PRISMS-Plasticity: An Open Source Crystal plasticity FE Code

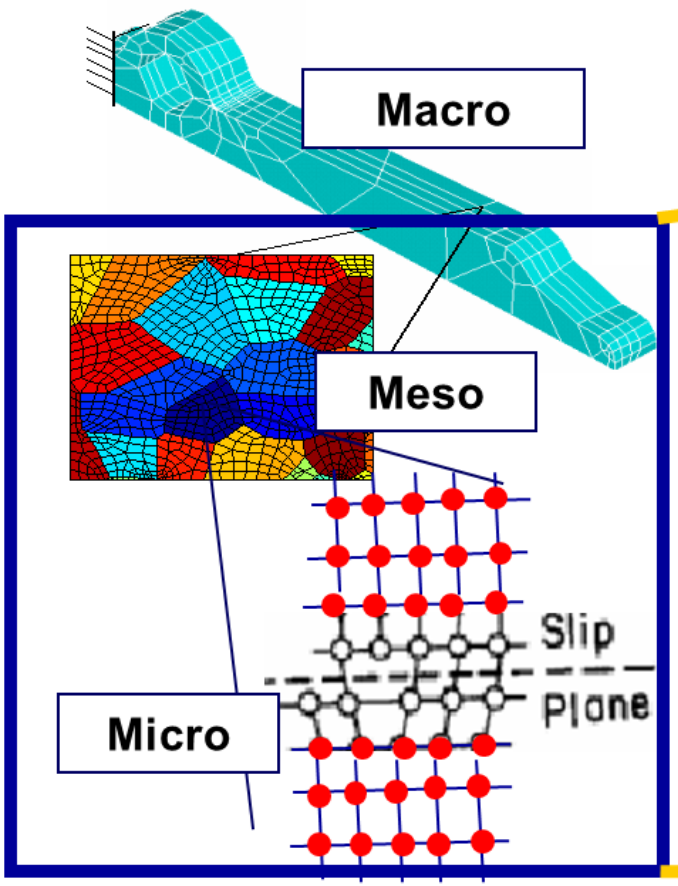
Mohammadreza Yaghoobi
Department of Materials Science and Engineering, University of Michigan, Ann Arbor

ICME Training Session
November 2021

TMS ICME
6TH WORLD CONGRESS ON INTEGRATED COMPUTATIONAL MATERIALS ENGINEERING (ICME 2021)
NOVEMBER 14-18, 2021 | www.tms.org/ICME2021
Hyatt Regency Lake Tahoe | Lake Tahoe, Nevada, USA

U.S. DEPARTMENT OF ENERGY | Office of Science
UNIVERSITY OF MICHIGAN
Center for PRedictive Integrated Structural Materials Science
PRISMS

Crystal plasticity FEM



Macro-deformation information

Homogenized (macro) properties

Boundary value problem for microstructure

Solve for deformation field

meso deformation gradient

Mesoscale stress, consistent tangent

Integration of constitutive equations

Continuum slip theory

Consistent tangent formulation (meso)

PRISMS-Plasticity

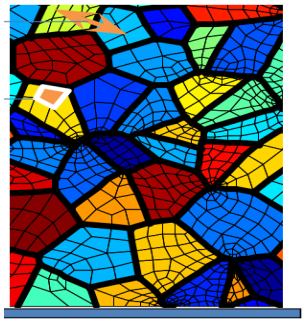
An Open Source Crystal plasticity Finite Elements Code
(github.com/prisms-center/plasticity)

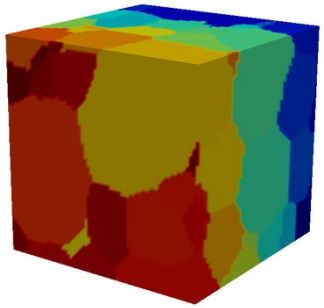
Advanced Capabilities:

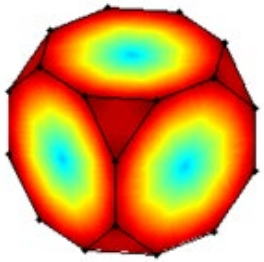
- Rate-independent and Rate-dependent Crystal plasticity Constitutive models.
- PTR and TDT Twinning model.
- Isotropic and Kinematic hardenings.
- Easily connected to the common pre-processors (DREAM3D, Neper).
- Integration with Materials Commons.
- High-Performance: Ideal scaling for >1,000 processors.
- Multiphase problems.
- Effect of grain size.

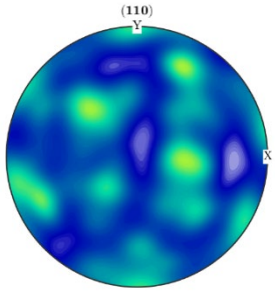
User-Friendly:

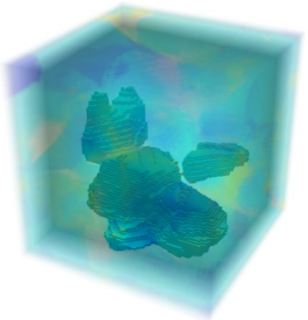
- Simple input file.
- Detailed online user guide for the code.
- post-processors (MTEX and Paraview).
- Detailed online user guide for linking pre-processing and post-processing.
- Applications to get you started.
- Highly modular and easy to add new features to the code.
- Framework to implement different crystal plasticity constitutive models.







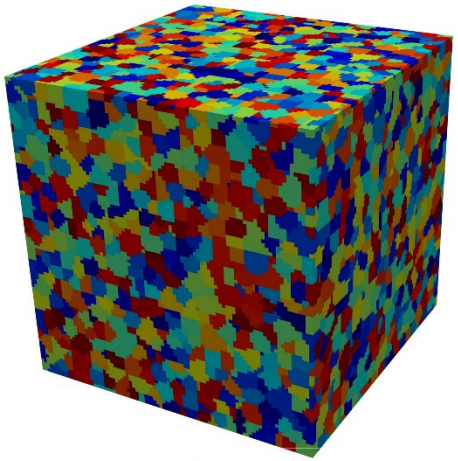


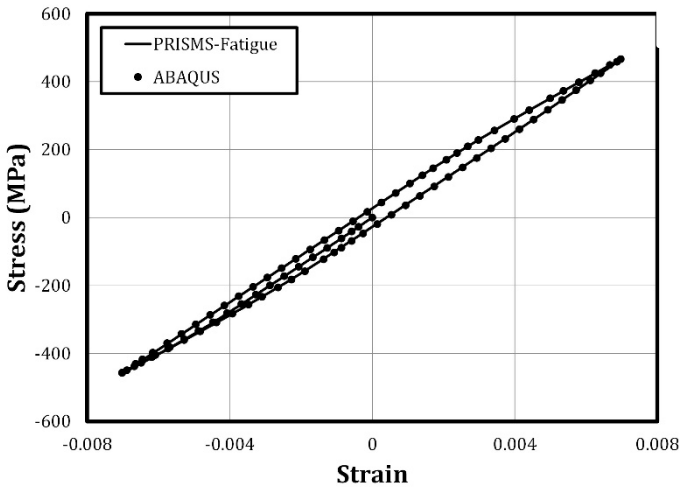


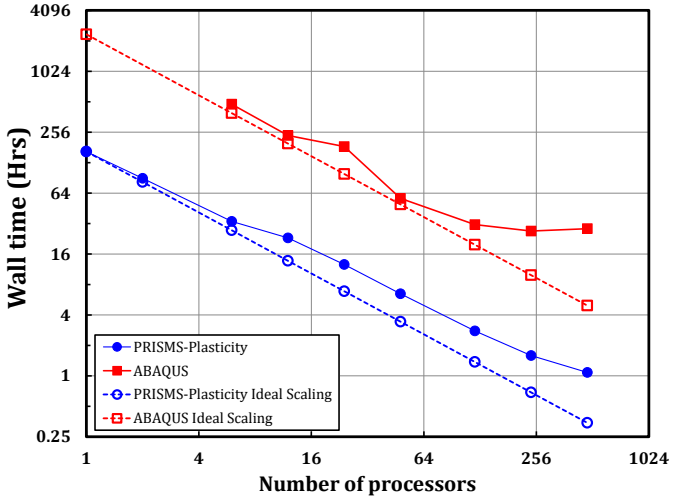
PRISMS Plasticity Scaling Study Versus Abaqus

Cyclic response of polycrystalline Al alloy 7075-T6

- Microstructure with Random texture and ~7,500 grains; 90^3 FE discretization
- Periodic BCs; Uniaxial cyclic loading; Kinematic Hardening;







The results shows that PRISMS-Plasticity can simulate the sample in 1.08 hours using 480 processors, while ABAQUS requires 28.5 hours using the same number of processors.

PRISMS-Plasticity website

- Google prisms-center/plasticity github
- Going over Training_Materials, applications, docs, src, include folders
- Going over an application folder: plasticity/applications/crystalPlasticity/fcc/compression/
- PRISMS-Plasticity required input files:

1) Main input file (prm.prm)
2) Grain ID file
3) Orientation file
4) BCs file
5) Latent Hardening Ratio file
6) Slip Direction/Normal (Twin Direction/Normal)

Microstructure generation

- Run DREAM3D by double clicking on Dream3D icon 

.
- Go to file->Open.
- Looking into the prisms-plasticity folder: /home/icme/tools/plasticity/Training_Materials/Pre-Processing/FCC.
- Select the PRISMS_pipeline_fcc.json file.

Microstructure generation

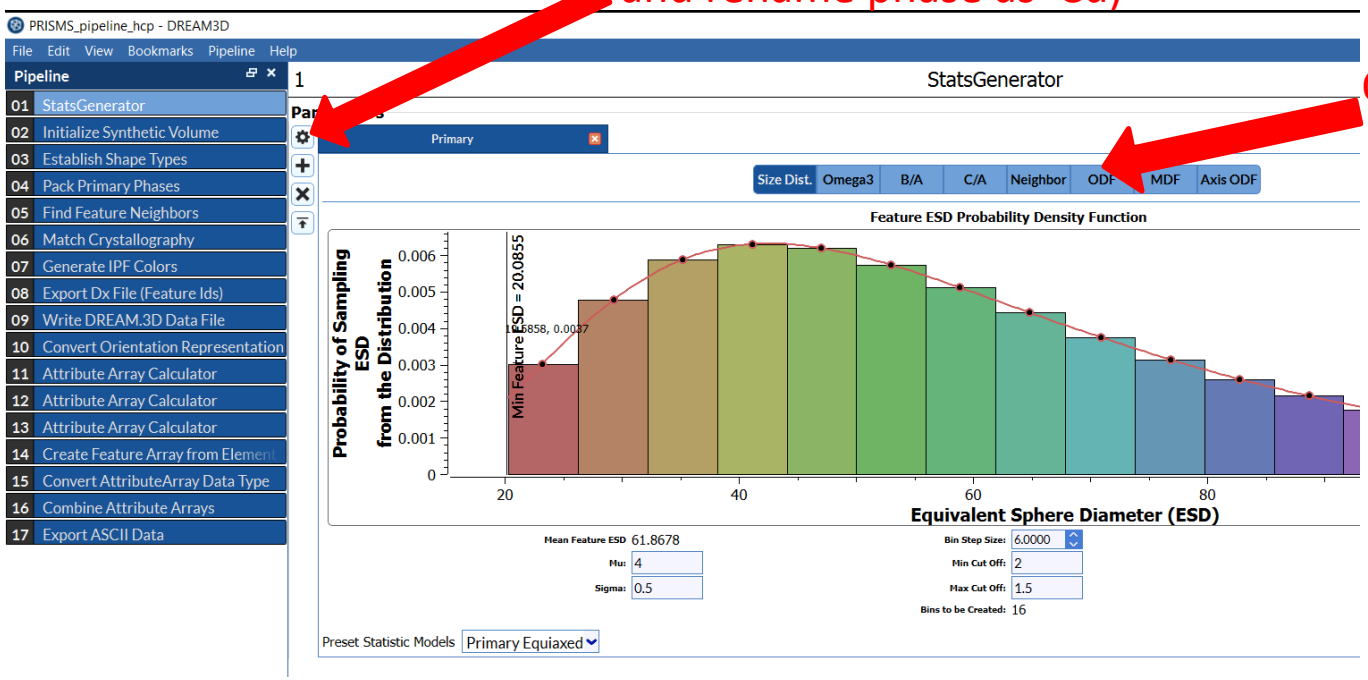
- This will open a prepared pipeline which can generate the PRISMS-Plasticity input files.

- In the pipeline, there are different parts numbered from 01 to 17.

01 StatsGenerator: Here, you can define the statistical information regarding the crystal structure, grain size, shape, orientation.

Crystal structure (We'll choose Cubic and rename phase as Cu)

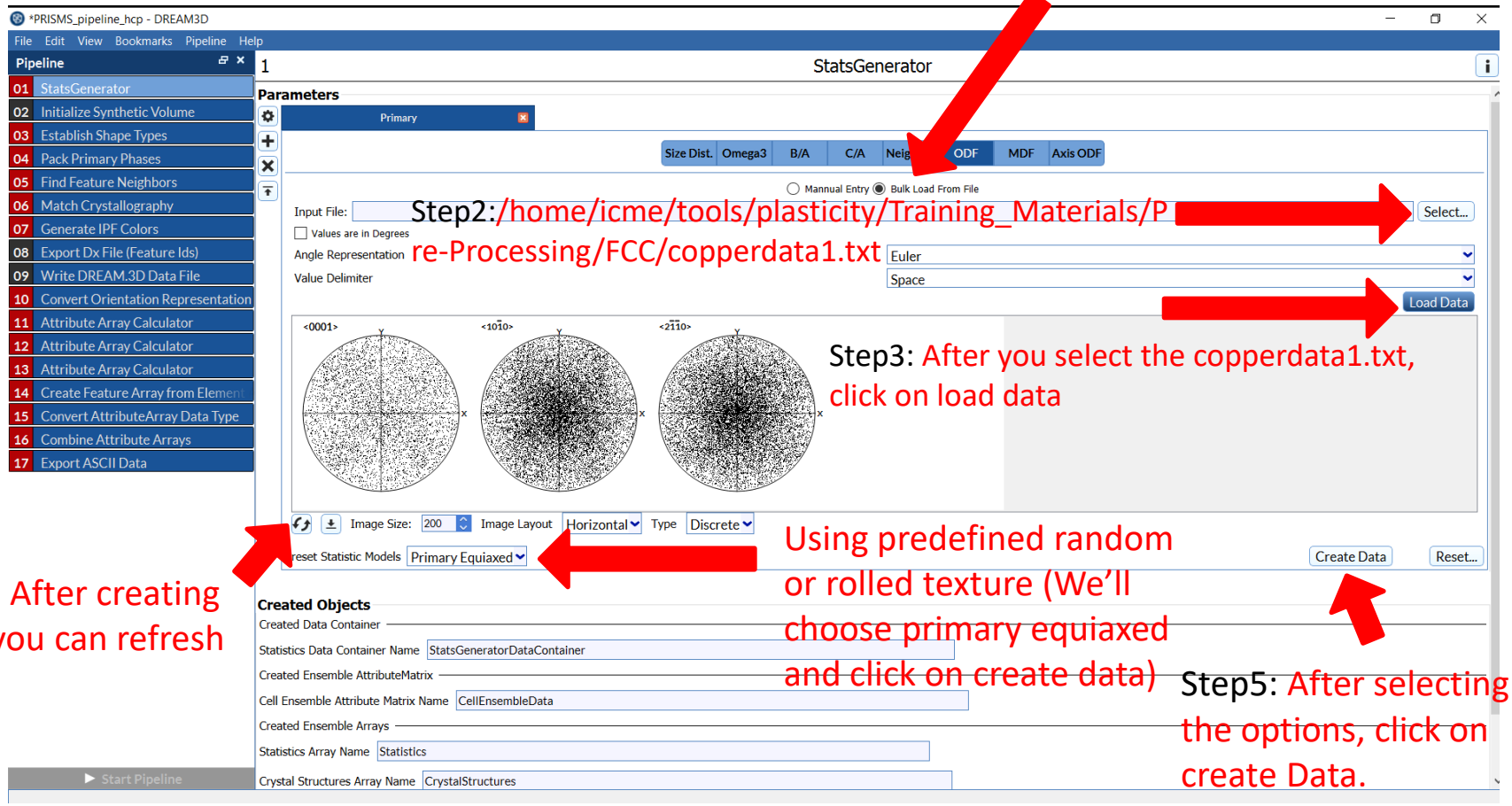
Orientations



Microstructure generation

Orientations:

Step1: Loading from EBSD experiment



Step2:/home/icme/tools/plasticity/Training_Materials/Pre-Processing/FCC/copperdata1.txt

Step3: After you select the copperdata1.txt, click on load data

Step4: After creating data, you can refresh

Using predefined random or rolled texture (We'll choose primary equiaxed and click on create data)

Step5: After selecting all the options, click on create Data.

Microstructure generation

Reading the texture from the EBSD experiment:

EBSD input file format for bulk load from file

File Edit Search View Encoding Language Settings Tools Macro Run Window ?

mgdata.txt

```
Angle Count:588638
1.05842 1.73449 0.49267 1    1
1.04793 1.73497 0.4861  1    1
0.70312 1.797   2.07639 1    1
0.94064 2.49537 5.27844 1    1
0.83093 1.1384  1.43888 1    1
0.69873 1.80268 2.07742 1    1
1.60536 2.10196 5.05798 1    1
3.85197 1.34008 4.20304 1    1
3.97084 2.00518 0.65303 1    1
0.68157 1.8056  2.07545 1    1
3.9958  2.03406 3.28571 1    1
3.75425 1.31183 0.01477 1    1
4.03747 1.55849 0.74614 1    1
3.06563 0.93898 5.6267  1    1
1.73482 1.30826 3.16308 1    1
1.74051 1.31024 3.15811 1    1
1.74679 1.31532 0.0167  1    1
1.74381 1.30945 0.01544 1    1
1.74193 1.31495 0.01806 1    1
1.7428  1.30853 0.02107 1    1
1.74027 1.30931 3.15521 1    1
1.7468  1.31609 3.15246 1    1
2.29945 1.73205 3.2328  1    1
3.04848 0.29176 1.61409 1    1
4.80671 1.82614 0.33644 1    1
```

Microstructure generation

02: Initialize Synthetic Volume

Estimates on the number of grains

File Edit View Bookmarks Pipeline Help

Pipeline

01 StatsGenerator
02 Initialize Synthetic Volume
03 Establish Shape Types
04 Pack Primary Phases
05 Find Feature Neighbors
06 Match Crystallography
07 Generate IPF Colors
08 Export Dx File (Feature Ids)
09 Write DREAM.3D Data File
10 Convert Orientation Represent
11 Attribute Array Calculator
12 Attribute Array Calculator
13 Attribute Array Calculator
14 Create Feature Array from Ele
15 Convert AttributeArray Data T
16 Combine Attribute Arrays
17 Export ASCII Data

▶ Start Pipeline

2 Initialize Synthetic Volume

Parameters

☑ Estimate Number of Features

Estimated Primary Features **324**

Dimensions 32 32 32

Resolution 11 11 11

Origin 0 0 0

Box Size in Length Units
X Range: 0 to 352 (Delta: 11)
Y Range: 0 to 352 (Delta: 11)
Z Range: 0 to 352 (Delta: 11)

Required Objects

Cell Ensemble Data

Statistics StatsGeneratorDataContainer / CellEnsembleData / Statistics

Phase Types StatsGeneratorDataContainer / CellEnsembleData / PhaseTypes

Created Objects

Synthetic Volume Data Container SyntheticVolumeDataContainer

Cell Data

Cell Attribute Matrix CellData

Ensemble Attribute Matrix CellEnsembleData

Data Structure

- StatsGeneratorDataContainer
  - CellEnsembleData
    - CrystalStructures
    - PhaseName
    - PhaseTypes
    - Statistics
- SyntheticVolumeDataContainer *
  - CellData *

Number of voxels in each direction

Define the number of voxel for each grain

Filter List | Filter Library | Bookmarks

Filter List

Search for filter

- Abaqus Hexahedron Exporter
- Export Abaqus Surface Mesh
- Adaptive Alignment (Feature)
- Adaptive Alignment (Misorientation)
- Adaptive Alignment (Mutual Information)
- Add Bad Data
- Add Orientation Noise
- Align Sections (Feature)
- Align Sections (Feature Centroid)
- Align Sections (List)
- Align Sections (Misorientation)
- Align Sections (Mutual Information)
- Append Z Slice (Image Geometry)
- Apply Transformation to Geometry
- Attribute Array Calculator

Filter Count: 363

Microstructure generation

08 Export Dx File, 09: Write DREAM.3D Data File, 17: Export ASCII Data

Update the addresses here to a folder on your system.

After that, hit the Start Pipeline (Green button below pipelines)

You can check the generated files in the folder you selected. The folder is: /home/icme/tools/plasticity/Training_Materials/Pre-Processing/FCC.

Microstructure generation

orientations.txt:

The first row is dummy (does not matter).
After that, each line has four columns:
Grain ID and three Rodrigues vector components:
grainID, Rod1,rod2,rod3

Multiphase simulations:

If you have a multiphase simulations, an additional column should be added to the orientations.txt as a phaseID (Check user manual for more detail).

Additional Voxel info:

In addition to orientations (and possibly phase ID), if you have any other grain level information you want to pass, such as grain size, residual strain, etc., you can use a feature of “Additional Voxel info”, and add that as a new column (Check user manual for more detail)

C:\Users\yaghoobi\Downloads\plasticity-master-2021\Training_Materials\Pre-Processing\HCP\orientations.txt - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

orientations.txt

```
0 0 0 0
1 0.706527 0.45221 0.198347
2 14.9515 9.41504 -10.786
3 -1.10348 1.2271 1.17724
4 2.82433 -0.543489 -3.4207
5 1.45413 2.19316 -0.529163
6 -0.108922 -2.01401 0.405957
7 -0.180539 -0.545749 0.177164
8 3.37263 -4.8475 -1.7216
9 -6.04981 -10.4885 -11.4154
10 0.551727 0.222017 -0.0430181
11 -9.2471 -5.3786 6.10939
12 -0.955772 0.056508 -0.789907
13 0.488871 -0.507244 0.74779
14 -22.0284 -39.3318 -0.739059
15 -12.2119 5.63069 -6.85142
16 -0.303606 0.372028 -0.42964
17 3.66351 -0.0208949 -1.37557
18 0.9062 0.490992 -0.19592
19 0.435739 1.06345 4.58736
20 5.67255 -0.288694 0.955549
21 0.421963 1.1515 -0.125092
22 -2.40555 -3.54481 -0.539058
23 -0.609045 0.61898 0.0524783
24 -0.0637942 -3.05449 -2.13411
25 1.25453 0.240392 -1.02011
26 -3.79384 0.0290665 -1.37979
27 0.254935 -0.314292 0.257108
```

Microstructure generation

grainID.txt:

This file assigns the grainID for each voxel. Important information from this file is:

- Number of header lines
- Number of Voxel in each direction

In the presented file, we have 20 lines as header. After that, the numbers are grainID for each voxel. A 3d microstructure of $N1 \times N2 \times N3$ voxels is mapped into a 2D array. The corner voxel closest to the origin may be indexed as (1,1,1) because it is in the first voxel in x, y and z directions, respectively. Using this notation, any voxel can be indexed as (i, j, k) with $i \in \{1,2, \dots, N1\}, j \in \{1,2, \dots, N2\}, k \in \{1,2, \dots, N3\}$. Then the grain ID of such a voxel will be present in the $(j + (i-1) \times N2)^{th}$ row and in the kth column of grainID.txt (after header lines).

C:\Users\yaghoobi\Downloads\plasticity-master-2021\Training_Materials\Pre-Processing\HCP\grainID.txt - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

grainID.txt

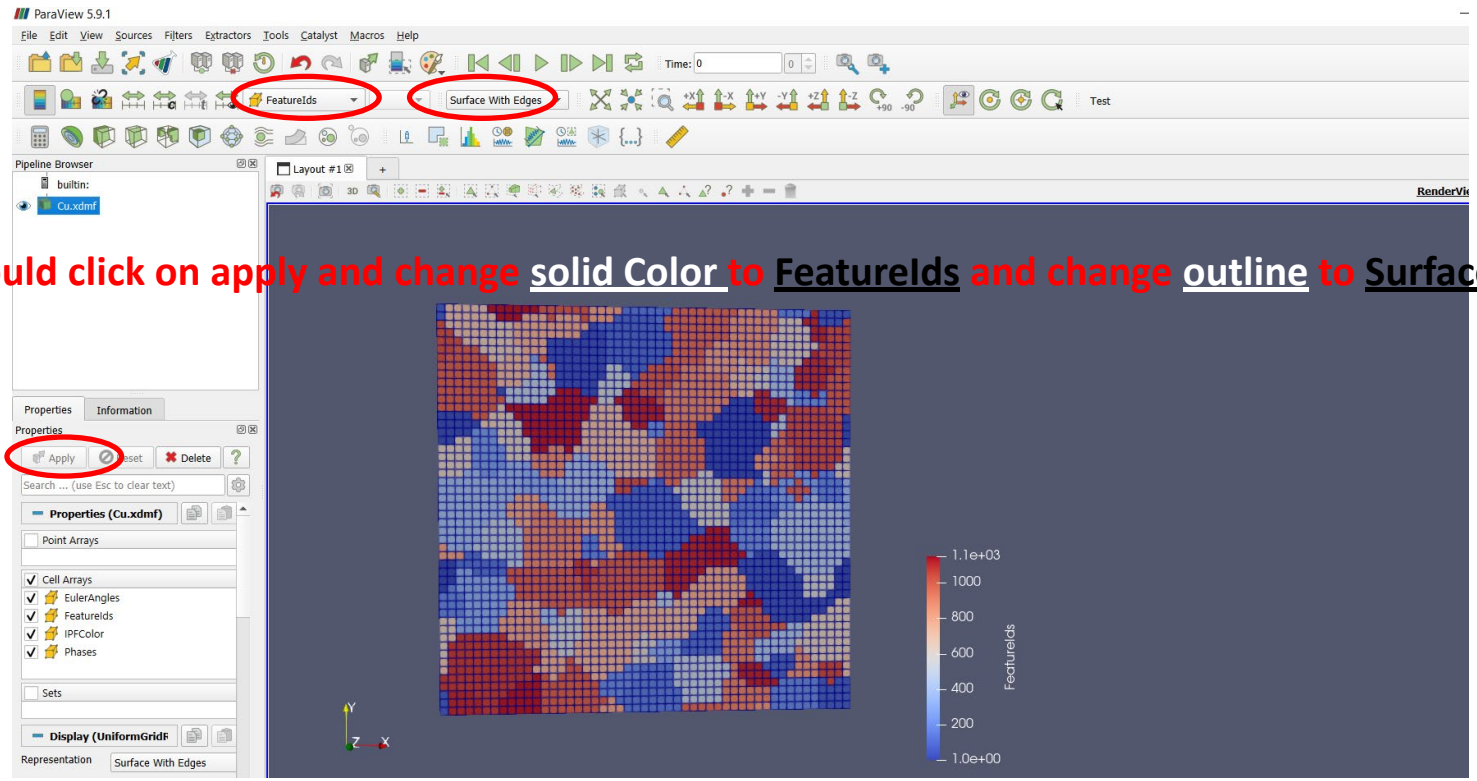
```
# object 1 are the regular positions. The grid is 50 50 50. The origin is
# at [0 0 0], and the deltas are 1 in the first and third dimensions, and
# 2 in the second dimension
#
object 1 class gridpositions counts 50 50 50
origin 0 0 0
delta  1 0 0
delta  0 1 0
delta  0 0 1
#
# object 2 are the regular connections
#
object 2 class gridconnections counts 50 50 50
#
# object 3 are the data, which are in a one-to-one correspondence with
# the positions ("dep" on positions). The positions increment in the order
# "last index varies fastest", i.e. (x0, y0, z0), (x0, y0, z1), (x0, y0, z2)
# (x0, y1, z0), etc.
#
object 3 class array type int rank 0 items 125000 data follows
1100 709 709 709 709 709 709 351 351 351 351 447 447 447 447 447 447 634 634
1100 709 709 709 709 709 557 351 351 351 351 447 447 447 447 447 447 634 634
1100 1100 709 557 557 557 557 557 351 351 351 447 447 447 447 447 634 634 63
1046 1046 1046 557 557 557 557 557 557 351 351 447 447 447 447 447 634 634 6
1046 1046 1046 557 557 557 557 557 557 557 557 447 447 447 447 447 634 634 6
1046 1046 1046 557 557 557 557 557 557 557 557 575 575 447 127 127 127 634 6
1046 1046 557 557 557 557 557 557 557 557 575 575 575 575 127 127 127 842 84
625 1046 557 557 557 557 557 557 557 575 575 575 575 575 127 127 127 842 842
```

Normal text file length : 507,247 lines : 2,531 Ln : 1 Col : 1

Microstructure generation

.xdmf and .dream3d files:

- These two files are not required by PRISMS-Plasticity. One can open the .xdmf file with paraview for the sake of visualization of the microstructure.
- Open Paraview on the desktop. Select the file from the folder */home/icme/tools/plasticity/Training _Materials/Pre-Processing/FCC*. After opening the .xdmf file with paraview, it opens a box, and you should select the **XdmfReaderS**.



- **You should click on apply and change solid Color to FeatureIds and change outline to Surface with edges.**

General Linux Terminal Commands

ls Used to show the contents of the current directory

pwd Prints the current working directory (where you are in the file structure)

cd Change directory (cd .. to go up one directory, cd [insert directory here] to go into a directory)

scp Copy a file or directory (cp [file to be copied] [where to copy it to], cp –r [directory to be copied] [where to copy it to])

mv Move a file or directory (mv [file to be moved] [where to move it to], mv –r [directory to be moved] [where to move it to])

rm Delete a file or directory (rm [file to be deleted], rm –r [directory to be moved])

mkdir Create a new directory (mkdir [directory name])

Main Input file (prm.prm)

FE parameters

set Order of finite elements = 1

Polynomial order of interpolation function (1 => linear basis functions)

set Order of quadrature = 2

Quadrature point order (2 => 2^n) quadrature points where *n* is the physical dimension)

Domain parameters

* Assuming that the simulation domain is a cuboid of arbitrary dimensions.

set Number of dimensions = 3

Number of physical dimensions for the simulation

set Domain size X = 1.0

set Domain size Y = 1.0

set Domain size Z = 1.0

The size of the domain in the x, y, and z directions.

Main Input file

Mesh parameters

Meshing can be performed through deal.II or by reading an external mesh through a file generated using Gmsh.

Deal.II mesh generator

The mesh generation starts with a single unit cell, and slices it in x, y and z directions as many times as the subdivisions or refinement factor indicates.

set Subdivisions X = 2

set Subdivisions Y = 2

set Subdivisions Z = 2

The number of mesh subdivisions in the x, y, and z direction (2by2by2 mesh so far).

set Refine factor = 2

The number of initial refinements of the coarse mesh (2 => each coarse element is subdivided into 2^2 = 4 smaller elements) -> 8by8by8 is the mesh.

It is always more efficient to use the set Refine factor to refine the mesh rather than using the subdivisions in the x,y, and z directions.

There are some exceptions in which you cannot use set refinement: (1) Applying periodic BCs (2) Applying grainID deletion. In these two cases, one can only use set Subdivisions for mesh refinement.

Main Input file

External Mesh Generation

The external mesh with the format of Gmsh can be read.

set Use external mesh = true

Flag to indicate whether to use external mesh.

set Name of file containing external mesh = n200-id4_hex.msh

Name of external mesh file.

set External mesh parameter = 0.05

The external mesh parameter: The ratio of defined region size to the Domain size.
When an external mesh is read, x, y, and z positions are not precisely defined due to roundoff errors. This matters in the case of BCs assignment. Accordingly, here, a margin is defined in which if a point is located in this margin, it will be considered as BCs. The size of this region in each direction is equal *to (External mesh parameter)×(Domain size in that direction)*.

Main Input file

Output parameters

In this section, all the features related to output generation are discussed..

set Write Output = true

Flag to write output vtu and pvtu files.

set Output Directory = results

Output Directory name

set Skip Output Steps = 1

Number of Output Steps to skip. The outputs will be generated every N steps, which N is defined in this line.

set Output Equivalent strain = true
set Output Equivalent stress = true
set Output Grain ID = true
set Output Twin fractions = true

By setting each of these flags to true, that will be outputted.

Advanced feature of adding variable to vtu and pvtu outputs!!!

Main Input file

Output parameters (Cntd.)

set Write Quadrature Output = true

Flag to write Quadrature Output, which is the information for each FE integration point.

set Skip Quadrature Output Steps = 1

Number of Quadrature Output Steps to skip.

What variables are outputted using Quadrature_Output feature?

https://github.com/prisms-center/plasticity/blob/master/src/materialModels/crystalPlasticity/updateAfterIncrement.cc
Starting line 264 which is: "*temp.push_back(cellOrientationMap[cellID]);*" up to the line 505 before "*} addToQuadratureOutput(temp);*"

Whatever variable is inside temp.push_back(variable); will be added to the QuadratureOutput.

If you do not want to print out a specific variable, you can comment it out by adding // to the start of that line.

These three lines in QuadratureOutput section of updateAfterIncrement.cc (lines 258-260) defines the spatial coordinate of the integration point (x, y, and z of the integration point):

temp.push_back(fe_values.get_quadrature_points()[q][0]);
temp.push_back(fe_values.get_quadrature_points()[q][1]);
temp.push_back(fe_values.get_quadrature_points()[q][2]);

If you apply any changes to **updateAfterIncrement.cc**, you need to recompile so the changes can be applied.

Main Input file

Output parameters (Cntd.)

set Tabular Output = true

One can use this feature to output the results (Both visualization and Quadrature outputs) at specific time table they define here. This is the flag for tabular time output.

set Tabular Time Output Table = 0,0.25,0.48

Simulation times at which outputs are generated.

Even if all output generation fields are disabled, a stressstrain.txt file is generated for each step which includes the average Green-Lagrange strain tensor, Cauchy stress tensor, twin activity volume (TwinRealVF), twin volume fraction (TwinMade), and slip activity (SlipTotal). If one wants to compare the average slip and twin activities, i.e., TwinRealVF vs SlipTotal, they should consider that average twin activity volume is the twin pseudo-slip systems shear activity divided by the characteristic twin shear constant.

The default Quadrature Output: GrainID, Phase ID (If Advanced Twinning model is not used), det(J), twin (1 if it is reoriented due to twinning and 0 if not), x,y,z (coordinates of quadrature point),rotnew(1), rotnew(2), rotnew(3) (updated Rodrigues vector of orientation), 9 components of Fe (elastic deformation gradient tensor), 9 components of Fp, 9 components of Cauchy stress tensor, slip activity for 84 slip systems (if there are less systems, it will be appeared as 0), and 6 twin activity volume. Again, the twin activity volume is the twin pseudo-slip systems shear activity divided by Characteristic twin shear constant.

Main Input file

Boundary conditions information

Boundary conditions (BCs) can be applied by defining the applied displacements or velocity gradient tensor. Different boundary conditions can be applied to the sample including:

Simple BCs
Velocity gradient tensor
Cyclic BCs
Tabular BCs
SEM-DIC (Digital Image Correlation) BCs
Periodic BCs

Simple BCs:

set Use Simple BCs = true

This flag defines if one use Simple BCs. It will be considered true if it is not defined. In the case of other BCs types, it should be defined as false.

set Boundary condition filename = BCinfo.txt

File name containing BC information.

set BC file number of header lines = 2

Number of header lines in BC file

set Number of boundary conditions = 4

Number of boundary conditions applied on the sample

Main Input file

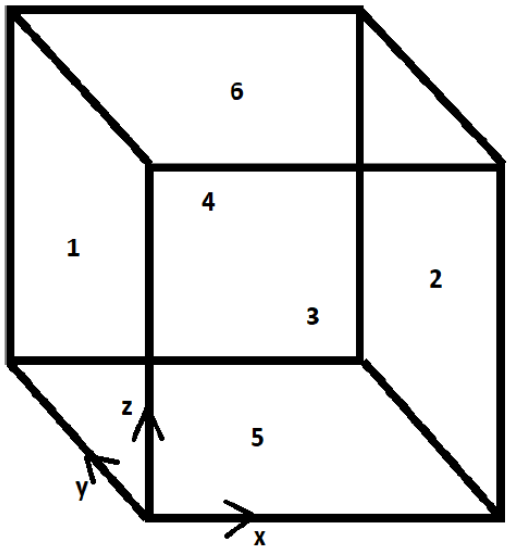
Boundary conditions information (Cntd.)

A sample boundary condition specification could look like:

BCinfo.txt

```
# Header lines = 2
# FaceID DoF FinalDisplacement
1 1 0
1 2 0
1 3 0
2 1 -0.05
```

Column 1: FaceID: refers to the face identifier and follows the numbering as indicated here:



Column 2: DoF: refers to the direction of application of displacement, where 1, 2 and 3 refer to the x, y and z directions respectively:

Column 3: Final displacement: The final displacement applied at the end of total time. The displacement will be linearly varied to reach this *Final displacement*.

Main Input file

Boundary conditions information (Cntd.)

Velocity gradient tensor:

set Use Simple BCs = false

This flag defines if one use Simple BCs. In this case, it should be defined as false.

set Use velocity gradient BC = true

Flag to indicate whether to use velocity gradient tensor to apply BCs.

set Velocity gradient row 1 = -0.005, 0, 0

set Velocity gradient row 2 = 0, -0.005, 0

set Velocity gradient row 3 = 0, 0, 0.01

Velocity gradient tensor including the multiplication factor.

Tabular BCs:

set Use Simple BCs = false

This flag defines if one use Simple BCs. In this case, it should be defined as false.

set Use Tabular BCs = true

Flag to indicate whether to use Tabular BCs.

set Tabular Boundary condition filename = BCinfoTable.txt

Flag to indicate whether to use Tabular BCs.

set Number of time data for Tabular BCs = 5

Number of time data for Tabular BCs (it includes the initial BCs).

Main Input file

Boundary conditions information (Cntd.)

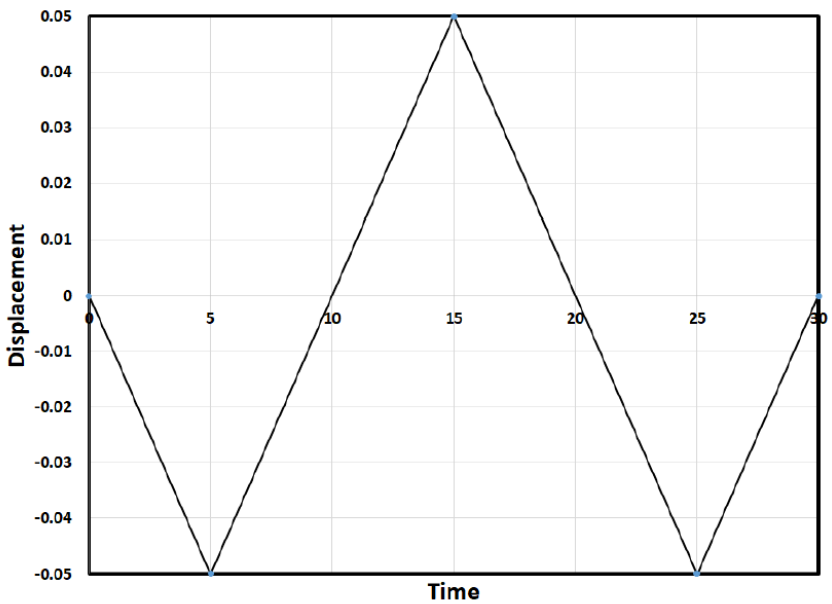
Tabular BCs (Cntd.):

set Tabular Time Table = 0,5,15,25,30

Table for Time intervals of Tabular BCs

The format for the BCinfoTable.txt (input file for Tabular BCs) is as follows:

1 1 0 0 0 0 0
3 2 0 0 0 0 0
5 3 0 0 0 0 0
6 3 0 -0.05 0.05 -0.05 0



The format is very similar to the Simple BCs input file. However, there is no header line for Tabular BCs input file. The first column is the FaceID. The second column is DoF. The following columns are the displacement at the times defined in the Tabular Time Table. For example, the Tabular BCs defined above is fixing faces 1,3 and 5 throughout the simulations. Face 6 is loaded in the 3 direction, i.e., z direction in Fig. 2.

Main Input file

Solver parameters

set Time increments = 0.1

ΔT for every increment

set Total time = 10

Maximum iterations for linear solver

Time increments depend on the total applied strain and total time. For a set of material system and loading/BCs, you need to first obtain the optimized ΔT for your simulation. As a general suggestion, your starting guess should be applying **strain increment** of 0.0001 in each step and then double it or divide it by 2.

set Maximum linear solver iterations = 50000

Maximum iterations for linear solver

set Relative linear solver tolerance = 1.0e-10

Relative linear solver tolerance

These two parameters controls the iterative solving the equilibrium equations using the BiCG method. The first one controls the maximum number of iteration, and the second one controls the criteria for the required residual to be satisfied. The convergence criteria threshold is the l2_norm of force vector multiplied by the relative tolerance we define here. The smaller the threshold, the more accurate the solution is but it takes longer.

Be careful, very large threshold may lead to divergence. For a set of material system and loading/BCs, you need to first obtain the optimized maximum iterations number and relative linear solver tolerance.

Main Input file

Input microstructure

set Voxels in X direction = 20

set Voxels in Y direction = 20

set Voxels in Z direction = 20

Number of voxels in X, Y, and Z directions. This should match the number of voxels in grain ID file.

set Grain ID file name = grainID.txt

Grain IDs file name.

set Header Lines GrainID File = 20

Number of header Lines in grain ID file (these are to be skipped).

set Orientations file name = orientations.txt

Grain orientations file name

Main Input file

Constitutive Model (Rate-Independent Model)

Multiplicative decomposition

$$\mathbf{F} = \mathbf{F}^{e}\mathbf{F}^{p}$$

Key idea of crystal plasticity

$$\mathbf{L}^{p} = \sum_{\alpha} \dot{\gamma}^{\alpha}\mathbf{S}^{\alpha}\,\mathrm{sign}(\tau^{\alpha})$$

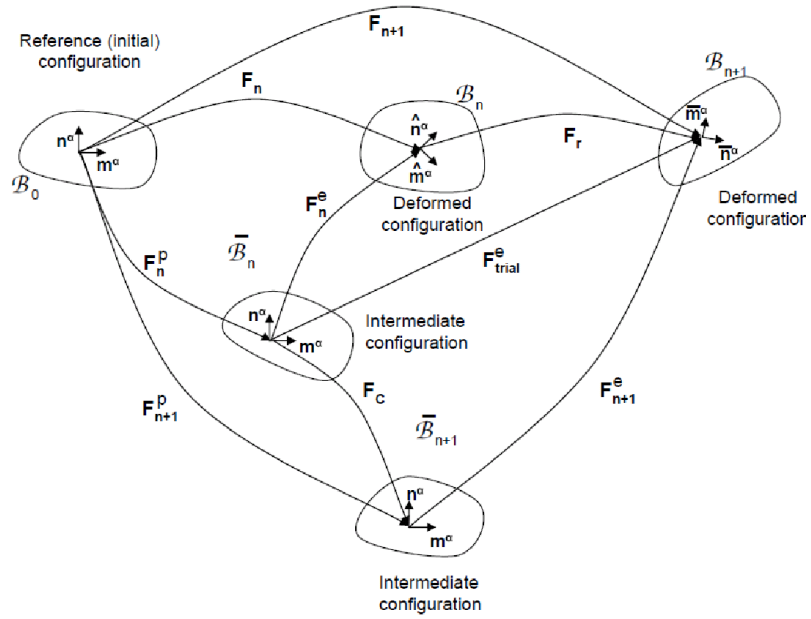
Yield surface

$$f^{\alpha} = |\tau^{\alpha}| - s^{\alpha}$$

Evolution of slip resistance

$$\dot{s}^{\alpha} = \sum_{\beta} h^{\alpha\beta}\dot{\gamma}^{\beta}$$

$$h^{\alpha\beta} = \begin{cases} h_0^{\beta}\left[1 - \dfrac{s^{\beta}}{s_s^{\beta}}\right]^{a^{\beta}} & \text{if } \alpha = \beta \text{ (coplanar systems)} \\ h_0^{\beta} q \left[1 - \dfrac{s^{\beta}}{s_s^{\beta}}\right]^{a^{\beta}} & \text{if } \alpha \neq \beta \end{cases}$$



Main Input file

Elasticity parameters

set Elastic Stiffness row 1 = 170.0e3, 124.0e3, 124.0e3, 0, 0,0
set Elastic Stiffness row 2 = 124.0e3, 170.0e3, 124.0e3, 0, 0,0
set Elastic Stiffness row 3 = 124.0e3, 124.0e3, 170.0e3, 0, 0,0
set Elastic Stiffness row 4 = 0, 0, 0, 75.0e3, 0, 0
set Elastic Stiffness row 5 = 0, 0, 0, 0, 75.0e3, 0
set Elastic Stiffness row 6 = 0, 0, 0, 0, 0, 75.0e3

Input elastic stiffness matrix in Voigt notation as separate rows. The units are MPa.

Slip parameters

set Number of Slip Systems = 18

Number of Slip Systems

set Latent Hardening Ratio filename = LatentHardeningRatio.txt

A text file including the latent hardening ratio matrix. This file does not have any headers. The size of the matrix is the (total systems×total systems)

set Initial Slip Resistance =0.25,0.25,0.25,10,10,10,10,10,10,10,10,10,15,15,15,15,15,15

This is the initial slip resistances: s_0^α

Main Input file

Slip parameters (Cntd.)

set Initial Hardening Modulus = 5.0, 5.0, 5.0, 100, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 200.0, 200.0, 200.0, 200.0, 200.0, 200.0

Initial hardening moduli of slip systems h_0^α

set Power Law Exponent = 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,1, 1, 1, 1, 1, 1

Power law coefficient in slip rate equation a^β

set Saturation Stress = 185.0, 185.0, 185.0, 160.0, 160.0,160.0, 160.0, 160.0, 160.0, 160.0, 160.0, 160.0, 200.0,200.0, 200.0, 200.0, 200.0, 200.0

Saturation Stress s_s^α

set Slip Directions File = slipDirections.txt

Slip Directions File

set Slip Normals File = slipNormals.txt

Slip Normals File

Uniaxial Compression in Cu with random Orientation

Let's start the simulation!!!

Open the terminal!

Moving to the fcc compression uniaxial folder with random texture:

cd tools/plasticity/applications/crystalPlasticity/fcc/FCC_RandomOrientationBlock/

Open the main input file with your preferred text editor (using vim):

vim prm.prm

Or going to the folder and open the prm.prm file using Visual Studio or Emacs.

This is a cubic of unit length, with 5*8*10 grains, each grain is modeled with an FE mesh.

Let's change the visualization setup, go to line 52, change false to true as:

set Write Output = true

Let's change the visualization skip output steps to generate the visualization results every 25 time steps, go to line 59, change 10000 to 25 as:

set Skip Output Steps = 25

ΔT=0.005; Total time=5

Save the changes and exit. In Vim, you can click on esc, and then type **:wq** and then hit enter.

Uniaxial Compression in Cu with random Orientation

Now, open the BCs file to explore as:

vim boundaryconditions.txt

(you can type a few character of the file and then hit Tab for auto completion)

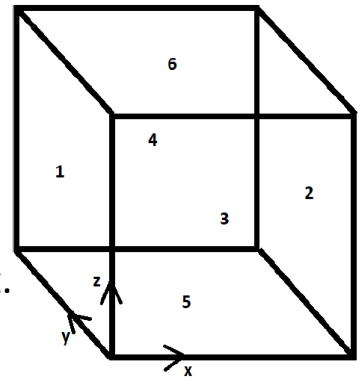
Symmetry boundary conditions are imposed on adjacent x, y, and z faces and the uniaxial compression is applied in the z direction on the opposite z face.

```
# Header lines = 2
# FaceID DoF FinalDisplacement
1 1 0
3 2 0
5 3 0
6 3 -0.05
~
~
```

The final displacement in z direction is -0.05. Considering the unit length of cube, the final strain is -5%.

ΔT=0.005; Total time=5, $\Delta\varepsilon = \frac{0.005\times\%5}{5} = 0.00005$

FaceIDs



Now, start the simulation as:

mpirun -n 2 ../../main prm.prm

Number of processors depends on the capabilities of your VM system.
If you assigned larger number of processors for your VM, you should use it.

Kill the simulation after 50 increments using ctrl+c.

Uniaxial Compression in Cu with random Orientation

Go to the results folder:

cd results

Check the list of files:

ls

stressstrain.txt: file is generated for each step which includes the average Green-Lagrange strain tensor, Cauchy stress tensor, twin activity volume (TwinRealVF), twin volume fraction (TwinMade), and slip activity (SlipTotal).

```
linear system solved in  82 iterations
nonlinear iteration   1 [current residual: 5.47e-01, initial residual: 1.44e+01, relative residual: 3.79e-02]
linear system solved in 131 iterations
nonlinear iteration   2 [current residual: 2.04e-01, initial residual: 1.44e+01, relative residual: 1.42e-02]
linear system solved in 135 iterations
^Cnonlinear iteration   3 [current residual: 1.28e-01, initial residual: 1.44e+01, relative residual: 8.88e-03]
linear system solved in 142 iterations

increment: 52
nonlinear iteration   0 [current residual: 1.44e+01, initial residual: 1.44e+01, relative residual: 1.00e+00]
linear system solved in  80 iterations
yaghoobi@MSE-AllisonZbk:~/plasticity/applications/crystalPlasticity/fcc/FCC_RandomOrientationBlock$ vim prm.prm
yaghoobi@MSE-AllisonZbk:~/plasticity/applications/crystalPlasticity/fcc/FCC_RandomOrientationBlock$ vim prm.prm
yaghoobi@MSE-AllisonZbk:~/plasticity/applications/crystalPlasticity/fcc/FCC_RandomOrientationBlock$ ls
GrainId.txt               boundaryconditions.txt  prm.prm  slipDirections.txt
LatentHardeningRatio.txt  orientations.txt        results  slipNormals.txt
yaghoobi@MSE-AllisonZbk:~/plasticity/applications/crystalPlasticity/fcc/FCC_RandomOrientationBlock$ Cd results
Cd: command not found
yaghoobi@MSE-AllisonZbk:~/plasticity/applications/crystalPlasticity/fcc/FCC_RandomOrientationBlock$ cd results
yaghoobi@MSE-AllisonZbk:~/plasticity/applications/crystalPlasticity/fcc/FCC_RandomOrientationBlock/results$ ls
projectedFields-0024.0000.vtu  projectedFields-0049.0001.vtu  solution-0024.0002.vtu  solution-0049.0003.vtu
projectedFields-0024.0001.vtu  projectedFields-0049.0002.vtu  solution-0024.0003.vtu  solution-0049.0004.vtu
projectedFields-0024.0002.vtu  projectedFields-0049.0003.vtu  solution-0024.0004.vtu  solution-0049.0005.vtu
projectedFields-0024.0003.vtu  projectedFields-0049.0004.vtu  solution-0024.0005.vtu  solution-0049.pvtu
projectedFields-0024.0004.vtu  projectedFields-0049.0005.vtu  solution-0024.pvtu      stressstrain.txt
projectedFields-0024.0005.vtu  projectedFields-0049.pvtu      solution-0049.0000.vtu
projectedFields-0024.pvtu      solution-0024.0000.vtu         solution-0049.0001.vtu
projectedFields-0049.0000.vtu  solution-0024.0001.vtu         solution-0049.0002.vtu
yaghoobi@MSE-AllisonZbk:~/plasticity/applications/crystalPlasticity/fcc/FCC_RandomOrientationBlock/results$
```

solution-N.pvtu: The visualization file of **nodal displacements.** You can open the file with paraview. **N** is increment number that output is generated. The .vtu files are small block building the .pvtu file which is generated for each processor for specific time step.

projectedFields-N.pvtu: The visualization file you need to open with paraview for **variables defined in the outputs section of prm.prm**. **N** is increment number that output is generated. The .vtu files are small block building the .pvtu file which is generated for each processor for specific time step. Unlike the solution (nodel displacements) which are directly obtained for each node, these variables were calculated for each integration point. The nodal values need then to be calculated by solving a linear equation.

Uniaxial Compression in Cu with random Orientation

I put the results in the following folder as well:

/home/icme/tools/plasticity/Training_Materials/ICME_Workshop_2021/FCC_RandomOrientationBlock/results

Uniaxial Compression in Cu with random Orientation

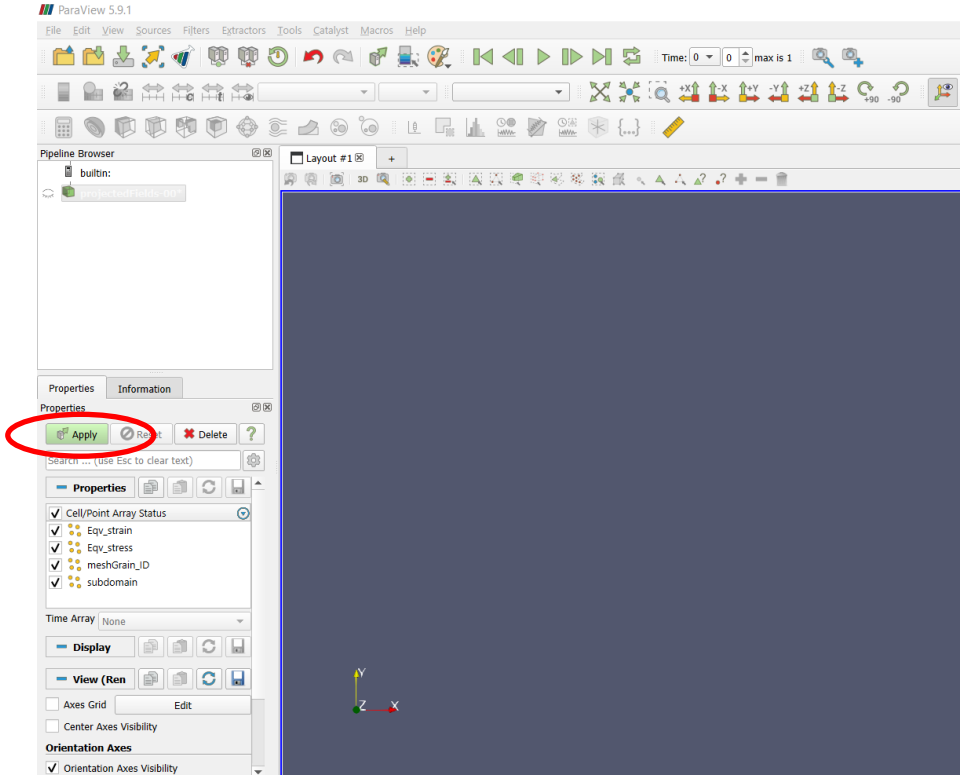
To visualize the outputs from CPFE we can go through the following steps using **Paraview**.

Find the Paraview icon on desktop and double click on it.

The address for results are in the folder:
/home/icme/tools/plasticity/applications/crystalPlasticity/fcc/FCC_RandomOrientationBlock/results

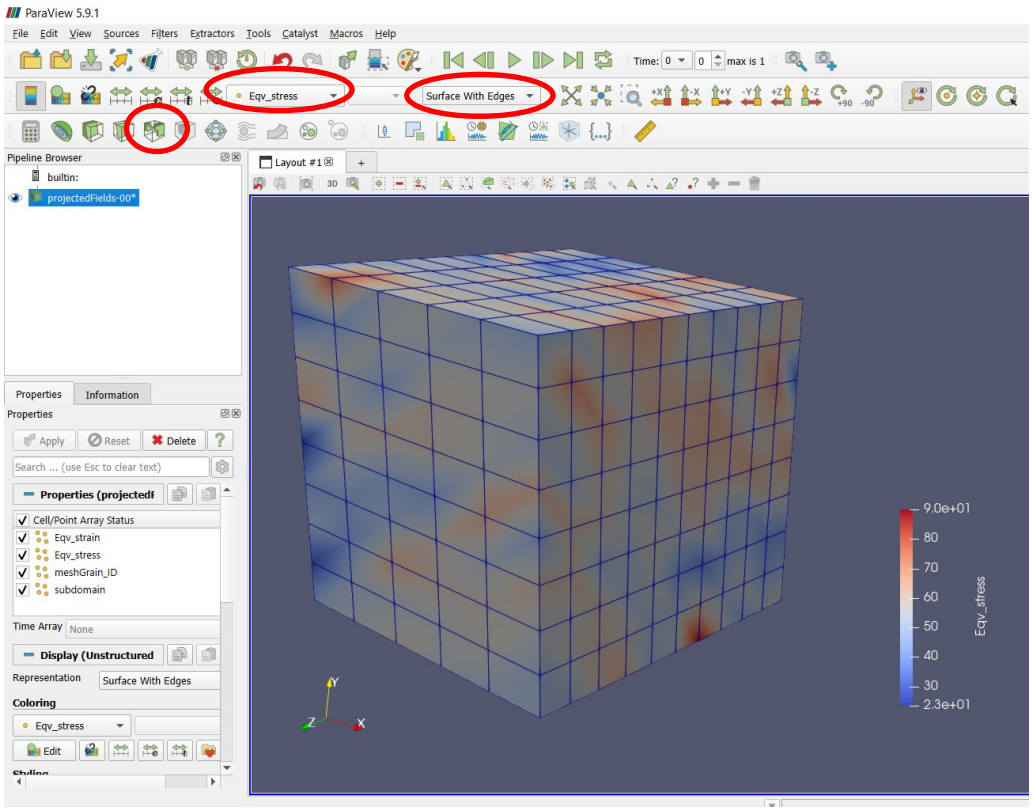
- Then click on **File-> Open**, and choose the file with **projectedFields-...pvtu** extension that you wish to visualize. This will load the file into Paraview.

- The filename is visible in a small section to the left hand side of the window. Click on **Apply** in the properties and make sure that the file is visible.



Uniaxial Compression in Cu with random Orientation

In the toolbar, there is a drop down menu that reads **Solid Color**. Click on that menu to choose a particular variable of choice. Adjacent to it is another drop down menu that reads **Outline**. Click on it and choose **Surface With Edges**. The variation of that variable in space is now visualized with a colorbar for reference. The viewer should look something like below.



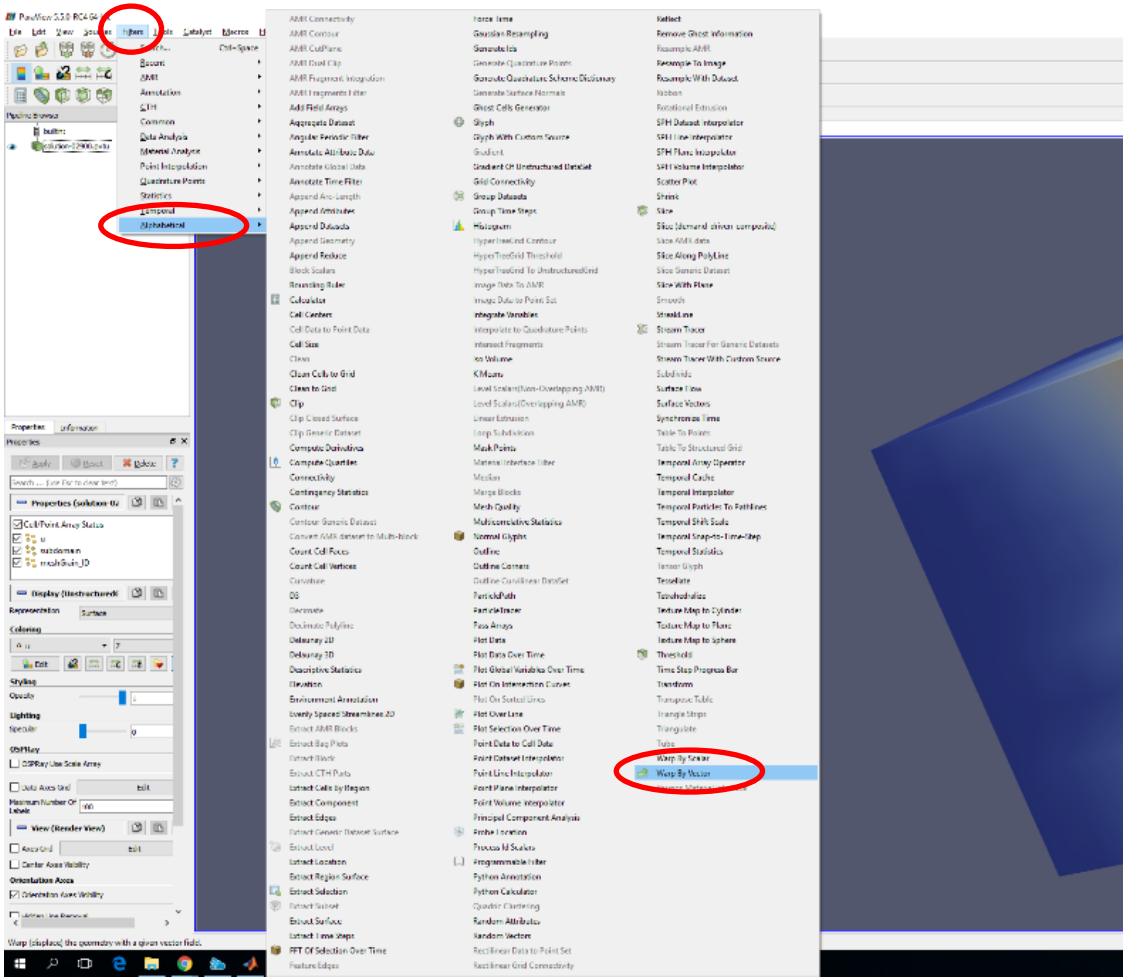
- The same can be repeated for other variables as well.

- Now you can delete this data by right clicking on the name of projectedFields-00* and select delete.

Uniaxial Compression in Cu with random Orientation

Visualizing a deforming mesh

- The deformed mesh can be visualized using the **solution-...pvtu** files and the Warp By Vector filter in Paraview by following the steps below.
- Load Paraview and load a **solution-...pvtu** file, and then click on apply to open it.
- Choose the field variable as **u**(for displacement), the **Z** component and select **Surface** for representation.
- Click on the **Filters** dropdown in the menu bar, click on **Alphabetical** and click on the **Warp By Vector filter**.



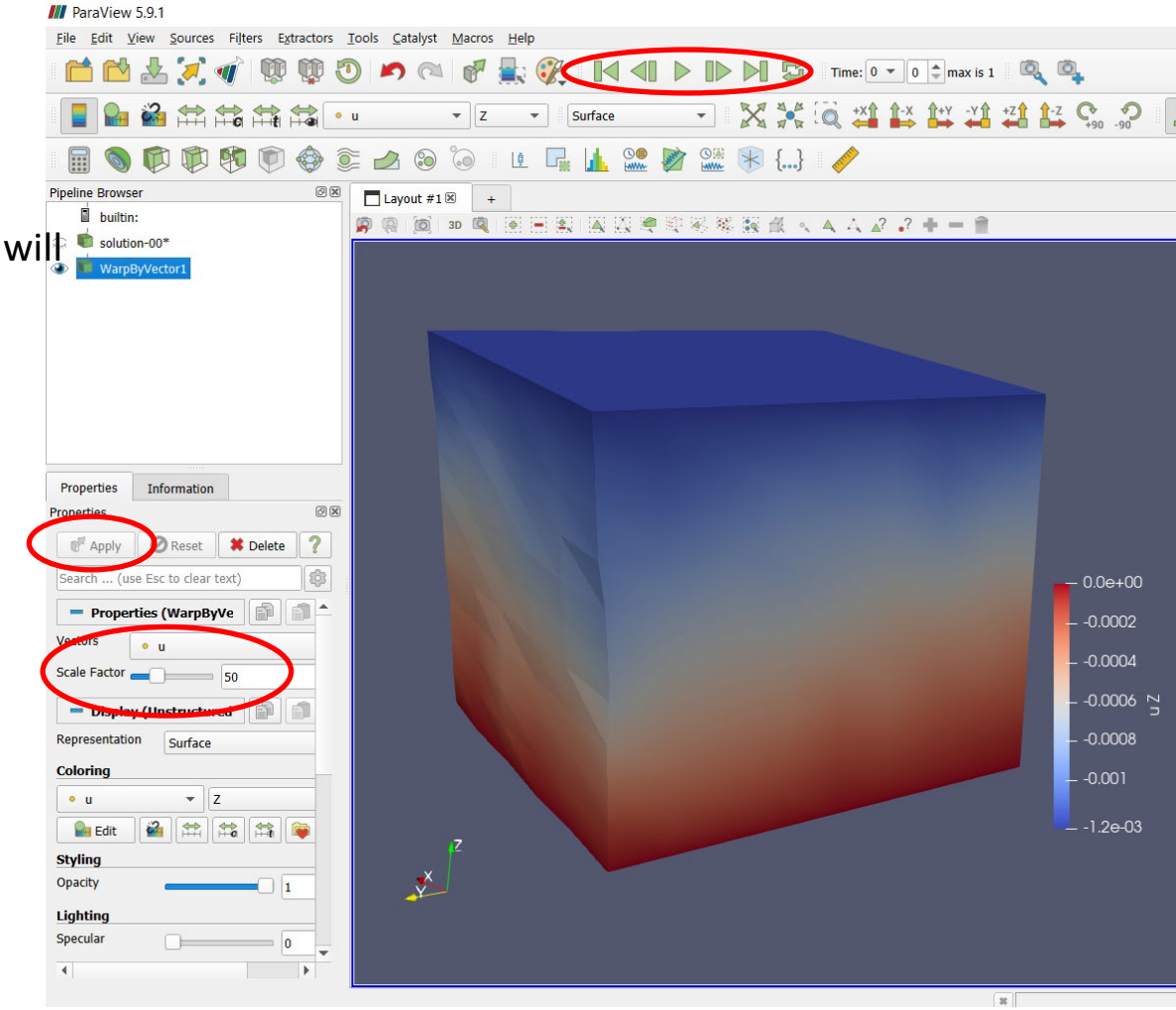
Uniaxial Compression in Cu with random Orientation

Visualizing a deforming mesh (Cntd.)

- This appears as a filter in the pipeline below the solution file. Then click on **Apply** button, and the deformed mesh will be visible in the viewer.

- Change the **Scale Factor** to 50.

- You can march in time by clicking on the **green arrow** (if you have more than one steps output were generated).



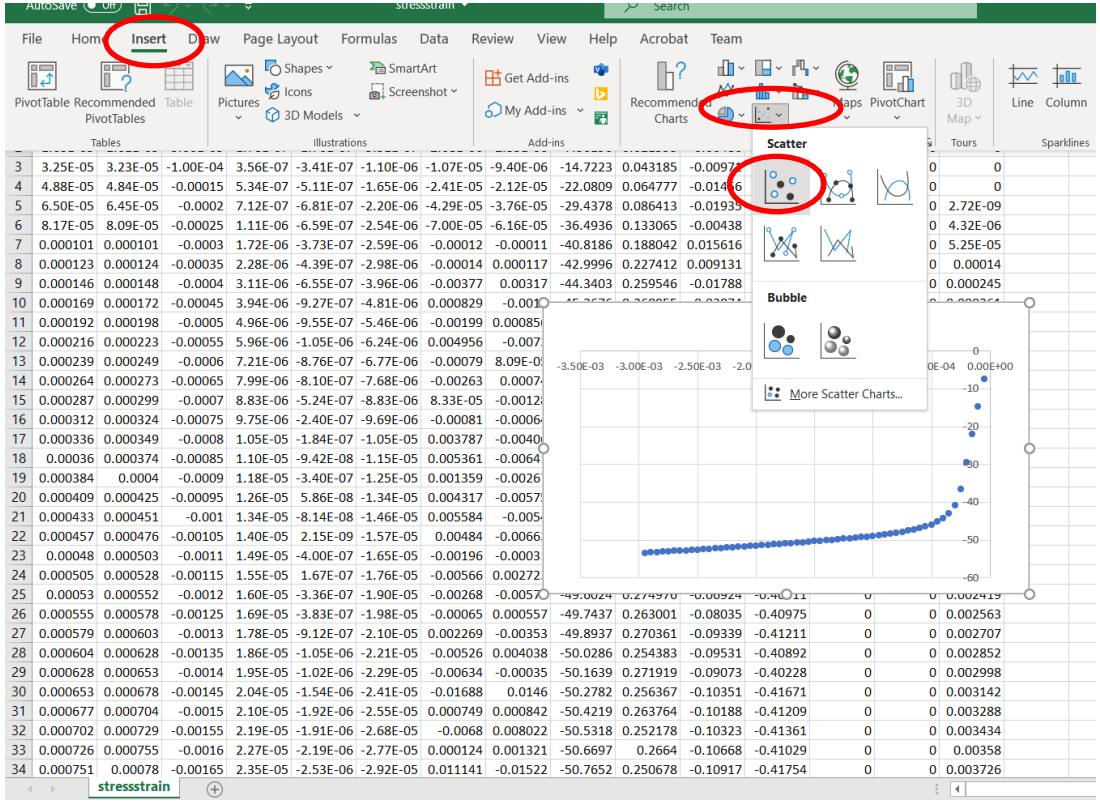
Uniaxial Compression in Cu with random Orientation

Stress-strain curve

- Open MS Excel.
- Click on File -> Open, and then choose the file **stressstrain.txt**. You might need to change the default from **All Excel Files** to **All Files** in the search box.

.

- Text Import wizard will be shown next. Click on Finish.
- Select the Ezz column (clicking on the column C) hold ctrl button and select the Tzz column by clicking on the I column. Both columns should be selected now.



- Click on the **Insert** tab, choose the **Scatter** option and **Scatter** suboption.
- You can right click on the plot and **move** it into a **new Sheet**.
- In the case of ubuntu libreoffice calc, you can select insert->Chart->XY (Scatter).

Uniaxial Compression in Cu with random Orientation

Using the microstructure which we generated using EBSD input and DREAM3D:

1) Open a terminal

2) Go to the directory of the example

cd tools/plasticity/applications/crystalPlasticity/fcc/FCC_RandomOrientationBlock/

2) Remove the available microstructure input files: *rm orientations.txt*
rm GrainId.txt

4) Copy the microstructure input file we generated into the folder:

scp ../../../../Training_Materials/Pre-Processing/FCC/orientations.txt ./

scp ../../../../Training_Materials/Pre-Processing/FCC/grainID.txt ./

5) Edit the main input file (prm.prm):

Lines 167,170, 173 -> Change the number of voxels to **32**.

Line 176-> Change the grainID name to **grainID.txt**.

Line 179-> Change the number of header lines to **20**.

Lines 37, 40, 43-> Change the number of Subdivisions to **32** (Every voxel is modeled using an FE element).

Line 52 set write output = **true** if it is not. *Line 59* set Skip output to **1**. (**Save the prm.prm!)**

5) remove the current results folder by typing in the terminal: rm -r results

6) Run the example for one increment (killing the run by *ctrl+c*): *mpirun -n 2 ../../main prm.prm*

7) Open the results using Paraview to see the microstructure.

Uniaxial Compression in Cu with random Orientation

Modifying the input files (Possible extension):

1) Change the loading to uniaxial tension.

2) Use a more refined mesh:

set Refine factor = 2
set Skip Output Steps = 1

3) Output QuadratureOutput and investigate the output file with just grainID, position and orientation:

set Refine factor = 0
set Skip Output Steps = 25
set Write Quadrature Output = true
set Skip Quadrature Output Steps = 25

If you are in the FCC_RandomOrientationBlock folder as your current directory:

vim ../../../../src/materialModels/crystalPlasticity/updateAfterIncrement.cc

Commenting all lines in the QuadratureOutput region except those you want:

Recompile the code using "*make release*" in the plasticity/applications/crystalPlasticity/fcc folder.

cd fcc/FCC_RandomOrientationBlock
mpirun -n 2 ../../main prm.prm