resultant force is shown for each mesh and each element formulation for the implicit cantilever beam problem.

Refinement	full	q1p0	sd	ugelastic	ugpronto
x1 → x2	-2440.99	-595.05	-595.44	-7599.31	crash
x2 → x4	-1119.13	-191.91	-192.02	-2352.20	crash
x4 → x8	-443.55	-58.15	-58.18	-682.19	crash
x8 → x16	-159.89	-20.23	-20.25	-200.76	crash

Table 23: The change in resultant force is shown for each successive mesh refinement and each element formulation for the implicit cantilever beam problem.

In these tables, we see several important trends.

- The `ugpronto` formulation crashed 3 out of 5 times, with no clear trend towards mesh discretization. The hourglass restoring forces applied with this model are discontinuous which causes the problem not to be locally convex. Because of this, the system of equations becomes ill-posed and a solution cannot be obtained. This is a known issue with this effective moduli model and the solution is to use a different model.
- Apart from `ugpronto`, all element formulations converged to the same solution as the mesh was refined. Additionally, they all approached the true solution from above. So at any given refinement, the approximate solution overpredicted the resultant load.
- The `ugelastic` formulation—which is the default—behaved the worst for a given mesh refinement. The system response was overly stiff. This is consistent with the assumptions made for this element formulation.

- Both the q_{1p0} and the sd formulations performed well and behaved almost identically. This is not surprising as the two formulations are very close to one another. In fact, for infinitesimal displacements, they are equivalent.

Using Richardson extrapolation, the true solution was estimated and used to generate a measure of the error for each run. The percentage error was calculated using a log-based scale. These error measures are given in Table 24.

Mesh	full	q_{1p0}	sd	ugelastic	ugpronto
x1	20.441	4.562	4.565	45.886	7.458
x2	9.236	1.486	1.488	16.336	<i>crash</i>
x4	3.647	0.474	0.475	5.085	<i>crash</i>
x8	1.343	0.165	0.166	1.570	0.227
x16	0.499	0.058	0.058	0.511	<i>crash</i>

Table 24: The percentage error is shown for each mesh and each element formulation for the implicit cantilever beam problem.

In this table, we immediately see that the q_{1p0} and sd formulations outperform all others for a given mesh discretization. They both outperform the default ugelastic formulation by a significant margin. For example, the x2 mesh with the q_{1p0} formulation outperforms the x8 mesh with the ugelastic formulation, despite the latter having 64 times the number of elements and 7 times the number of integration points.

The total cpu-hours required for each run are given in Table 25. Note that the number of cores used was not constant in general, but was constant for a given mesh size.

Mesh	full	q_{1p0}	sd	ugelastic	ugpronto
x1	0.0104	0.0127	0.0099	0.0067	0.0063
x2	0.0928	0.1093	0.0538	0.0202	<i>crash</i>
x4	1.9797	2.2520	1.2232	0.2990	<i>crash</i>
x8	28.189	31.561	19.031	5.2043	16.282
x16	433.64	479.78	298.12	72.925	<i>crash</i>

Table 25: The total cpu-hours are shown for each mesh and each element formulation for the implicit cantilever beam problem.

In this table, we see that the full, q_{1p0} , and sd formulations are significantly slower on a per-element basis than the ugelastic formulation. However, even taking this slowdown into effect, the multiple integration point elements outperform the single integration point element.

The difference in time between the q_{1p0} formulation and the sd formulation is at least partly on account of the latter using a quicker but less accurate calculation for the strain increment.

Constrained block compression—Implicit

In this test, a uniform cube of an elastic-plastic material is smashed between two rigid plates. Nodes on the top and bottom surface (which contact the plate first) are glued to the plate such that they cannot move tangentially or normally away from it. Nodes on the sides of the block are constrained not to pass through the plates though a frictionless contact.

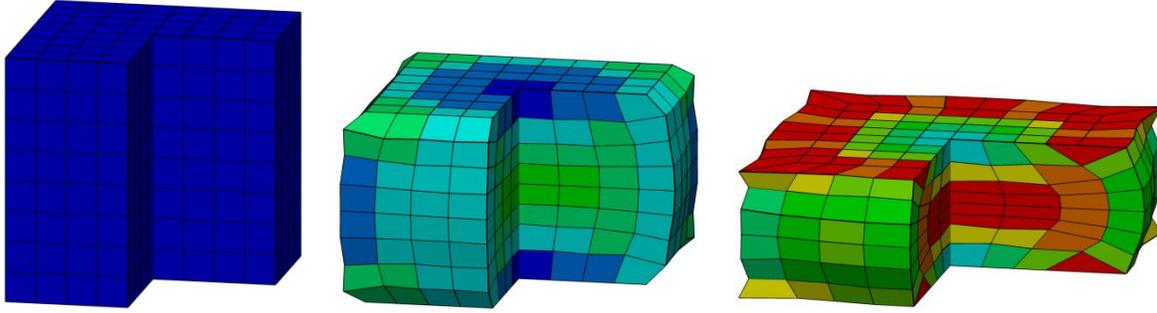


Figure 52: The constrained block is shown being progressively deformed.

The scaled load at full displacement is shown in Table 26. The load values in this table have been scaled such that a value of 1 represents approximately the true solution.

Mesh	full	q1p0	sd	ugelastic	ugpronto
x1	10.896	1.2956	1.1966	6.4308	1.3817
x2	4.9641	1.0452	0.9907	2.7880	1.0582
x4	2.5246	1.0149	1.0170	1.6561	0.9866
x8	1.6179	1.0291	1.0280	1.2558	1.0231
x16	1.2637	1.0098	1.0129	1.0799	1.0107
x32	1.0839	1.0062	1.0104	1.0320	1.0105
x64	1.0051	1.0031	<i>crash</i>	1.0246	<i>crash</i>

Table 26: The scaled load at displacement is shown for each mesh and each element formulation for the implicit constrained block compression problem.

Constrained block compression—Explicit

In addition for testing for element robustness under localized deformations, this study also tests for robustness of the element timestep calculation.

Mesh	full	q1p0	sd	ugelastic	ugpronto
x1	16.178	2.2386	1.8876	10.672	2.4751
x2	7.4393	1.5791	1.4893	4.3407	1.709
x4	3.8140	1.2705	1.2125	2.5082	1.4482
x8	2.4639	1.1708	1.1582	1.8666	1.2217
x16	1.9351	1.0866	1.0795	1.5756	1.1165
x32	1.6536	1.0419	1.0392	1.2728	1.0654
x64	1.3793	1.0175	<i>crash</i>	1.1368	1.038

Table 27: The scaled load at displacement is shown for each mesh and each element formulation for the implicit constrained block compression problem.

Distribution

1	MS9042	A. Brown	8259 (electronic copy)
1	MS9042	M. Chiesa	8259 (electronic copy)
1	MS9042	J. Dike	8259 (electronic copy)
1	MS9042	S. English	8259 (electronic copy)
1	MS9042	J. Foulk	8256 (electronic copy)
1	MS9042	M. Gonzales	8250 (electronic copy)
1	MS9042	T. Kostka	8259 (electronic copy)
1	MS9042	J. Ostien	8256 (electronic copy)
1	MS9042	M. Veilleux	8259 (electronic copy)
1	MS0359	D. Chavez, LDRD Office	1911 (electronic copy)
1	MS0899	RIM-Reports Management	9532 (electronic copy)

