PRISMS-Plasticity: An Open Source Crystal plasticity FE Code (Session2)

Mohammadreza Yaghoobi [1], Aaditya Lakshmanan [2]

[1] Department of Materials Science and Engineering, University of Michigan, Ann Arbor
[2] Department of Aerospace Engineering, University of Michigan, Ann Arbor

PRISMS Center Training Session
August 10, 2021

Uniaxial Compression in Cu with random Orientation

Playing with the input files:

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 and position:

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 6 ../../main prm.prm

Uniaxial Compression in Cu with random Orientation

Playing with the input files:

4) Applying a cyclic loading using the Tabular BCs.

set Use Simple BCs = false

set Use Tabular BCs = true

set Tabular Boundary condition filename = boundaryconditions.txt

set Number of time data for Tabular BCs = 7

set Number of tabular boundary conditions = 4

set Tabular Time Table = 0,0.3,0.9,1.5,2.1,2.7,3

boundaryconditions.txt

```
1 1 0 0 0 0 0 0 0
3 2 0 0 0 0 0 0 0
5 3 0 0 0 0 0 0 0
6 3 0 -0.003 0.003 -0.003 0.003 -0.003 0
```

set Total time = 3

set Skip Output Steps = 20

I made the simulation shorter in each cycle (strain amplitude of 0.3%) to make the simulation faster. Let's do the simulation.

Uniaxial Compression in Cu with random Orientation

Adding another variable to the visualization:

Adding Tzz as an ouput variable.

set Output Variable 1= true

Edit the updateAfterIncrement.cc line 125 to:

this->postprocessValues(cellID, q, 3, 0) = T[2][2];

Recompile the code.

Open the projectedFields-0024.pvtu using Paraview and check the variables. You should have the **outputVar1** which is Tzz.

Microstructure generation

- We start with the Dream3D to generate a microstructure from EBSD and study its response using PRISMS-Plasticity
- We have a pipeline prepared for Dream3D to generate the input file required for PRISMS-Plasticity in:

  /Training_Materials/Pre-Processing/fcc

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

Microstructure generation

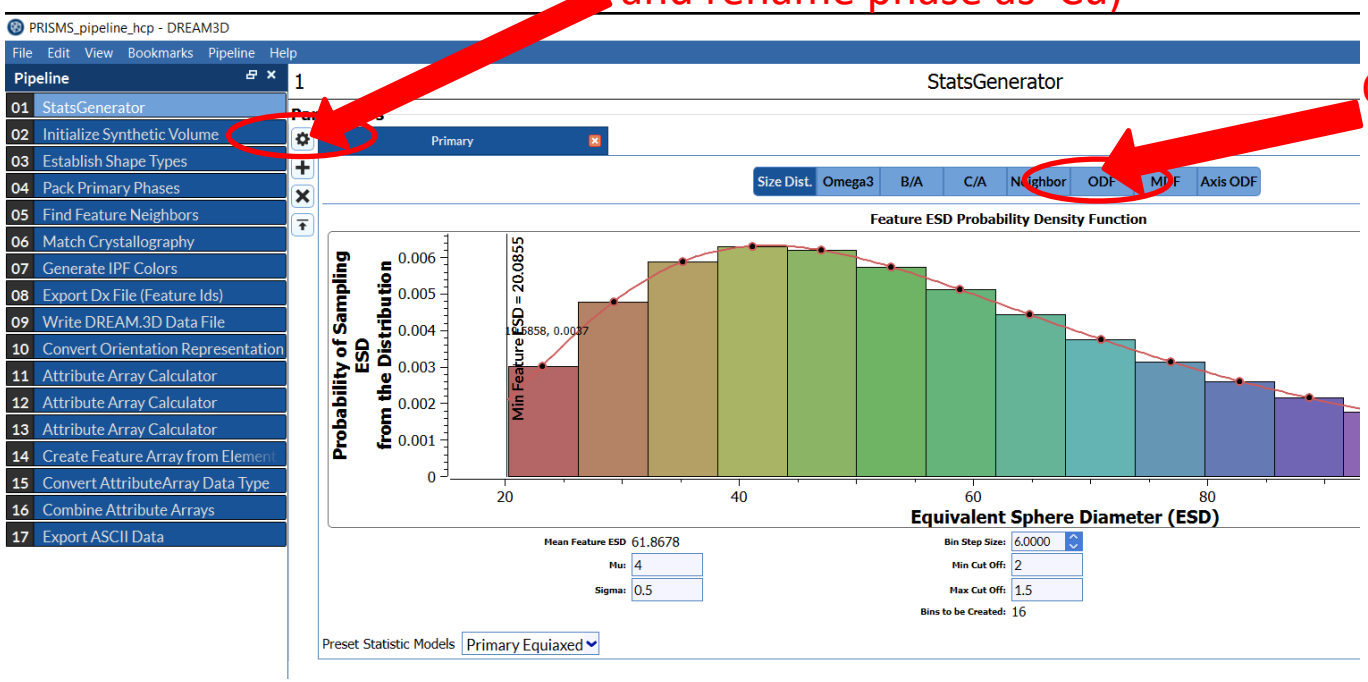
- This will open a prepared pipeline which can generate the PRISMS-Plasticity input files.

- 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:

Loading from EBSD experiment

*PRISMS_pipeline_hcp - DREAM3D

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 Representation
11 Attribute Array Calculator
12 Attribute Array Calculator
13 Attribute Array Calculator
14 Create Feature Array from Element
15 Convert AttributeArray Data Type
16 Combine Attribute Arrays
17 Export ASCII Data

Start Pipeline

1 StatsGenerator

Parameters

Primary

Size Dist. | Omega3 | B/A | C/A | Neig | ODF | MDF | Axis ODF

Mannual Entry | Bulk Load From File 1

2

Input File: Select...

Values are in Degrees

Angle Representation: Euler

Value Delimiter: Space

3 Load Data

<0001> <10$\bar{1}$0> <2$\bar{1}\bar{1}$0>

Image Size: 200 Image Layout: Horizontal Type: Discrete

4 Preset Statistic Models: Primary Equiaxed

5 Create Data Reset...

Created Objects

Created Data Container

Statistics Data Container Name: StatsGeneratorDataContainer

Created Ensemble AttributeMatrix

Cell Ensemble Attribute Matrix Name: CellEnsembleData

Created Ensemble Arrays

Statistics Array Name: Statistics

Crystal Structures Array Name: CrystalStructures

After selecting all the options, click on create Data.

- From select, open the file Training_Materials\Pre-Processing\fcc\copperdata1. Click on Load Data, then click on refresh, and finally 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
```

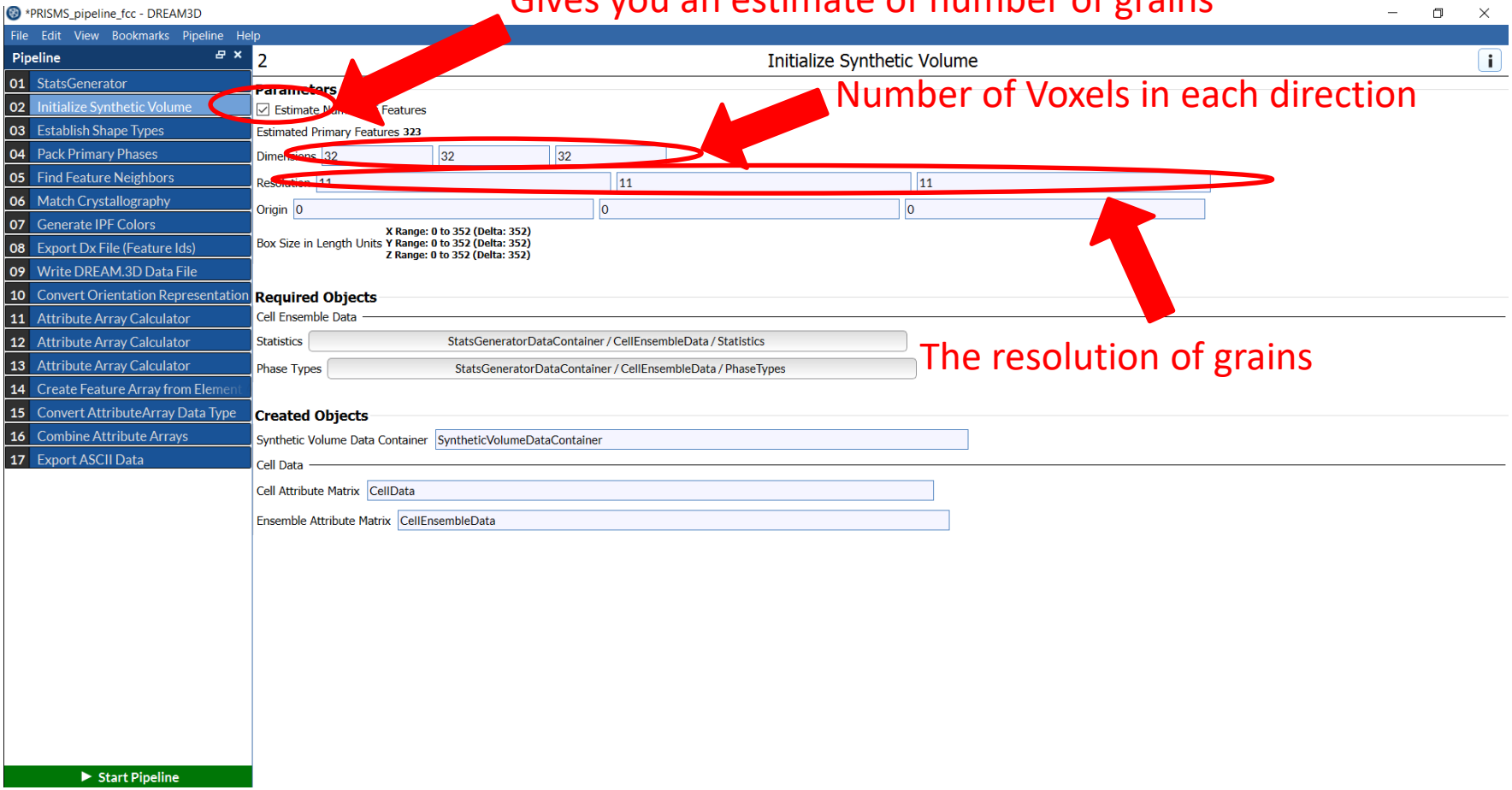
Microstructure generation

02 Initialize Synthetic volume: Here, you define the number of voxels in each direction and resolution of the grains.

Gives you an estimate of number of grains

Number of Voxels in each direction

The resolution of grains



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.

Two outputs will be grainID.txt and orientations.txt.

Microstructure generation

grainID.txt:

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

- Number of header lines: 20
- Number of Voxel in each direction: 32*32*32

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

Rate-independent formulation

We want to use the generated microstructures using the rate-dependent crystal plasticity model.

To do so, first we need to copy the following files: plasticity/src/materialModels/crystalPlasticity/MaterialModels/RateDependentModel/calculatePlasticity.cc and userFunctions.cc and paste it in plasticity/src/materialModels/crystalPlasticity.

Then we need to recompile the code by going to the folder: plasticity/applications/crystalPlasticity and type:

Make release

Sometimes, by just doing make release, the updated file will not be compiled and the code still uses the previous version. To make sure that the code compiling with the recent version, you can temporarily move the updated files from the folder (calculatePlasticity.cc and userFunctions.cc) and type *make release*. It will give you errors. And then move those file back in the folder and recompile. This should work.

Rate-independent formulation

Now, we'll copy the generated microstructure files of grainID.txt and orientations.txt to the folder plasticity/applications/crystalPlasticity/fcc/FCC_Random_RateDependent.

We need to update the prm.prm file in the following fields:

FE mesh: change to 32by32by32 (using set Subdivisions)

Set Output: True, and skip output steps of 5

Total time=5 DeltaT=0.005

Voxels in x,y,z=32; GrainID Filename=grainID.txt; Number Header lines=20

Rate-independent formulation

We're using some of the features useful for User Defined Crystal plasticity models:

User Material Constants: If you need to define new material constants in your model, you can use this field.

User Material State Variables: If you need to define new state variables in your model, you can use this field. You can also define the initial value for your sate variables.

$\dot{\gamma}_0$ m $cr1$ $cr2$ $Max1$ $Max2$

set User Material Constants 1 = 1.0e-3, 0.04 , **1.0e-4 , 1.0e-4** , 10000 , 10000, $\underbrace{0, 0, 0, 0 , 0, 0}_{\text{Backstress parameters}}$

$$\dot{\gamma}^{\alpha} = \dot{\gamma}_0 \left| \frac{\tau^{\alpha} - \chi^{\alpha}}{s^{\alpha}} \right|^{m} \operatorname{sign}(\tau^{\alpha} - \chi^{\alpha})$$

$\dot{\gamma}^{\alpha}$:shearing rate $\dot{\gamma}_0$ reference shearing rate s^{α} slip resistance χ^{α} back stress

The rate-dependent model is numerically solved using two iterative schemes, one of them is located inside the other one. Each of these iterative schemes has its own threshold of error (cr1 and cr2) and maximum number of iterations (Max1 and Max2).

Implementing your own crystal plasticity model

The only files you need to update are:
plasticity/src/materialModels/crystalPlasticity/MaterialModels/RateDependentModel/calculatePlasticity.cc and userFunctions.cc

calculatePlasticity.cc: The main implementation.

userFunctions.cc: Auxiliary files you need to use in the calculatePlasticity.cc.

Documentation for user defined crystal plasticity models: https://github.com/prisms-center/plasticity/blob/master/docs/User%20defined%20material%20model%20manual.pdf

Let's briefly go over the calculatePlasticity.cc of the rate-dependent model as an example to see how it looks like.

Periodic BCs

To use the periodic BCs, it is better that the generated microstructure is periodic as well. To use the periodic BCs, one cannot use ***set Refine factor*** to refine the mesh.

C.P. Przybyla, "Microstructure-sensitive extreme value probabilities of fatigue in advanced engineering alloys." PhD Thesis, Georgia Institute of Technology (2010) section 5.5.

This is crucial in calculations of fatigue resistance when smaller microstructure instantiations are employed that represent bulk, i.e., subsurface, material response. Przybyla (2010) (See the references) presented the details of the periodic BCs (PBCs) required to mimic the subsurface microstructure during the fatigue simulation. In PRISMS-Fatigue framework, **PBCs are implemented using a set of linear constraints**.

These constraints are defined on three components of opposing Faces, parallel Edges, and Vertices. Rigid body motion is also excluded by assigning additional constraints. Three sets of constraints are applied on nodes on opposing Faces (in the direction perpendicular to those Faces), nodes on each set of four parallel Edges (in the two directions perpendicular to the direction in which the Edges are parallel), and Vertices (in all directions). The contraction or expansion of the dimensions of the actual microstructure volume are considered.

Periodic BCs

Table 5.1: Description of symbols used to describe nodes sets at vertices and faces.

Symbol	Description
REF	Reference node located at $x = 0$, $y = 0$ & $z = 0$
V000	Node at vertex located at $x = 0$, $y = 0$ & $z = 0$
V001	Node at vertex located at $x = 0$, $y = 0$ & $z = h$
V010	Node at vertex located at $x = 0$, $y = h$ & $z = 0$
V100	Node at vertex located at $x = h$, $y = 0$ & $z = 0$
V011	Node at vertex located at $x = 0$, $y = h$ & $z = h$
V101	Node at vertex located at $x = h$, $y = 0$ & $z = h$
V110	Node at vertex located at $x = h$, $y = h$ & $z = 0$
V111	Node at vertex located at $x = h$, $y = h$ & $z = h$
FXP	Face nodes excluding edges and vertices for surface normal to x axis at $x = h$
FXN	Face nodes excluding edges and vertices for surface normal to x axis at $x = 0$
FYP	Face nodes excluding edges and vertices for surface normal to y axis at $y = h$
FYN	Face nodes excluding edges and vertices for surface normal to y axis at $y = 0$
FZP	Face nodes excluding edges and vertices for surface normal to z axis at $x = h$
FZN	Face nodes excluding edges and vertices for surface normal to z axis at $x = 0$

Periodic BCs

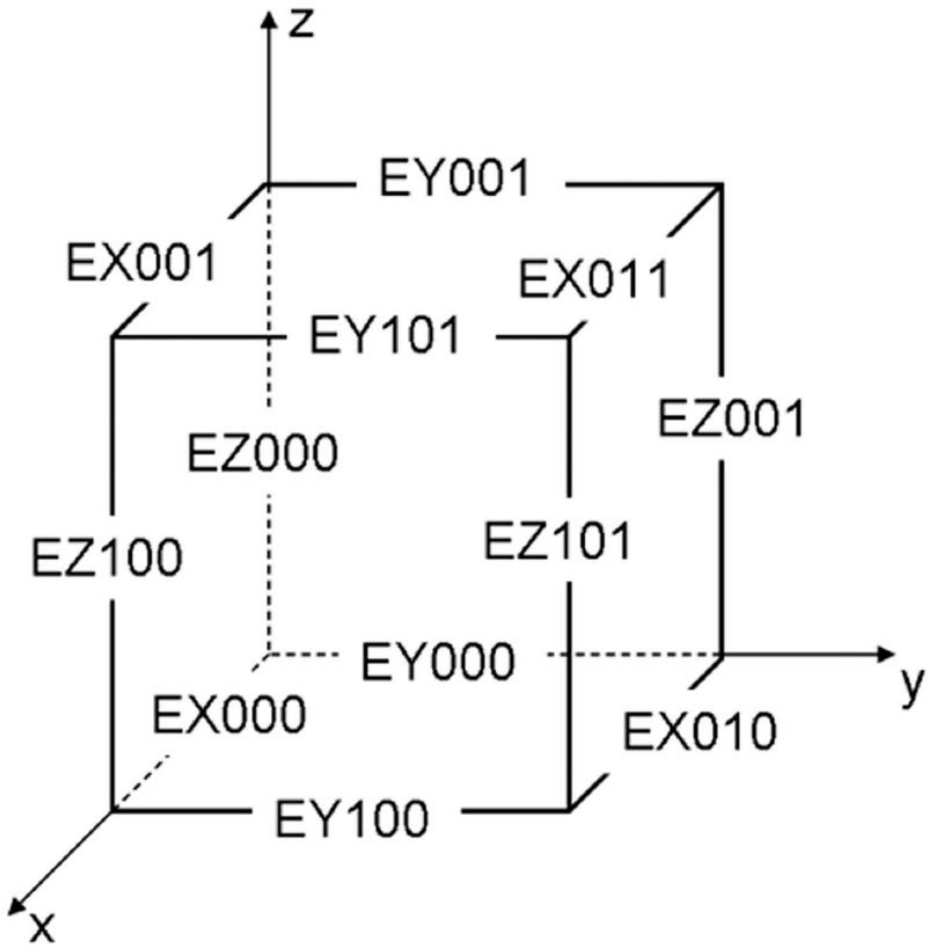


Figure 5.9: Location of edge node sets. Note that these edge node sets do not include the nodes at the vertices of the model.

Periodic BCs (Main Input file)

set Use Simple BCs = false

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

set Use Periodic BCs = true

Flag to indicate whether to use Periodic BCs.

set Periodic Boundary condition Constraint filename = PeriodicBCsConstraints.txt

The name of the file which include the constraints between the Vertices, Edges, and Faces. For a specific type of BCs, this should not be changed. For example, if one models the fully periodic BCs with different loading conditions, this file should not be changed.

This file includes the linear constraints for Vertices, Edges, and Faces. For each constraint, the file has two lines, first one for DOFs and second one for their corresponding coefficients. The convention for the DOFs are defined in section 5.5 of Przybyla PhD Thesis (2010).

Periodic BCs (Main Input file)

For the Vertices constraints, each row has 5 columns. The first column defines the number of DOFs involves in this constraint. All remaining 4 columns correspond to the DOFs of Vertices. In the case of Edges and Faces, we only have the latter 4 columns. In the cases of constraints for Edges, the first two columns correspond to the DOFs of the Edges, while the third and fourth columns correspond to the Vertices. In the case of Faces, the first two columns correspond to the DOFs of the Faces, while the third and fourth columns correspond to the Vertices. DOFs convention for Vertices, Edges, and Faces used here are described in the file as headers.

```
# Vertices Constraints: Each line has two rows, one for DOFs and one for their corresponding coefficient.
#set Vertices Periodic BCs row order:
# 0=V000_1 1=V000_2 2=V000_3 3=V100_1 4=V100_2 5=V100_3 6=V010_1 7=V010_2 8=V010_3 9=V001_1 10=V001_2 11=V001_
# 12=V110_1 13=V110_2 14=V110_3 15=V101_1 16=V101_2 17=V101_3 18=V011_1 19=V011_2 20=V011_3 21=V111_1 22=V111_
23=V111_3
4 12 6 3 0
4 1 -1 -1 1
……..
```

```
# Edges Constraints: Each line has two rows, one for DOFs and one for their corresponding coefficient.
#set Edges Periodic BCs row order:
# 0=EX000_1 1=EX000_2 2=EX000_3 3=EX001_1 4=EX001_2 5=EX001_3 6=EX010_1 7=EX010_2 8=EX010_3 9=EX011_1
10=EX011_2 11=EX011_3
# 12=EY000_1 13=EY000_2 14=EY000_3 15=EY001_1 16=EY001_2 17=EY001_3 18=EY100_1 19=EY100_2 20=EY100_3
21=EY101_1 22=EY101_2 23=EY101_3
# 24=EZ000_1 25=EZ000_2 26=EZ000_3 27=EZ010_1 28=EZ010_2 29=EZ010_3 30=EZ100_1 31=EZ100_2 32=EZ100_3
33=EZ110_1 34=EZ110_2 35=EZ110_3
3 9 18 9
-1 1 -1 1
…….
```

Periodic BCs (Main Input file)

For the Vertices constraints, each row has 5 columns. The first column defines the number of DOFs involves in this constraint. All remaining 4 columns correspond to the DOFs of Vertices. In the case of Edges and Faces, we only have the latter 4 columns. In the cases of constraints for Edges, the first two columns correspond to the DOFs of the Edges, while the third and fourth columns correspond to the Vertices. In the case of Faces, the first two columns correspond to the DOFs of the Faces, while the third and fourth columns correspond to the Vertices.

```
# Faces Constraints: Each line has two rows, one for DOFs and one for their corresponding coefficient.
#set Faces Periodic BCs row order:
# 0=FXN_1 1=FXN_2 2=FXN_3 3=FXP_1 4=FXP_2 5=FXP_3 6=FYN_1 7=FYN_2 8=FYN_3 9=FYP_1 10=FYP_2 11=FYP_3
# 12=FZN_1 13=FZN_2 14=FZN_3 15=FZP_1 16=FZP_2 17=FZP_3
# Important note: The first column of all face constraint lines must always start with the positive faces, i.e., FXP,FYP, and FZP.
3 0 3 0
1 -1 -1 1
……
```

Important note: The first column of all **Faces** constraint lines must always start with the positive Faces, i.e., FXP,FYP, and FZP.

Periodic BCs (Main Input file)

Vertices:
V110$_x - V010_x - V100_x + V000_x = 0$
In the periodic BCs file, this becomes:
4 12 6 3 0
4 1 -1 -1 1

Edges:
-EX001$_x + EX011_x - V011_x + V001_x = 0$
In the periodic BCs file, this becomes:
3 9 18 9
-1 1 -1 1

One should note that we named this section Edges to show that it includes the DOFs of the Edges. However, the last two columns belong to the Vertices DOFs.

Faces:
-FXN$_x + FXP_x - V100_x + V000_x = 0$
As mentioned earlier, the first column of all face constraint lines must always start with the positive Faces, i.e., FXP,FYP, and FZP.
So we first rearrange the equation as follows:
FXP$_x - FXN_x - V100_x + V000_x = 0$
In the periodic BCs file, this becomes:
3 0 3 0
1 -1 -1 1

One should note that we named this section Faces to show that it includes the DOFs of the Faces. However, the last two columns belong to the Vertices DOFs.

Periodic BCs (Main Input file)

The number of linear constrains for Vertices, Edges, and Faces are defined next.

```
set Number of Vertices Constraints  = 12
set Number of Edges Constraints  = 14
set Number of Faces Constraints = 6
```

The above numbers describes the sample with periodic BCs in 2 directions and free surface in the third one. In the case of fully periodic BCs, the numbers become as follows:

```
set Number of Vertices Constraints = 12
set Number of Edges Constraints  = 27
set Number of Faces Constraints  = 9
```

Periodic BCs (Main Input file)

The next part defines the BCs which are applied to the Vertices. The notation used here is in line with the **section 5.5** of Przybyla PhD Thesis (2010). The first line control which Vertices DOF the BCs are applied to by assigning the value of 1. Otherwise, the value is 0 and that DOF has no predefined BCs. The second line define the values for BCs at the end of the total time.

```
# 0=V000_1;1=V000_2;2=V000_3; 3=V100_1;4=V100_2;5=V100_3; 6=
  V010_1;7=V010_2;8=V010_3; 9=V001_1;10=V001_2;11=V001_3;
# 12=V110_1;13=V110_2;14=V110_3; 15=V101_1;16=V101_2;17=V101_3
  ; 18=V011_1;19=V011_2;20=V011_3; 21=V111_1;22=V111_2;23=
  V111_3;
set Vertices Periodic BCs row 1 = 1,1,1, 1,1,1, 1,0,1, 1,1,0,
  1,0,0, 1,0,0, 0,0,0, 1,0,0
```

```
set Vertices Periodic BCs row 2 = 0,0,0, 0.05,0,0, 0,0,0,
  0,0,0, 0.05,0,0, 0.05,0,0, 0,0,0, 0.05,0,0
```

Periodic BCs (Main Input file)

One can also define the periodic BCs along Tabular loading. First, this flag should be set to true:

```
set Use Tabular Periodic BCs = true
```

Next, the second line of the applying BCs in the input file lines, i.e., **set Vertices Periodic BCs row 2**, will be the applied BCs at the time of the periodic Tabular time, which is defined as follows:

```
set periodic Tabular time = 10
```

This is to define the **base rate** at which the BCs are applied. In other words, the **base rate** of BCs is equal to **(Vertices Periodic BCs row 2)/(periodic Tabular time)** for each DOF of Vertices.

For example, here, for the DOF of **3=V100**$_1$, we defined the value of 0.05 in **set Vertices Periodic BCs row 2**. The **base rate** can then be calculated as:

base rate=0.05/(set periodic Tabular time)=0.05/10=0.005.

Periodic BCs (Main Input file)

Finally, the Tabular BCs is applied using Tabular Periodic Time Table and their coefficients as follows:

```
set Tabular Periodic Time Table = 0,10,30,50,70,90,110
set Tabular Periodic Time Table Coefficient = 1,-1,1,-1,1,-1,1
```

The first line defines the time table, and second line defines the corresponding coefficients for each of these time spans. For each time span (i.e., between each two times inside the **Time Table**), the rate of deformation is applied as **(Corresponding Coefficient*base rate)**.

For example, the above values for these two lines state that:
Time$_1 = 0; Time_2 = 10; Time_3 = 30; Time_4 = 50; Time_5 = 70; Time_6 = 90; Time_7 = 110;$

The corresponding coefficients are:
Coefficient$_1 = 1; Coefficient_2 = -1; Coefficient_3 = 1; Coefficient_4 = -1; Coefficient_5 = 1;$
$Coefficient_6 = -1; Coefficient_7 = 1;$

For the first time span, the applied BCs between *(Time$_1$=0)* to *(Time$_2$=10)* has the rate of:

*(Coefficient$_1$=1)**(**base rate)**

From *(Time$_2$=10)* and *(Time$_3$=30)*, the rate is the *(Coefficient$_2$=-1)**(**base rate)**.

This tabular loading defines a cyclic loading with equal max compression and tension strains.

Periodic BCs (Simulation)

Let's try a simulation set (rate_independent)!!!

Change the directory to the folder fcc/periodicBCs

Running **prm_UniaxialCyclicTabular.prm**

The microstructure is 32*32*32-> Let's have the FE mesh 8*8*8 (for the sake of time) for 6 processors, if you have less processors, try 4*4*4!

set Skip Output Steps = 10

I made the simulation shorter in each cycle (strain amplitude of 0.3%) to make the simulation faster. Let's do the simulation.

set Vertices Periodic BCs row 2 = 0,0,0, 0.002,0,0, 0,0,0, 0,0,0, 0.002,0,0, 0.002,0,0, 0,0,0, 0.002,0,0

set periodic Tabular time = 0.2

set Tabular Periodic Time Table = 0,0.2,0.6,1,1.4,1.8,2.2

set Total time = 2.2

Finally, this prm file is for rate_independent model. Let's modify it for rate_dependent.

Periodic BCs (Simulation)

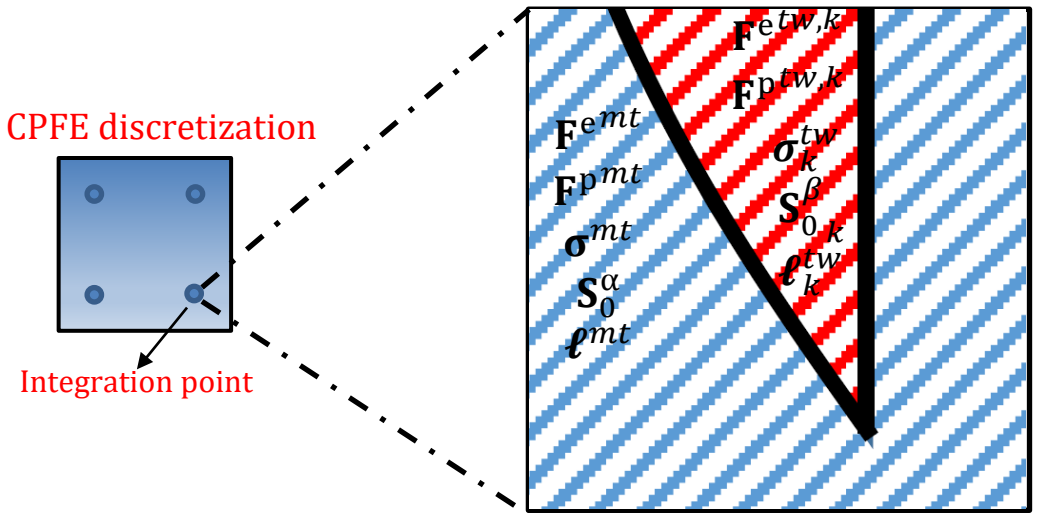
Let's modify **prm_UniaxialCyclicTabular.prm** file for rate_dependent by adding these lines before elastic stiffness:

```
# Flag to indicate if User Material Model is enabled
set Enable User Material Model                = true

# Flag to indicate if User Material Model is enabled Phase 1
set Enable User Material Model 1              = true

# Number of User Material Constants in a Material model Phase 1
set Number of User Material Constants 1       = 12

# Number of User Material State Variables in a Material model Phase 1
set Number of User Material State Variables 1     = 62

# Material Constants in a Material model Phase 1
set User Material Constants 1         = 1.0e-3, 0.04 , 1.0e-4 , 1.0e-4 ,  10000 , 10000,  0, 0, 0, 0 , 0.0, 0.0

# Material State Variables in a Material model Phase 1
set User Material State Variables Initial Values 1     = 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0
```

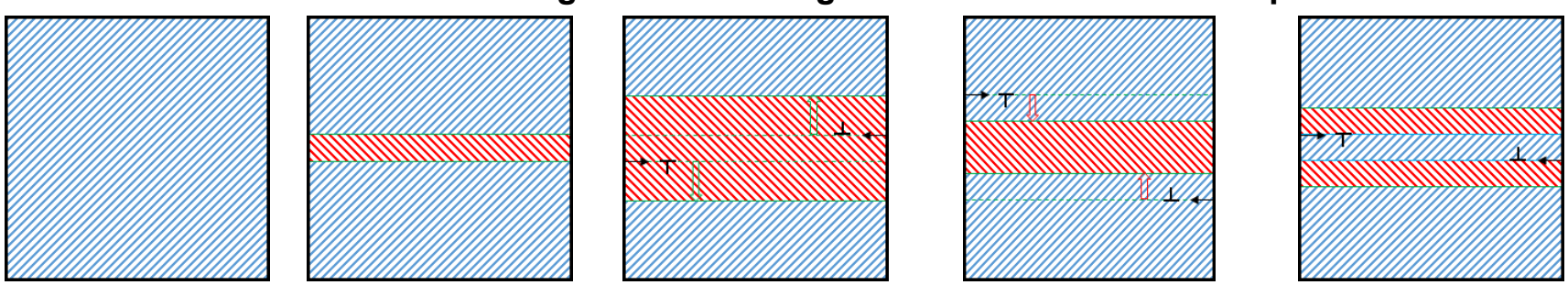
New CPFEM twinning-detwinning model

- The response of the material point is homogenized using Taylor model.
- Twinning and detwinning mechanisms can capture twin nucleation, twin growth, twin shrinkage, and detwinning.
- Multiple twin variants can be activated inside the material point.

Schematic of a material point with twin



Schematic of the twinning and detwinning mechanisms in a material point



twin-free | **Initial twin nucleation** | **twin growth due to the child propagation** | **twin shrinkage due to the parent propagation** | **detwinning**

New CPFEM twinning-detwinning model

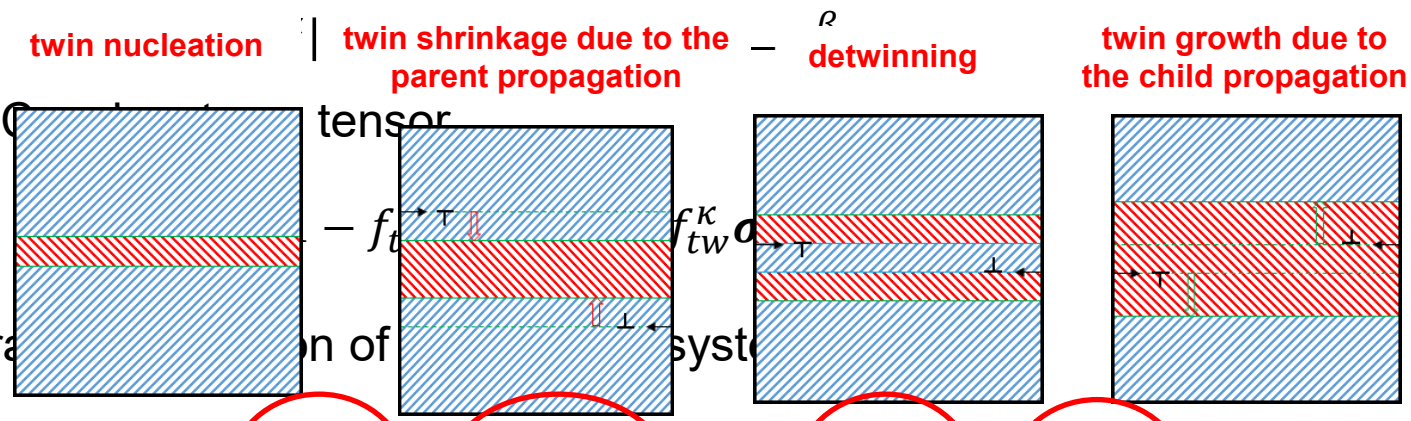
- Multiplicative decomposition:

$$\mathbf{F} = \mathbf{F}^{\mathrm{e}mt}\mathbf{F}^{\mathrm{p}mt} = \mathbf{F}^{\mathrm{e}tw,k}\mathbf{F}^{\mathrm{p}tw,k}$$

- Key idea of crystal plasticity

$$\mathbf{L}^{\mathrm{p}mt} = \sum_{\alpha=1}^{N_s^{mt}+N_t^{mt}} \dot{\gamma}^{\alpha}\mathbf{S}^{\alpha}\,\mathrm{sign}(\tau^{\alpha}), \qquad \mathbf{L}^{\mathrm{p}tw,k} = \sum_{\beta=1}^{N_s^{tw}+N_t^{tw}} \dot{\bar{\gamma}}_{\kappa}^{\beta}\mathbf{S}_k^{\beta}\,\mathrm{sign}\left(\tau_k^{\beta}\right)$$

- Yield surface



- Homogenized C[illegible] tensor

$$- f_t[illegible] f_{tw}^{\kappa}\boldsymbol{\sigma}$$

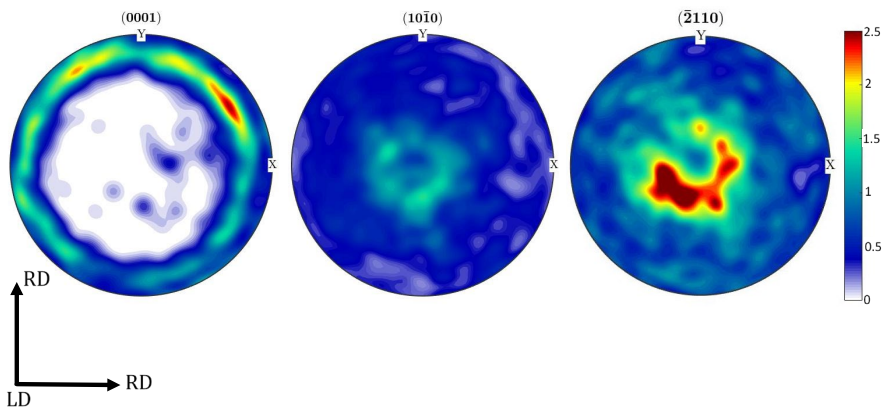
- Twin volume fra[illegible]n of [illegible]syst[illegible]

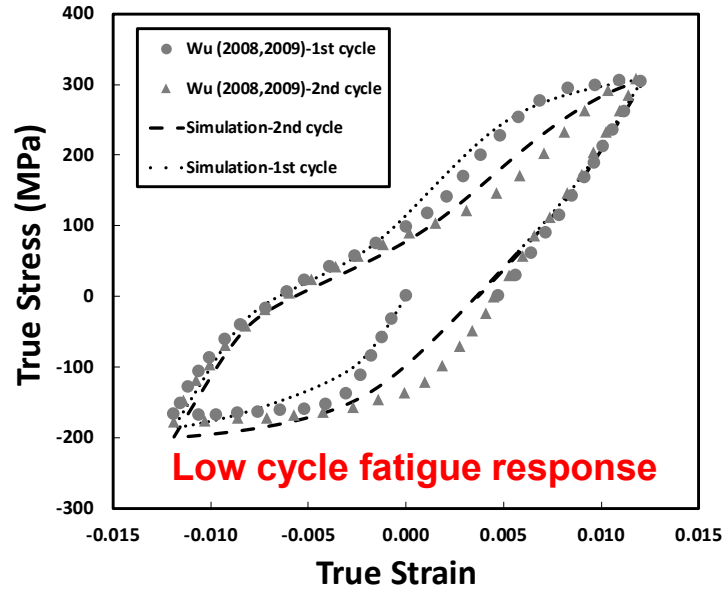
$$\dot{f}_{tw}^{\kappa *} = [1 - f_{tw}]\frac{\left(\dot{\gamma}^{N_{mt}^{s}+\kappa} - \dot{\gamma}^{N_{mt}^{s}+\kappa+6}\right)}{S} - f_{tw}^{\kappa}\frac{\left(\dot{\bar{\gamma}}_{\kappa}^{N_{tw}^{s}+1} - \dot{\bar{\gamma}}_{\kappa}^{N_{tw}^{s}+2}\right)}{S}$$

New CPFEM twinning-detwinning model

Mg alloy ZK60A sample (Wu et al., 2008,2009)

- Basal $\langle a \rangle$ ($\{0001\}\langle 11\bar{2}0\rangle$), Prismatic $\langle a \rangle$ ($\{10\bar{1}0\}\langle 11\bar{2}0\rangle$), and Pyramidal $\langle c+a \rangle$ ($\{\bar{1}\bar{1}22\}\langle \bar{1}\bar{1}23\rangle$),
- Extension twin mode ($\{10\bar{1}2\}\langle \bar{1}011\rangle$)





Calibrated values of crystal plasticity models

Mode	s_0^α (MPa)	h_0^α (MPa)	s_s^α (Mpa)	θ^α
Basal	20	10	21	2
Prismatic	140	20	155	2
Pyramidal<c+a>	240	1000	350	6
Twinning (*operation A*)	48	800	52	0
Twinning (*operations B,C, D*)	15	800	25	0

- $8 \times 9 \times 15$ FE cubic mesh
- Each element represents a single grain

New CPFEM twinning-detwinning model

Run the example of Cyclic deformation!!!

Copy the material model from plasticity/src/materialModels/crystalPlasticity/MaterialModels/RateIndependentAdvancedTwinModel/calculatePlasticity.cc and paste it into the folder plasticity/src/materialModels/crystalPlasticity/.

Recompile the code!!! (it may not get recompiled appropriately just by make release!)

Go to folder plasticity/applications/crystalPlasticity/hcp/advancedTwinModel-Cyclic

Check the prm.prm file

Run the example