Pre- and Post-Processing Files for ρ-CP Simulations

Namit Pai, Suketa Chaudhary, Anirban Patra

Department of Metallurgical Engineering and Materials Science

Indian Institute of Technology Bombay

Geometry and Mesh File Generation

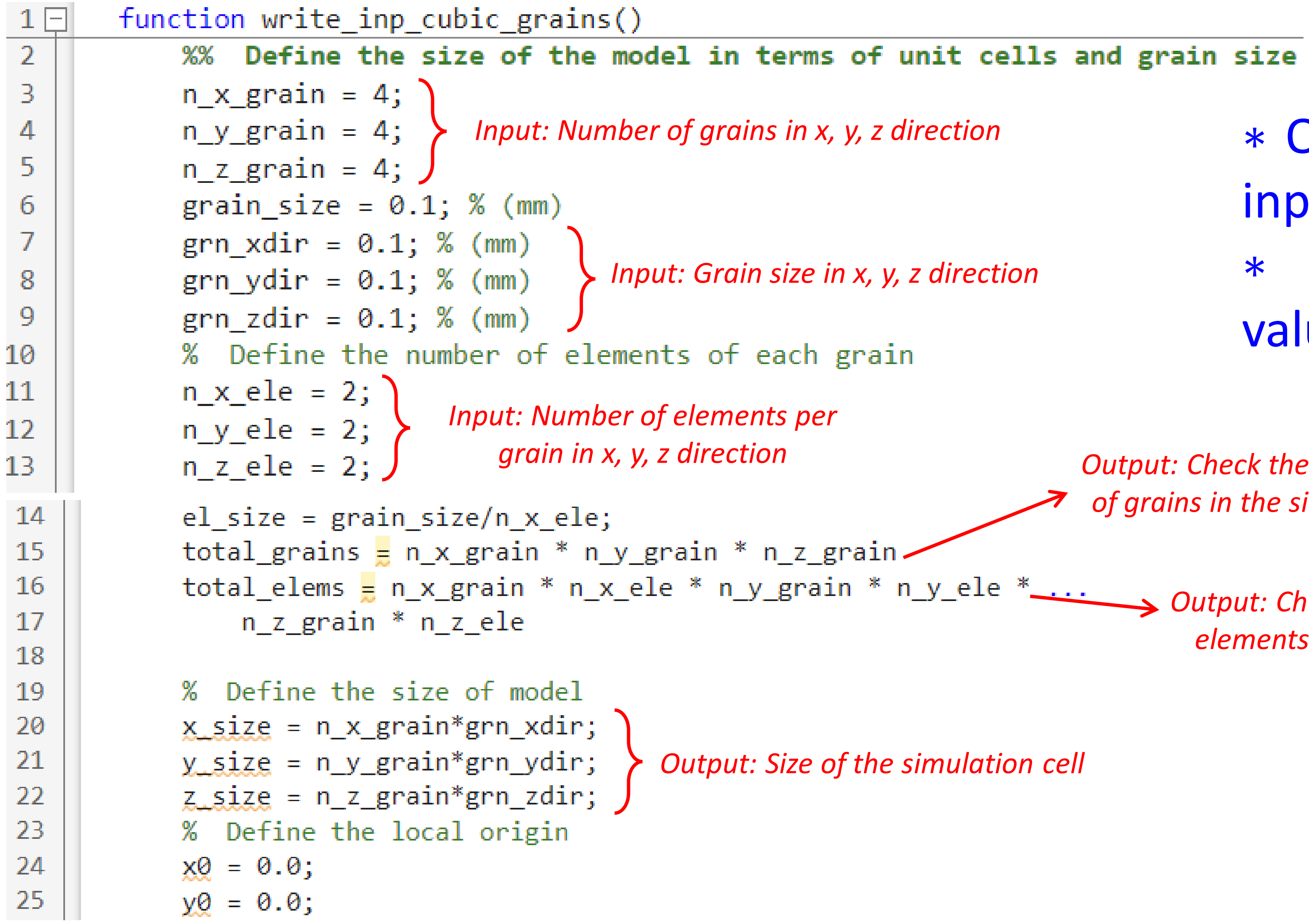
Steps to Generate Mesh File With Cubic Grains - 1

- File: tools/geometry_and_mesh_file_generators/write_inp_cubic_grains.m

- MATLAB script to generate meshes (.inp format): write_inp_cubic_grains.m
- MATLAB version used for this example: R2022b
- Input details: number of grains, grain size, number of elements per grain
- Output filename: ($number of grains)grains_($number of elements)elements.inp

Steps to Generate Mesh File With Cubic Grains - 2

- MATLAB script to generate meshes (.inp format): tools/geometry_and_mesh_file_generators/write_inp_cubic_grains.m



```
function write_inp_cubic_grains()
    %% Define the size of the model in terms of unit cells and grain size
    n_x_grain = 4;
    n_y_grain = 4;
    n_z_grain = 4;
    grain_size = 0.1; % (mm)
    grn_xdir = 0.1; % (mm)
    grn_ydir = 0.1; % (mm)
    grn_zdir = 0.1; % (mm)
    % Define the number of elements of each grain
    n_x_ele = 2;
    n_y_ele = 2;
    n_z_ele = 2;

    el_size = grain_size/n_x_ele;
    total_grains = n_x_grain * n_y_grain * n_z_grain
    total_elems = n_x_grain * n_x_ele * n_y_grain * n_y_ele * ...
        n_z_grain * n_z_ele

    % Define the size of model
    x_size = n_x_grain*grn_xdir;
    y_size = n_y_grain*grn_ydir;
    z_size = n_z_grain*grn_zdir;
    % Define the local origin
    x0 = 0.0;
    y0 = 0.0;
```

Input: Number of grains in x, y, z direction

Input: Grain size in x, y, z direction

Input: Number of elements per grain in x, y, z direction

Output: Check the total number of grains in the simulation cell

Output: Check the total number of elements in the simulation cell

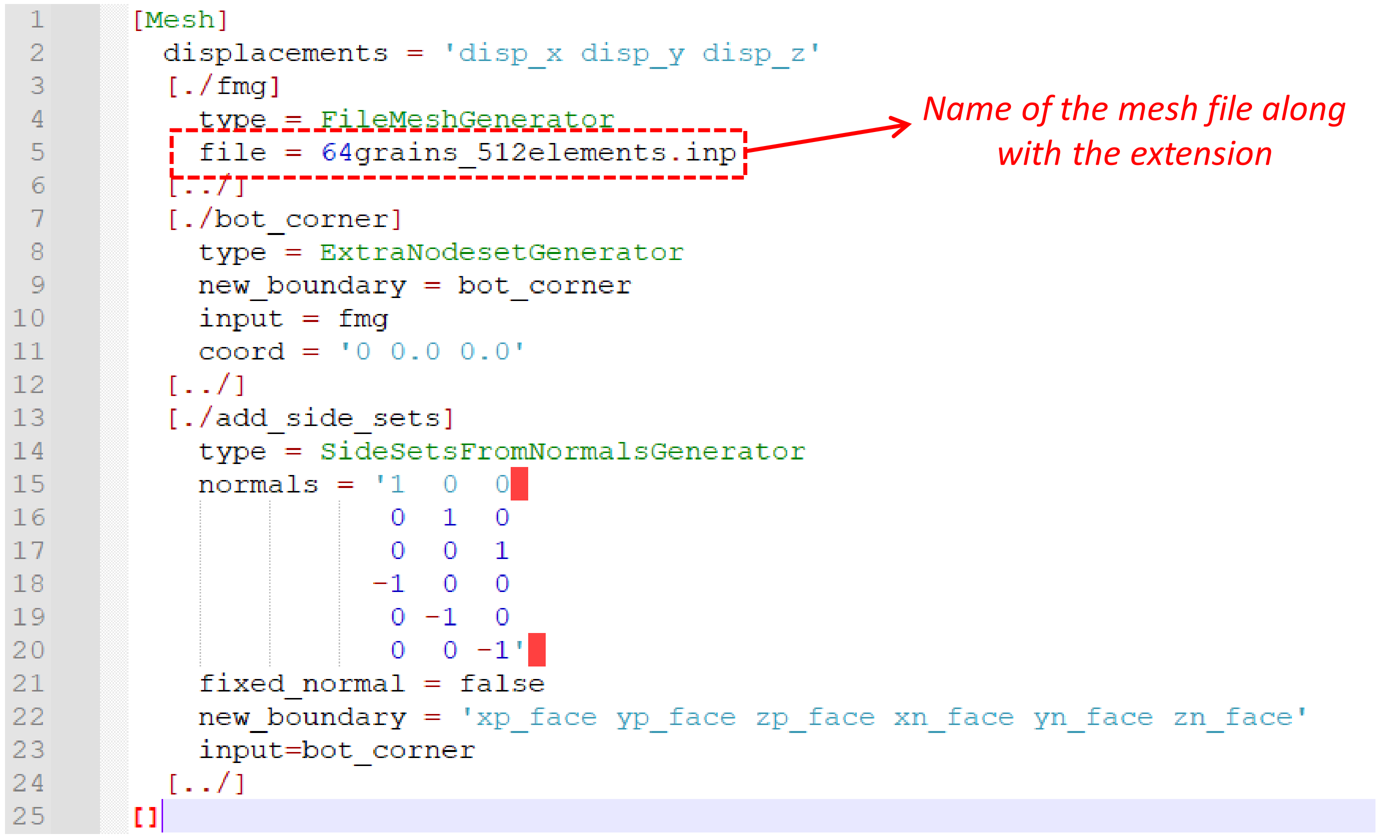
Output: Size of the simulation cell

∗ Only the segments marked as input should be altered

∗ Units dependent upon the values provided in the props file

Steps to Generate Mesh File With Cubic Grains - 3

- Output filename: ($number of grains)grains_($number of elements)elements.inp
- Steps for importing the mesh into input file:



```
[Mesh]
  displacements = 'disp_x disp_y disp_z'
  [./fmg]
    type = FileMeshGenerator
    file = 64grains_512elements.inp
  [../]
  [./bot_corner]
    type = ExtraNodesetGenerator
    new_boundary = bot_corner
    input = fmg
    coord = '0 0.0 0.0'
  [../]
  [./add_side_sets]
    type = SideSetsFromNormalsGenerator
    normals = '1  0  0
               0  1  0
               0  0  1
              -1  0  0
               0 -1  0
               0  0 -1'
    fixed_normal = false
    new_boundary = 'xp_face yp_face zp_face xn_face yn_face zn_face'
    input=bot_corner
  [../]
[]
```

Name of the mesh file along with the extension

Steps to Generate Mesh File with Polyhedral Grains Using Neper-1

- Neper installation: https://neper.info/doc/introduction.html#installing-neper
- Neper commands to generate a 3D polycrystal (Tested on versions 4.5.1-4 and 4.7.0)

Tessellation:

```
neper -T -n from_morpho -domain "cube(1,1,1)" -dim 3 -morpho "diameq:normal(0.5,0.15)" -reg 1 -fmax 20  -mloop 5 -o cube2.tess
```

Input: Cube dimension

3D

Input: Representative of grain size

Input: Regularization removes small edges and faces

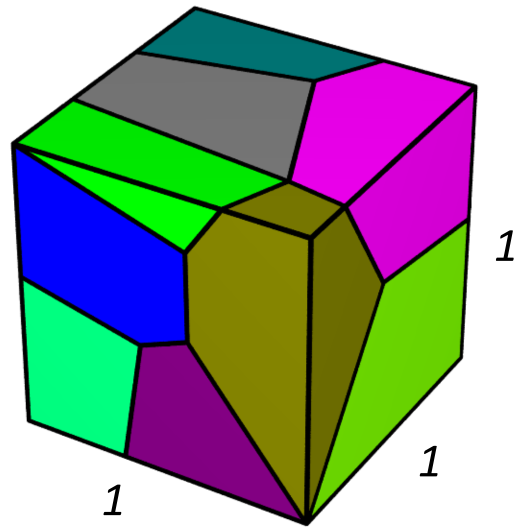
Input: Output file name

Viewing the Tessellation:

```
neper -V cube2.tess -print img
```

Input: Tessellation file generated above

Input: Name of .png file



∗ Only the segments marked as input should be altered

∗ Units dependent upon the values provided in the props file

Steps to Generate Mesh File with Polyhedral Grains Using Neper-2

Tessellation:

```
neper -T -n from_morpho -domain "cube(1,1,1)" -dim 3 -morpho "diameq:normal(0.5,0.15)" -reg 1 -fmax 20  -mloop 5 -o cube2.tess
```

Input: Cube dimension

3D

Input: Representative of grain size

Input: Regularization removes small edges and faces

Input: Output file name

Meshing:

```
neper -M cube2.tess -cl 0.2 -clmin 0.15 -meshqualmin 1 -o cube2_meshed
```

Input: Mesh size parameters

Mesh quality

Input: Name of output mesh file

* Only the segments marked as input should be altered

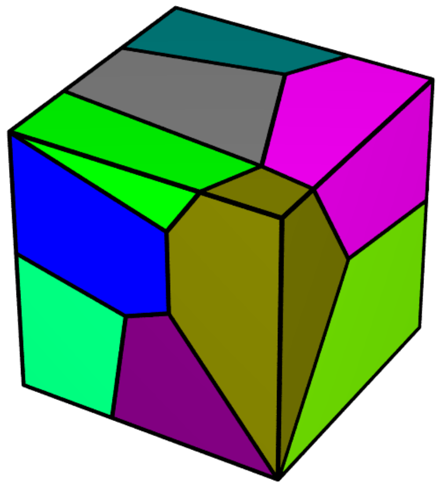
Viewing the mesh:

```
neper -V cube2_meshed -print img
```

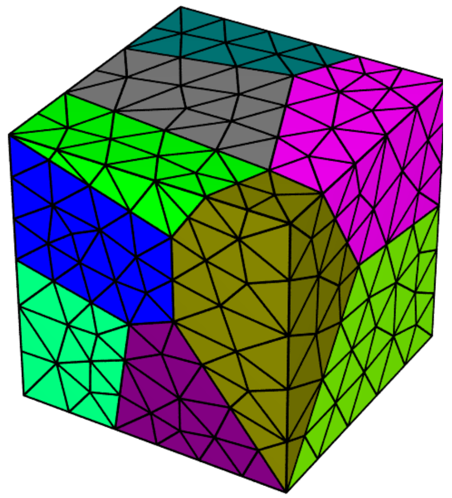
Input: Tessellation file generated above

Input: Name of .png file

Tessellated



Meshed



Steps to Generate Mesh File with Polyhedral Grains Using Neper-3

- Convert to inp (ABAQUS input) format

Meshing:

neper -M cube2.tess -cl 0.2 -clmin 0.15 -meshqualmin 1 -o cube2_meshed -format inp

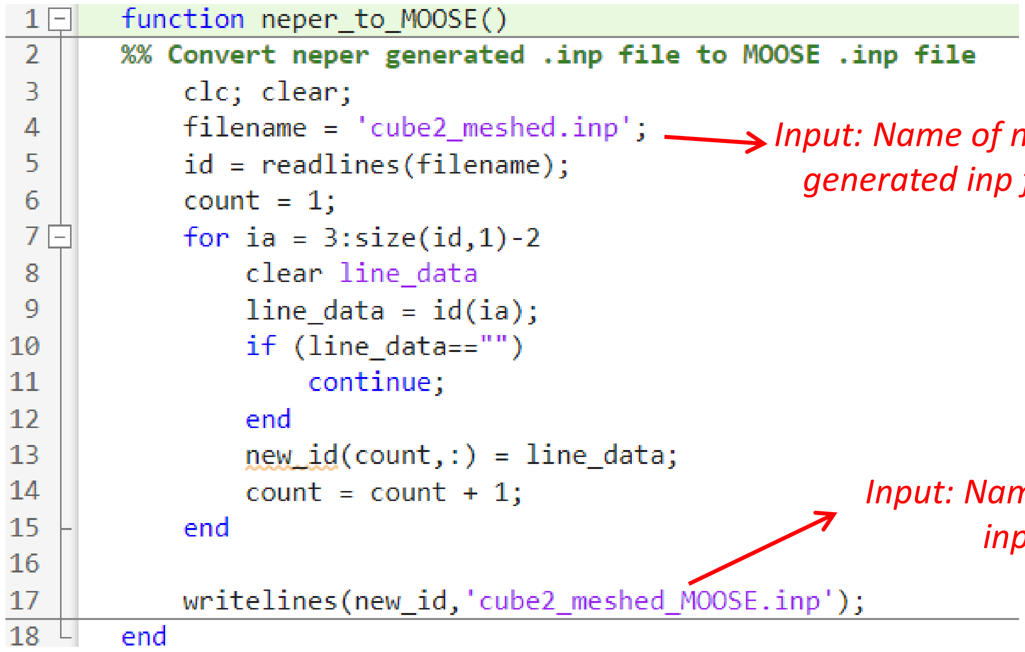
Input: Mesh size parameters

Mesh quality

Input: Name of output mesh file

Format of output file
Default: .msh

- File: tools/geometry_and_mesh_file_generators/neper_to_MOOSE.m
- Use a MATLAB script to convert the .inp file to MOOSE readable format (MATLAB code used: *neper_to_MOOSE.m*)



```
function neper_to_MOOSE()
%% Convert neper generated .inp file to MOOSE .inp file
    clc; clear;
    filename = 'cube2_meshed.inp';
    id = readlines(filename);
    count = 1;
    for ia = 3:size(id,1)-2
        clear line_data
        line_data = id(ia);
        if (line_data=="")
            continue;
        end
        new_id(count,:) = line_data;
        count = count + 1;
    end

    writelines(new_id,'cube2_meshed_MOOSE.inp');
end
```

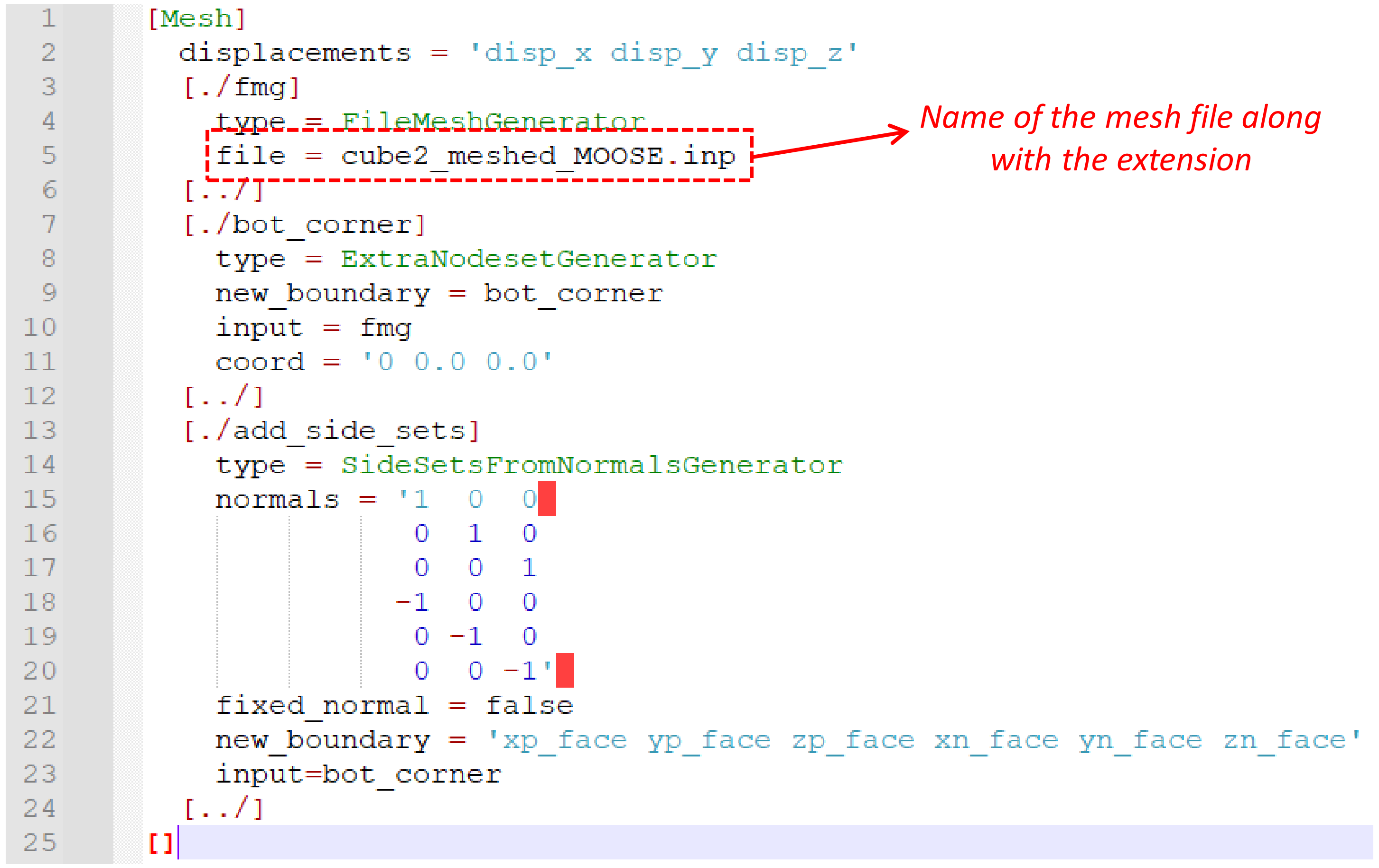
Input: Name of neper generated inp file

Input: Name of output inp file

∗ Only the segments marked as input should be altered

Steps to Generate Mesh File with Polyhedral Grains Using Neper-4

- Importing into MOOSE



```
[Mesh]
  displacements = 'disp_x disp_y disp_z'
  [./fmg]
    type = FileMeshGenerator
    file = cube2_meshed_MOOSE.inp
  [../]
  [./bot_corner]
    type = ExtraNodesetGenerator
    new_boundary = bot_corner
    input = fmg
    coord = '0 0.0 0.0'
  [../]
  [./add_side_sets]
    type = SideSetsFromNormalsGenerator
    normals = '1  0  0
               0  1  0
               0  0  1
              -1  0  0
               0 -1  0
               0  0 -1'
    fixed_normal = false
    new_boundary = 'xp_face yp_face zp_face xn_face yn_face zn_face'
    input=bot_corner
  [../]
[]
```

Name of the mesh file along with the extension

Steps to generate mesh file using neper-5

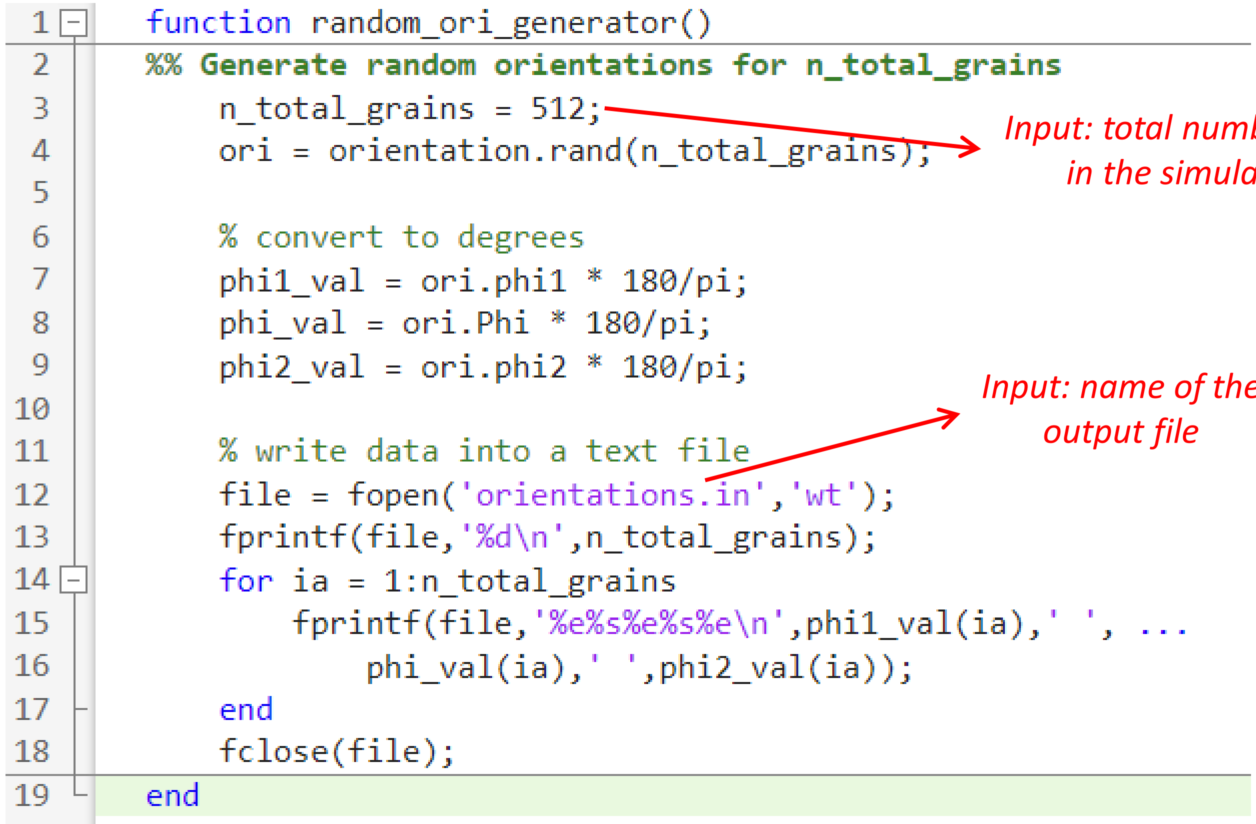
- Summary

1. *(Optional) export OMP_NUM_THREADS=$number of processors$*
2. *neper -T -n from_morpho -domain "cube(1,1,1)" -dim 3 -morpho "diameq:normal(0.5,0.15)" -reg 1 -fmax 20 -mloop 5 -o cube2.tess*
3. *neper -V cube2.tess -print img*
4. *neper -M cube2.tess -cl 0.2 -clmin 0.15 -meshqualmin 1 -o cube2_meshed -format inp*
5. *Run the MATLAB script (neper_to_MOOSE.m) to convert neper .inp file to MOOSE readable .inp file*
6. *Import into MOOSE.*

Random Orientation Generator

Steps to Generate Random Input Texture Using MTEX-1

- File: tools/random_orientation_generator/random_ori_generator.m
- MATLAB script to generate random texture : random_ori_generator.m
- MATLAB version used for this example: R2022b
- MTex version: 5.4.0/5.11.1 (https://mtex-toolbox.github.io/download)
- Input details: number of grains
- Output filename: orientations.in



```
function random_ori_generator()
%% Generate random orientations for n_total_grains
    n_total_grains = 512;
    ori = orientation.rand(n_total_grains);

    % convert to degrees
    phi1_val = ori.phi1 * 180/pi;
    phi_val = ori.Phi * 180/pi;
    phi2_val = ori.phi2 * 180/pi;

    % write data into a text file
    file = fopen('orientations.in','wt');
    fprintf(file,'%d\n',n_total_grains);
    for ia = 1:n_total_grains
        fprintf(file,'%e%s%e%s%e\n',phi1_val(ia),' ', ...
            phi_val(ia),' ',phi2_val(ia));
    end
    fclose(file);
end
```

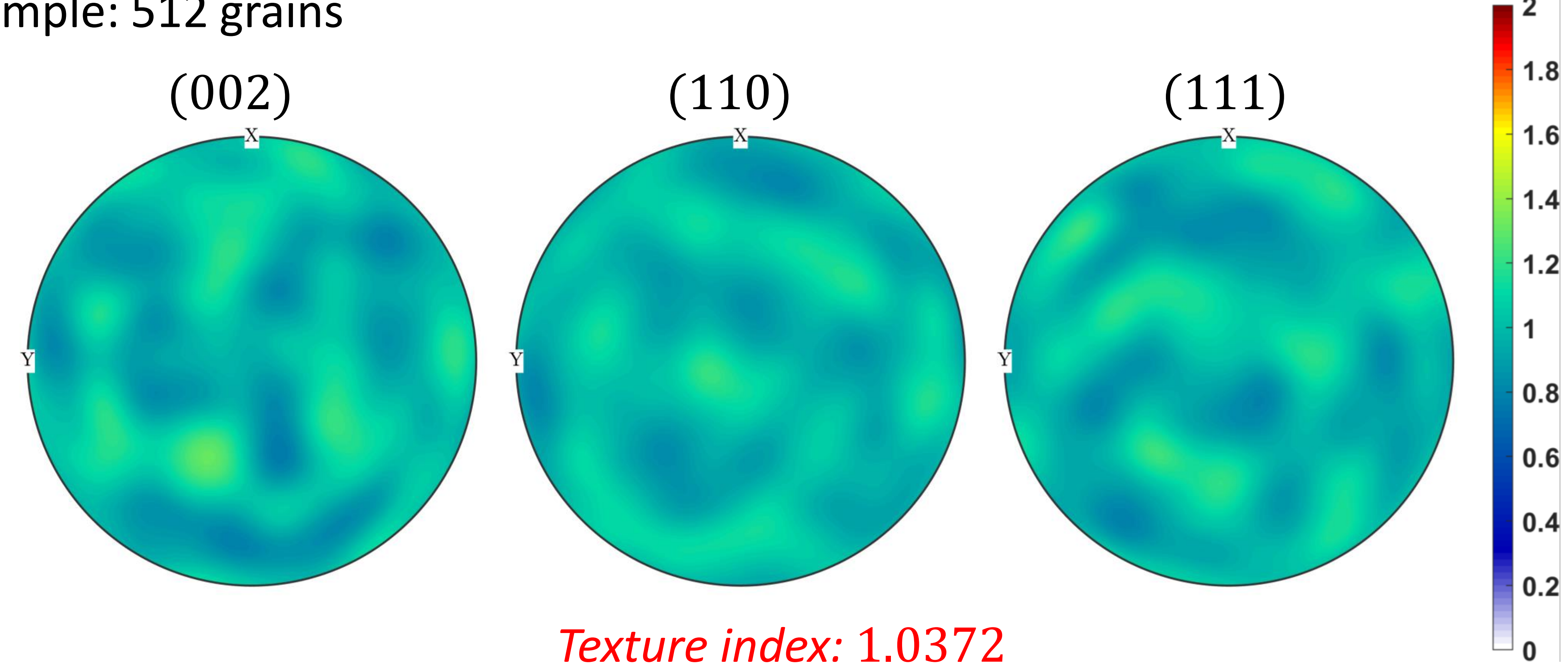
Input: total number of grains in the simulation cell

Input: name of the output file

* Only the segments marked as input should be altered

Steps to Generate Random Input Texture Using MTEX-2

- MATLAB script to generate random texture : random_ori_generator.m
- MATLAB version used for this example: R2022b
- MTex version: 5.4.0/5.11.1 (*https://mtex-toolbox.github.io/download*)
- Example: 512 grains



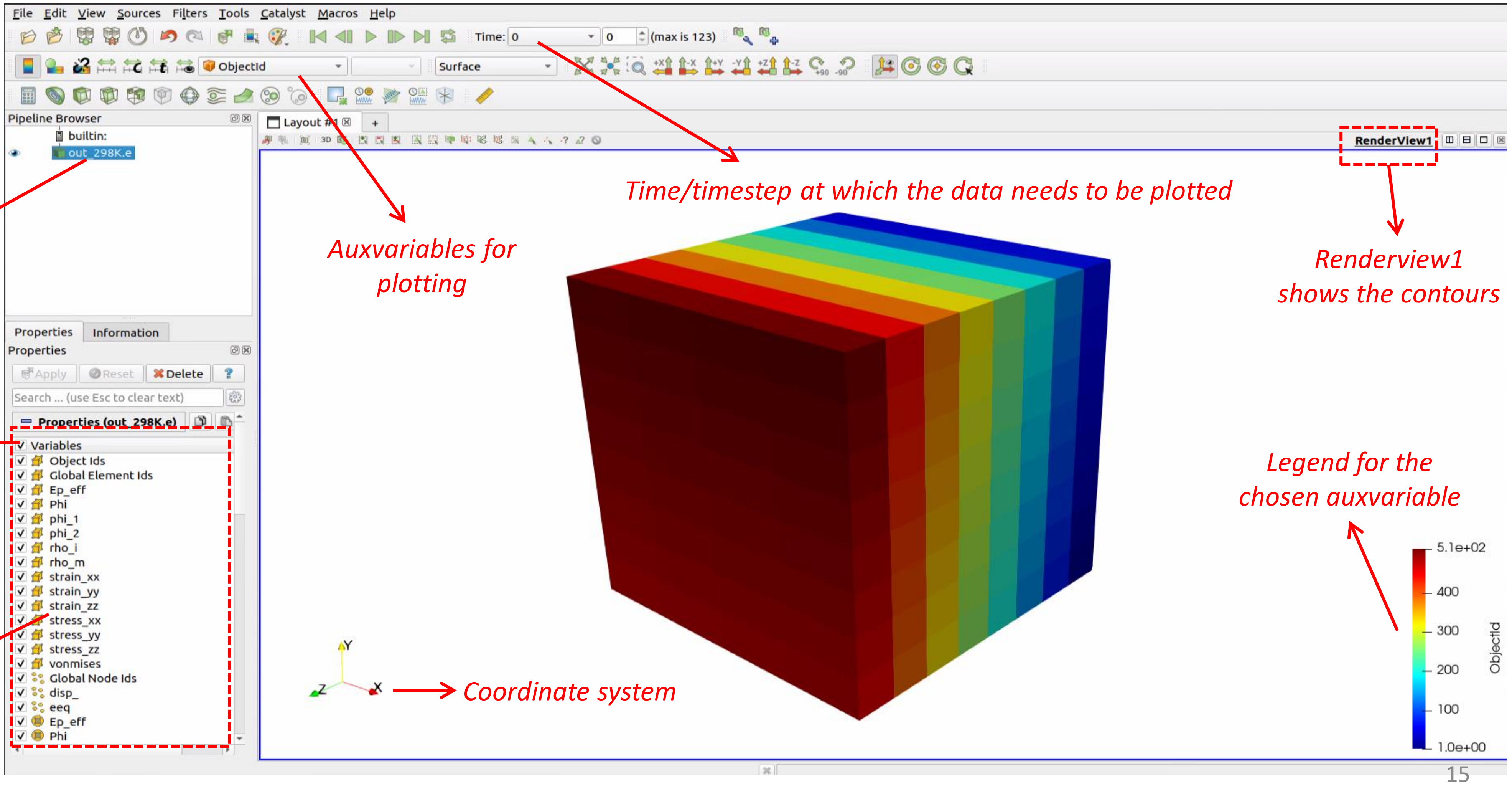
Texture index: 1.0372

- Please refer to https://mtex-toolbox.github.io/OrientationDefinition.html for further understanding on this topic

Plotting Textures using MTEX

Steps to Plot Texture Using Paraview and MTEX-1

- Paraview version: 5.7.0 (https://www.paraview.org/download/)
- File used for this exercise: out_298K.e (Figure 14 from Patra et al., Comp. Mat. Sci. (2023))



Variables
- ✓ Object Ids
- ✓ Global Element Ids
- ✓ Ep_eff
- ✓ Phi
- ✓ phi_1
- ✓ phi_2
- ✓ rho_i
- ✓ rho_m
- ✓ strain_xx
- ✓ strain_yy
- ✓ strain_zz
- ✓ stress_xx
- ✓ stress_yy
- ✓ stress_zz
- ✓ vonmises
- ✓ Global Node Ids
- ✓ disp_
- ✓ eeq
- ✓ Ep_eff
- ✓ Phi

filename

Auxvariables for plotting

Time/timestep at which the data needs to be plotted

Renderview1 shows the contours

Select the desired variables by checking the respective boxes under 'Properties' bar

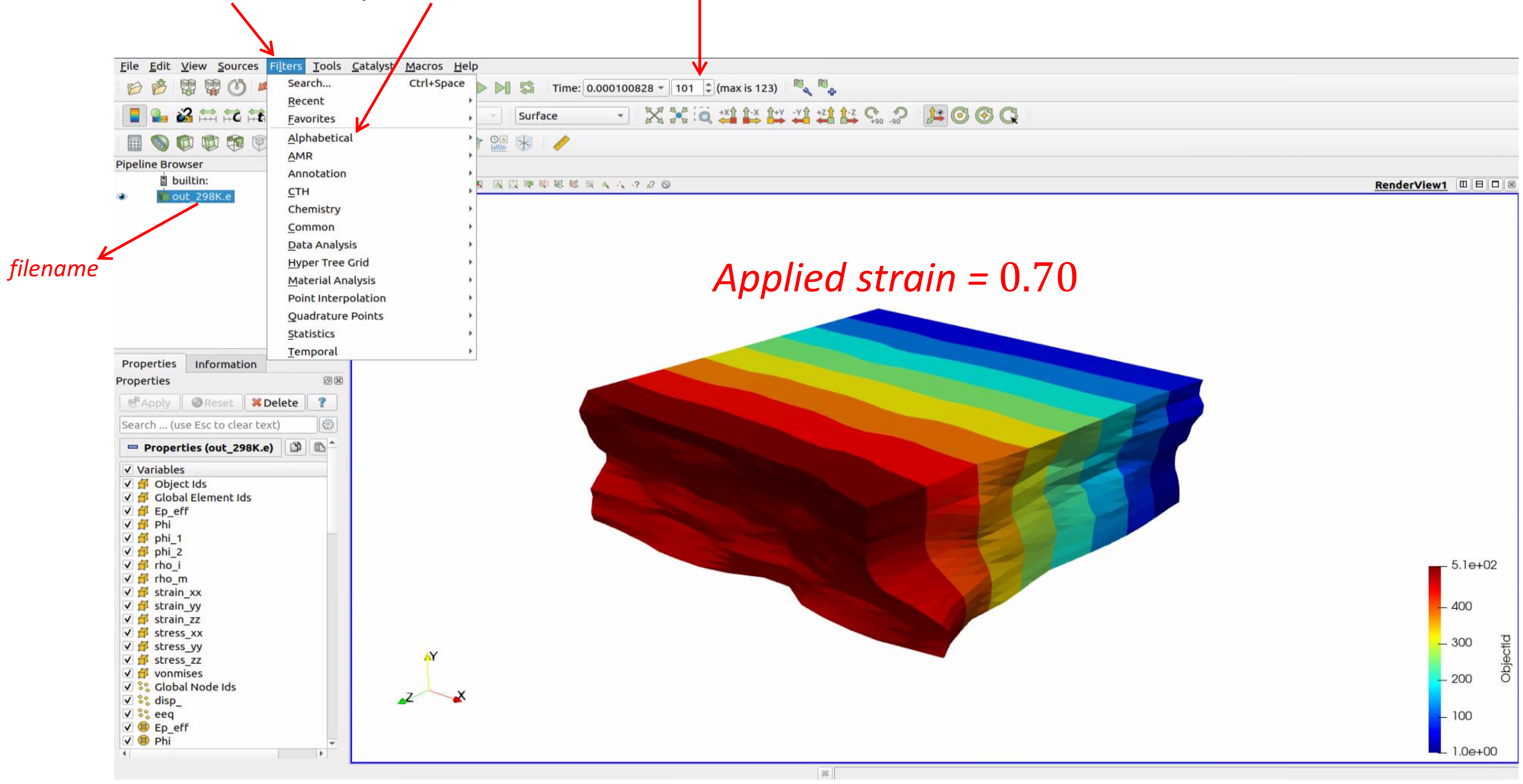
Legend for the chosen auxvariable

Auxvariables from input file appear over here

Coordinate system

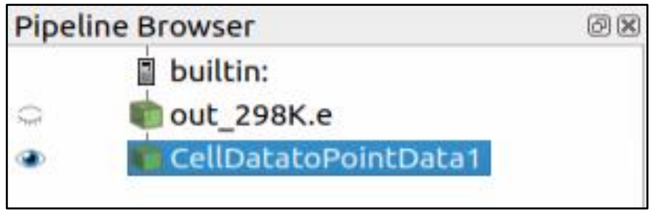
Steps to Plot Texture Using Paraview and MTEX-2

1. Select the desired time/timestep
2. Go to Filters→Alphabetical→Cell Data to Point Data

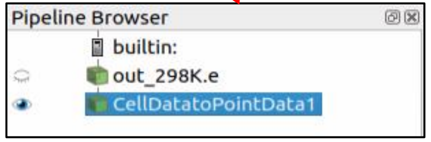


Steps to Plot Texture Using Paraview and MTEX-3

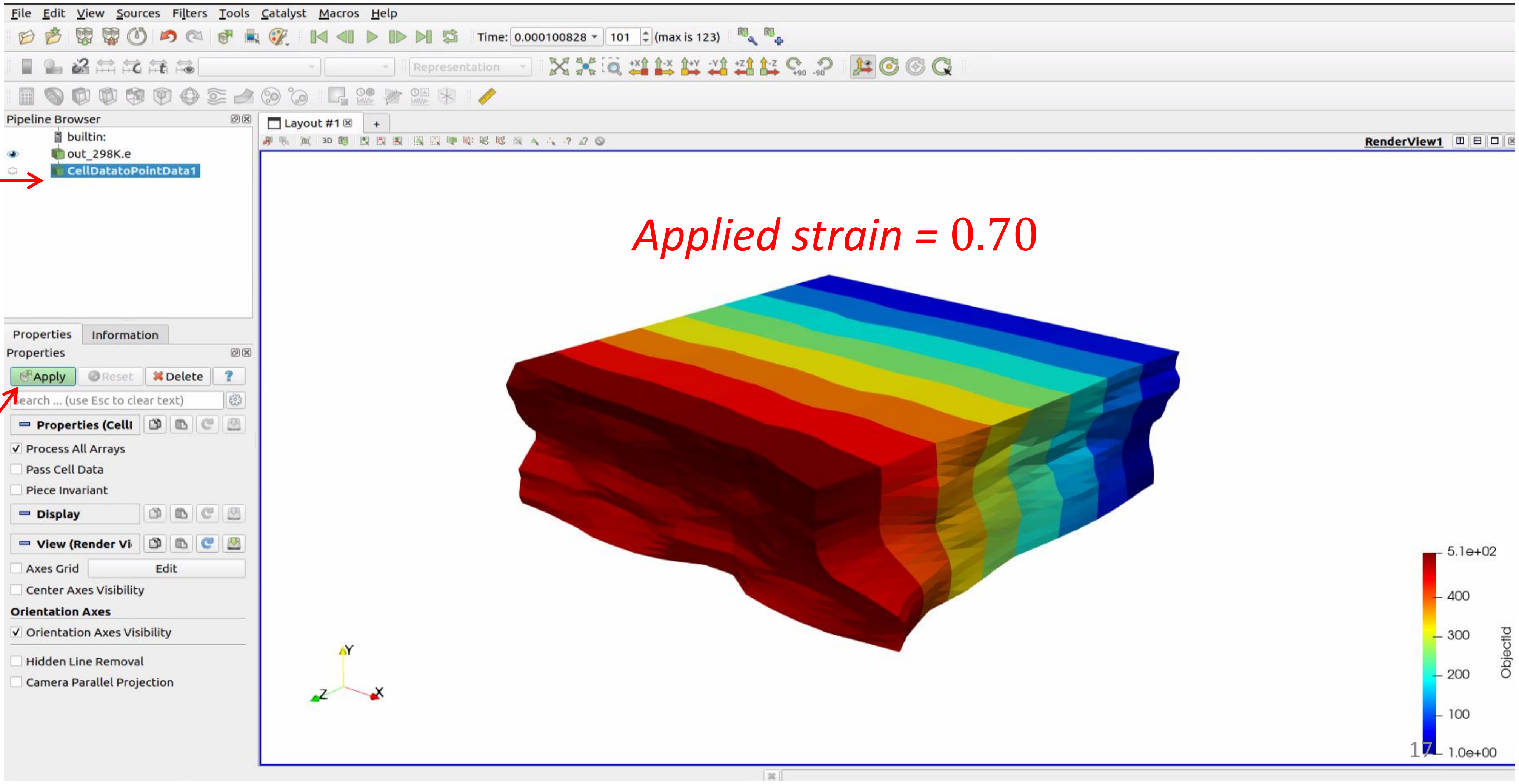
3. Select Apply
4. The highlighted block should be placed on the CellDatatoPointData as shown here:



The selected filter should appear over here. The highlight shifts to this block once you click on Apply



Click on Apply

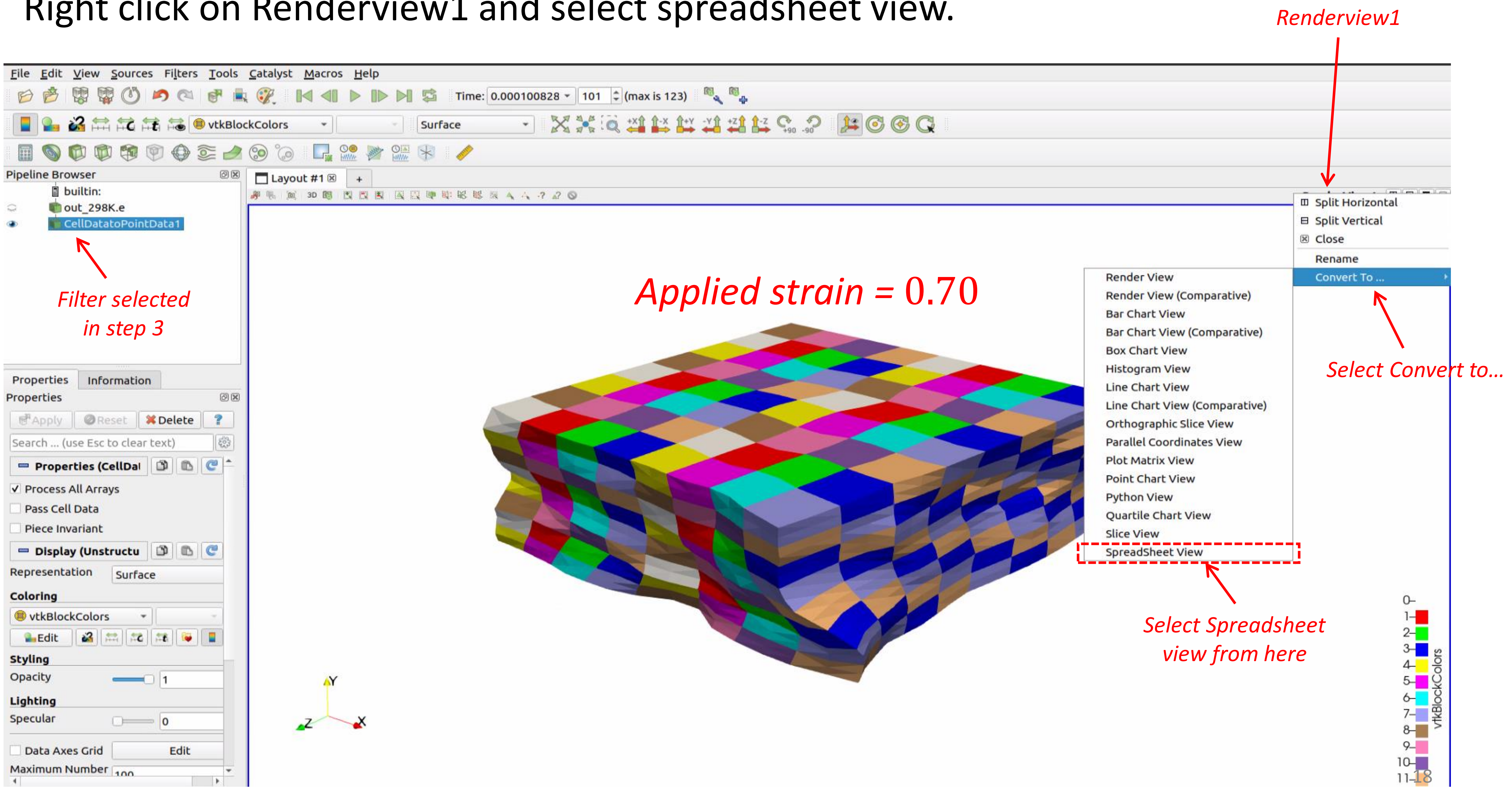


☑ Process All Arrays
☐ Pass Cell Data
☐ Piece Invariant
☐ Axes Grid
☐ Center Axes Visibility
☑ Orientation Axes Visibility
☐ Hidden Line Removal
☐ Camera Parallel Projection

Applied strain = 0.70

Steps to Plot Texture Using Paraview and MTEX-4

5. Right click on Renderview1 and select spreadsheet view.



Time: 0.000100828 | 101 | (max is 123)

vtkBlockColors | Surface

- [✓] Process All Arrays
- [ ] Pass Cell Data
- [ ] Piece Invariant

Representation: Surface

Coloring: vtkBlockColors

Opacity: 1

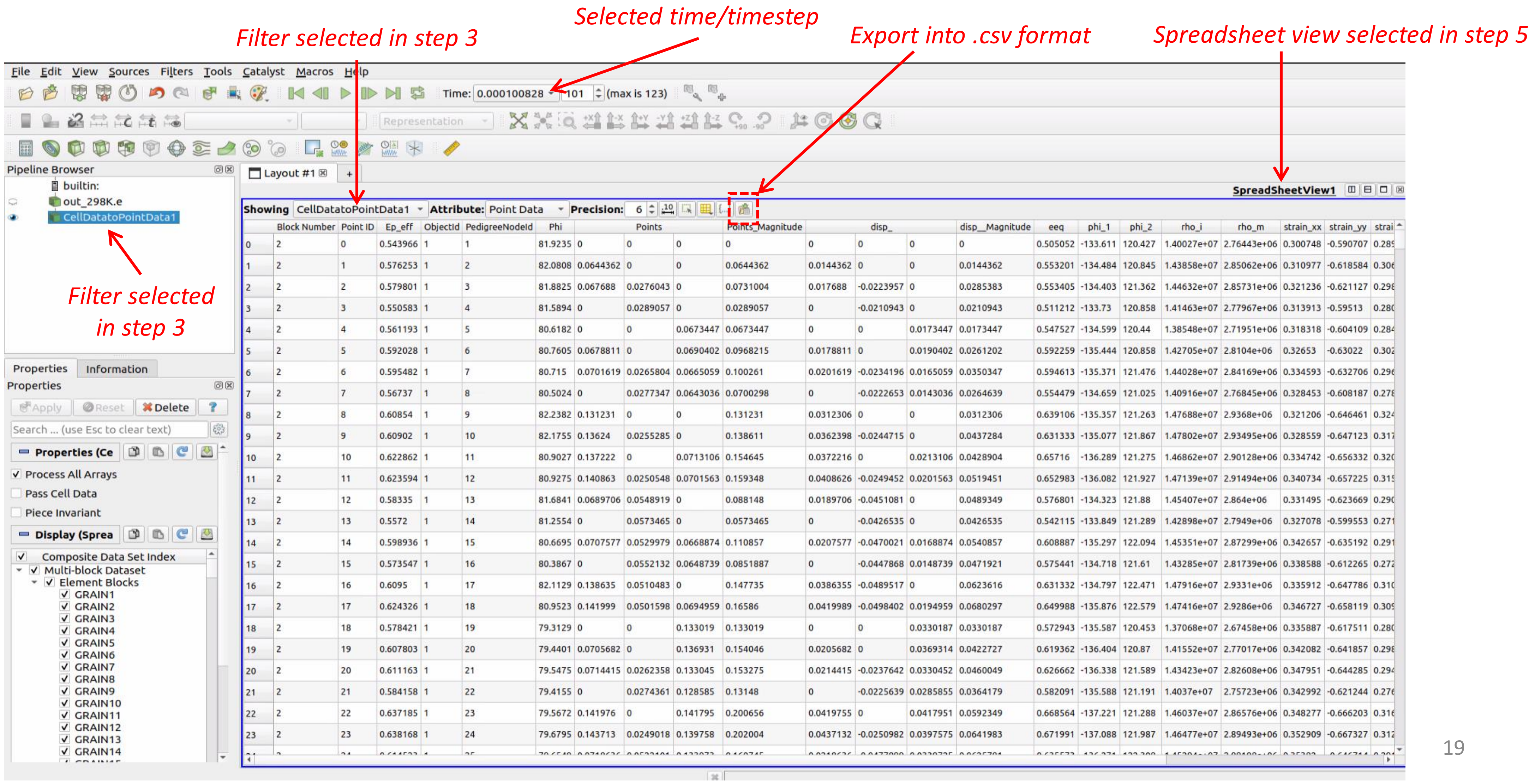
Specular: 0

- [ ] Data Axes Grid

Maximum Number: 100

Steps to Plot Texture Using Paraview and MTEX-5

6. Export the spreadsheet view as .csv file. (Cross-check if all desired variables are present, if not, revisit slide 14 ‘Properties’ bar)


Filter selected in step 3
Selected time/timestep
Export into .csv format
Spreadsheet view selected in step 5
Filter selected in step 3
File Edit View Sources Filters Tools Catalyst Macros Help
Representation
Pipeline Browser
Layout #1
SpreadSheetView1
Properties
Information
Apply
Reset
Delete

Time: 0.000100828 | 101 | (max is 123)

Pipeline Browser: builtin: | out_298K.e | CellDatatoPointData1

Properties (CellDatatoPointData1)

- [✓] Process All Arrays
- [ ] Pass Cell Data
- [ ] Piece Invariant

Display (SpreadSheetRepresentation)

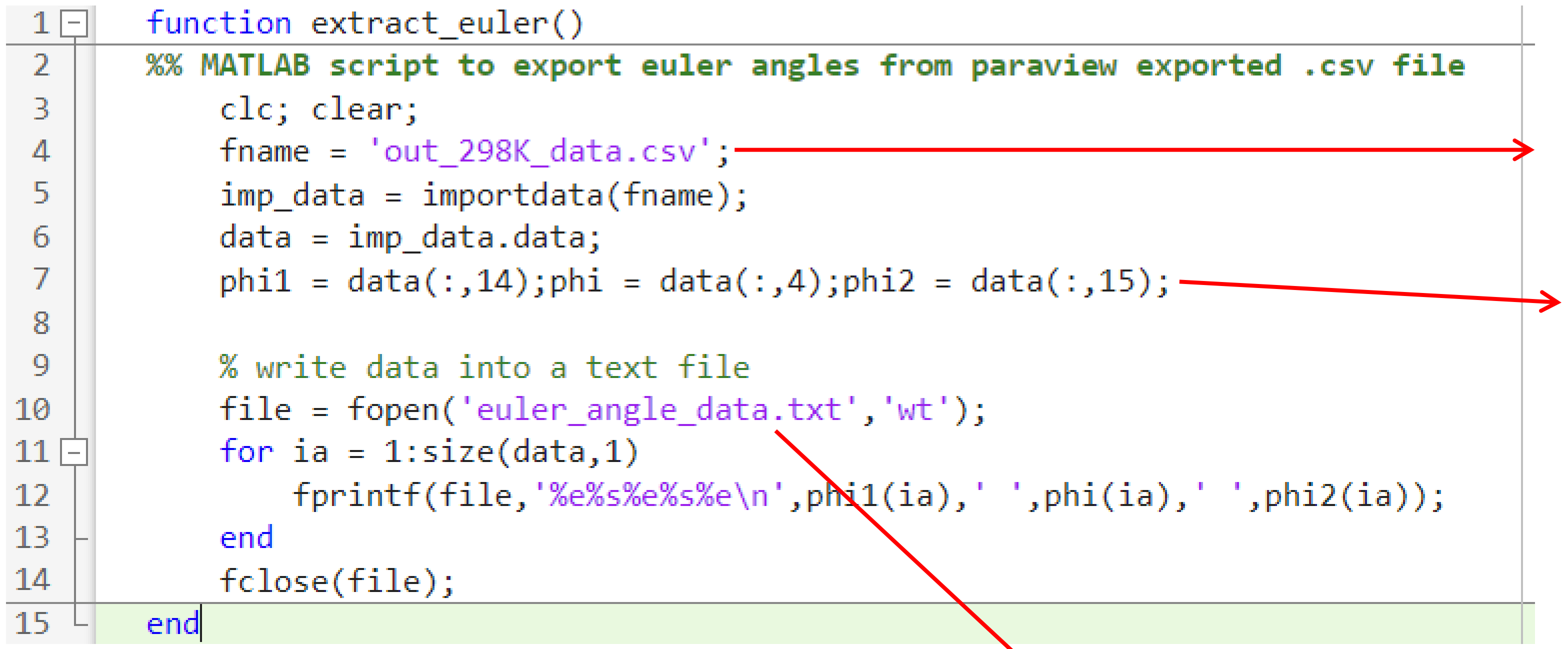
- [✓] Composite Data Set Index
  - [✓] Multi-block Dataset
    - [✓] Element Blocks
      - [✓] GRAIN1
      - [✓] GRAIN2
      - [✓] GRAIN3
      - [✓] GRAIN4
      - [✓] GRAIN5
      - [✓] GRAIN6
      - [✓] GRAIN7
      - [✓] GRAIN8
      - [✓] GRAIN9
      - [✓] GRAIN10
      - [✓] GRAIN11
      - [✓] GRAIN12
      - [✓] GRAIN13
      - [✓] GRAIN14

Showing: CellDatatoPointData1 | Attribute: Point Data | Precision: 6

	Block Number	Point ID	Ep_eff	ObjectId	PedigreeNodeId	Phi	Points			Points_Magnitude	disp_			disp__Magnitude	eeq	phi_1	phi_2	rho_i	rho_m	strain_xx	strain_yy	strai
0	2	0	0.543966	1	1	81.9235	0	0	0	0	0	0	0	0	0.505052	-133.611	120.427	1.40027e+07	2.76443e+06	0.300748	-0.590707	0.28
1	2	1	0.576253	1	2	82.0808	0.0644362	0	0	0.0644362	0.0144362	0	0	0.0144362	0.553201	-134.484	120.845	1.43858e+07	2.85062e+06	0.310977	-0.618584	0.306
2	2	2	0.579801	1	3	81.8825	0.067688	0.0276043	0	0.0731004	0.017688	-0.0223957	0	0.0285383	0.553405	-134.403	121.362	1.44632e+07	2.85731e+06	0.321236	-0.621127	0.298
3	2	3	0.550583	1	4	81.5894	0	0.0289057	0	0.0289057	0	-0.0210943	0	0.0210943	0.511212	-133.73	120.858	1.41463e+07	2.77967e+06	0.313913	-0.59513	0.280
4	2	4	0.561193	1	5	80.6182	0	0	0.0673447	0.0673447	0	0	0.0173447	0.0173447	0.547527	-134.599	120.44	1.38548e+07	2.71951e+06	0.318318	-0.604109	0.284
5	2	5	0.592028	1	6	80.7605	0.0678811	0	0.0690402	0.0968215	0.0178811	0	0.0190402	0.0261202	0.592259	-135.444	120.858	1.42705e+07	2.8104e+06	0.32653	-0.63022	0.302
6	2	6	0.595482	1	7	80.715	0.0701619	0.0265804	0.0665059	0.100261	0.0201619	-0.0234196	0.0165059	0.0350347	0.594613	-135.371	121.476	1.44028e+07	2.84169e+06	0.334593	-0.632706	0.296
7	2	7	0.56737	1	8	80.5024	0	0.0277347	0.0643036	0.0700298	0	-0.0222653	0.0143036	0.0264639	0.554479	-134.659	121.025	1.40916e+07	2.76845e+06	0.328453	-0.608187	0.278
8	2	8	0.60854	1	9	82.2382	0.131231	0	0	0.131231	0.0312306	0	0	0.0312306	0.639106	-135.357	121.263	1.47688e+07	2.9368e+06	0.321206	-0.646461	0.324
9	2	9	0.60902	1	10	82.1755	0.13624	0.0255285	0	0.138611	0.0362398	-0.0244715	0	0.0437284	0.631333	-135.077	121.867	1.47802e+07	2.93495e+06	0.328559	-0.647123	0.317
10	2	10	0.622862	1	11	80.9027	0.137222	0	0.0713106	0.154645	0.0372216	0	0.0213106	0.0428904	0.65716	-136.289	121.275	1.46862e+07	2.90128e+06	0.334742	-0.656332	0.320
11	2	11	0.623594	1	12	80.9275	0.140863	0.0250548	0.0701563	0.159348	0.0408626	-0.0249452	0.0201563	0.0519451	0.652983	-136.082	121.927	1.47139e+07	2.91494e+06	0.340734	-0.657225	0.315
12	2	12	0.58335	1	13	81.6841	0.0689706	0.0548919	0	0.088148	0.0189706	-0.0451081	0	0.0489349	0.576801	-134.323	121.88	1.45407e+07	2.864e+06	0.331495	-0.623669	0.290
13	2	13	0.5572	1	14	81.2554	0	0.0573465	0	0.0573465	0	-0.0426535	0	0.0426535	0.542115	-133.849	121.289	1.42898e+07	2.7949e+06	0.327078	-0.599553	0.271
14	2	14	0.598936	1	15	80.6695	0.0707577	0.0529979	0.0668874	0.110857	0.0207577	-0.0470021	0.0168874	0.0540857	0.608887	-135.297	122.094	1.45351e+07	2.87299e+06	0.342657	-0.635192	0.291
15	2	15	0.573547	1	16	80.3867	0	0.0552132	0.0648739	0.0851887	0	-0.0447868	0.0148739	0.0471921	0.575441	-134.718	121.61	1.43285e+07	2.81739e+06	0.338588	-0.612265	0.272
16	2	16	0.6095	1	17	82.1129	0.138635	0.0510483	0	0.147735	0.0386355	-0.0489517	0	0.0623616	0.631332	-134.797	122.471	1.47916e+07	2.9331e+06	0.335912	-0.647786	0.310
17	2	17	0.624326	1	18	80.9523	0.141999	0.0501598	0.0694959	0.16586	0.0419989	-0.0498402	0.0194959	0.0680297	0.649988	-135.876	122.579	1.47416e+07	2.9286e+06	0.346727	-0.658119	0.309
18	2	18	0.578421	1	19	79.3129	0	0	0.133019	0.133019	0	0	0.0330187	0.0330187	0.572943	-135.587	120.453	1.37068e+07	2.67458e+06	0.335887	-0.617511	0.280
19	2	19	0.607803	1	20	79.4401	0.0705682	0	0.136931	0.154046	0.0205682	0	0.0369314	0.0422727	0.619362	-136.404	120.87	1.41552e+07	2.77017e+06	0.342082	-0.641857	0.298
20	2	20	0.611163	1	21	79.5475	0.0714415	0.0262358	0.133045	0.153275	0.0214415	-0.0237642	0.0330452	0.0460049	0.626662	-136.338	121.589	1.43423e+07	2.82608e+06	0.347951	-0.644285	0.294
21	2	21	0.584158	1	22	79.4155	0	0.0274361	0.128585	0.13148	0	-0.0225639	0.0285855	0.0364179	0.582091	-135.588	121.191	1.4037e+07	2.75723e+06	0.342992	-0.621244	0.276
22	2	22	0.637185	1	23	79.5672	0.141976	0	0.141795	0.200656	0.0419755	0	0.0417951	0.0592349	0.668564	-137.221	121.288	1.46037e+07	2.86576e+06	0.348277	-0.666203	0.316
23	2	23	0.638168	1	24	79.6795	0.143713	0.0249018	0.139758	0.202004	0.0437132	-0.0250982	0.0397575	0.0641983	0.671991	-137.088	121.987	1.46477e+07	2.89493e+06	0.352909	-0.667327	0.312

Steps to Plot Texture Using Paraview and MTEX-6

7. Extract the Euler angles from the csv file exported in step 6 (MATLAB code used: tools/texture_plotting/extract_euler.m)



```matlab
function extract_euler()
%% MATLAB script to export euler angles from paraview exported .csv file
    clc; clear;
    fname = 'out_298K_data.csv';
    imp_data = importdata(fname);
    data = imp_data.data;
    phi1 = data(:,14);phi = data(:,4);phi2 = data(:,15);

    % write data into a text file
    file = fopen('euler_angle_data.txt','wt');
    for ia = 1:size(data,1)
        fprintf(file,'%e%s%e%s%e\n',phi1(ia),' ',phi(ia),' ',phi2(ia));
    end
    fclose(file);
end
```

Filename for the csv file exported from Paraview

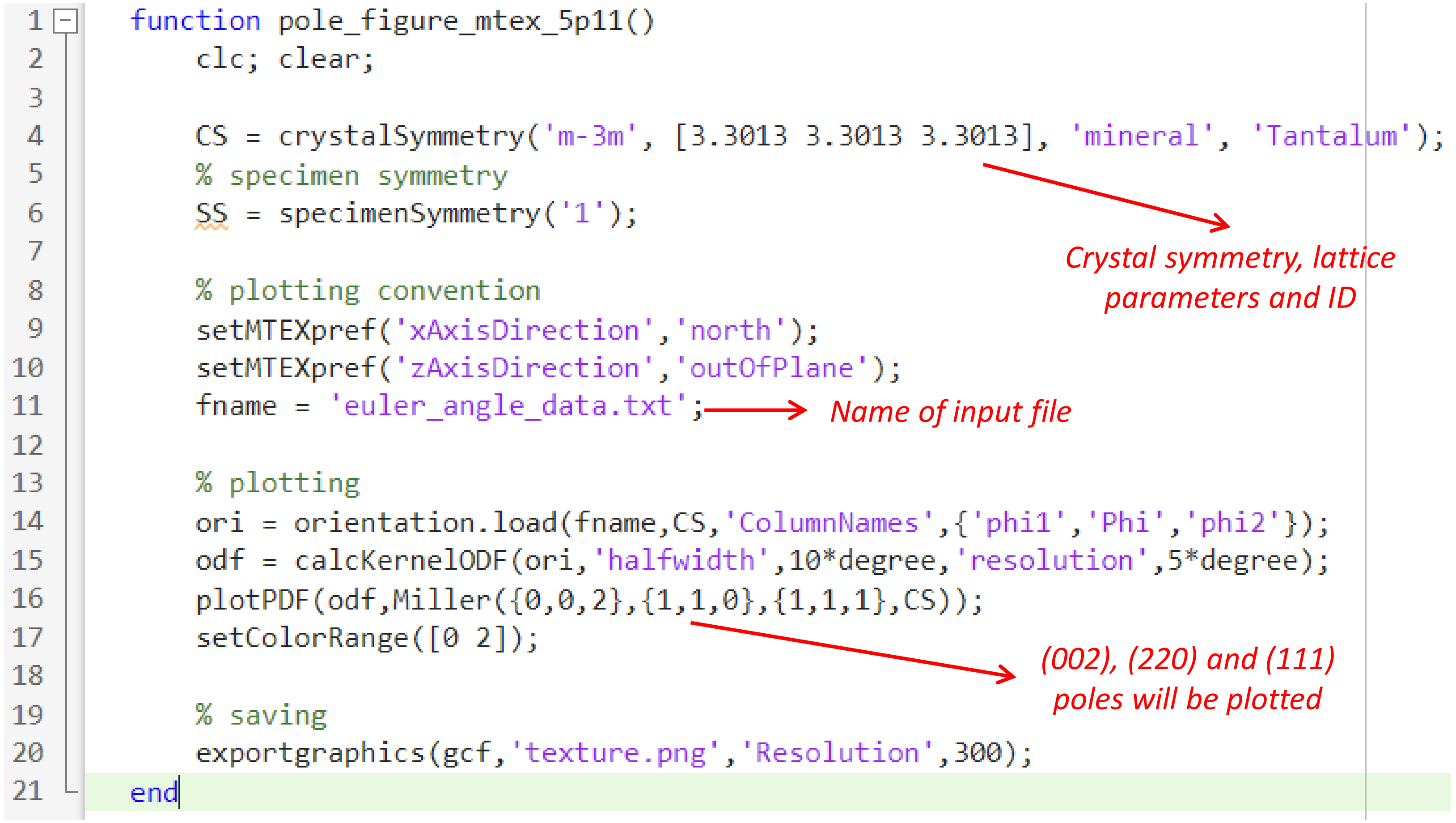
Columns containing the ϕ_1, ϕ and ϕ_2 data in the .csv file.

***This can change based on the number of Auxvariables declared in the input file.**

Name of the file storing the Euler angles in txt format

Steps to Plot Texture Using Paraview and MTEX-7

- MTex version: 5.11.1 (https://mtex-toolbox.github.io/download)
- Exported File from step 7: euler_angle_data.txt (Figure 14 from Patra et al. 2023)
- Code to be used for plotting: tools/texture_plotting_mtex_5p11/pole_figure_mtex_5p11.m

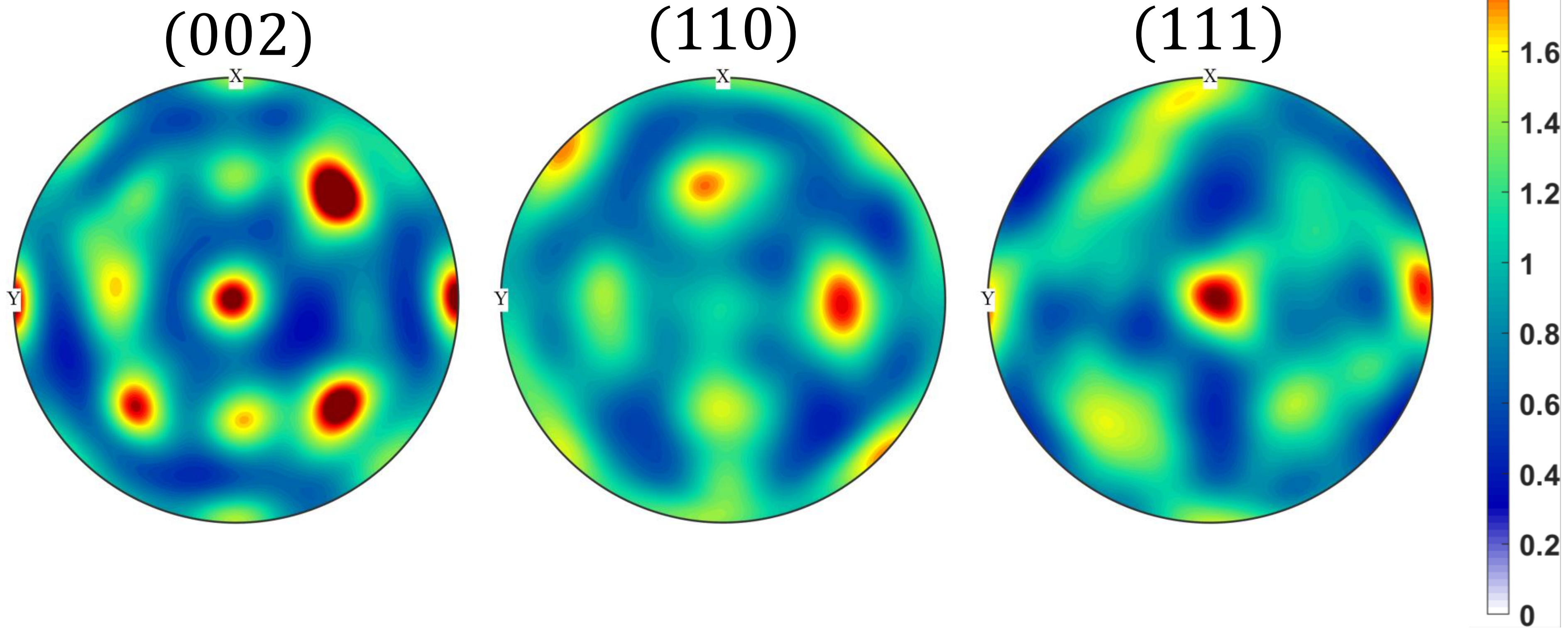


```
function pole_figure_mtex_5p11()
    clc; clear;

    CS = crystalSymmetry('m-3m', [3.3013 3.3013 3.3013], 'mineral', 'Tantalum');
    % specimen symmetry
    SS = specimenSymmetry('1');

    % plotting convention
    setMTEXpref('xAxisDirection','north');
    setMTEXpref('zAxisDirection','outOfPlane');
    fname = 'euler_angle_data.txt';

    % plotting
    ori = orientation.load(fname,CS,'ColumnNames',{'phi1','Phi','phi2'});
    odf = calcKernelODF(ori,'halfwidth',10*degree,'resolution',5*degree);
    plotPDF(odf,Miller({0,0,2},{1,1,0},{1,1,1},CS));
    setColorRange([0 2]);

    % saving
    exportgraphics(gcf,'texture.png','Resolution',300);
end
```

Crystal symmetry, lattice parameters and ID
Name of input file
(002), (220) and (111) poles will be plotted

Steps to Plot Texture Using Paraview and MTEX-8

MTEX 5.11.1

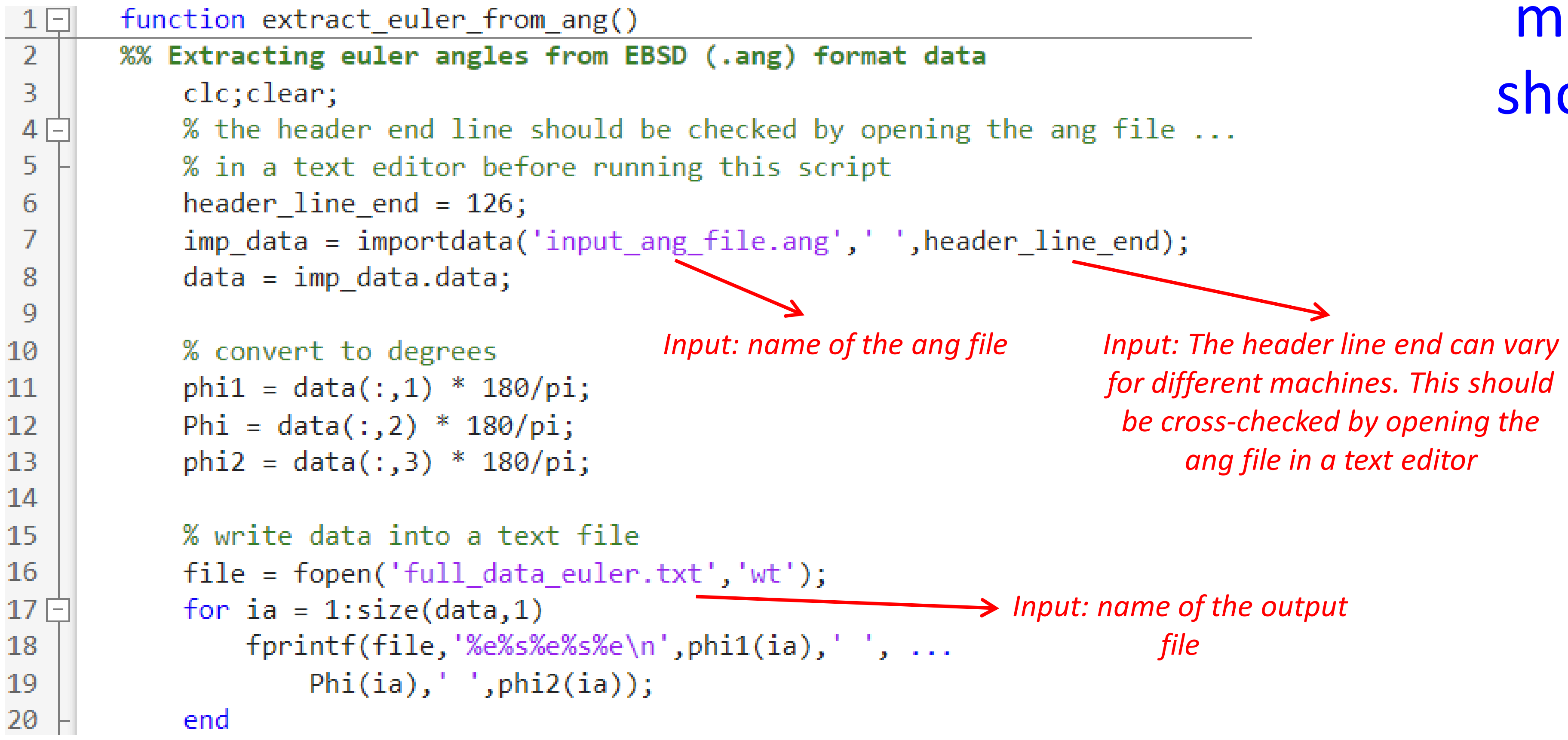


Extracting and Reducing Data from .ang Files

Steps to Extract Data From .ang File

- MATLAB script to extract Euler angles: tools/euler_angle_extractor_ang_file/extract_euler_from_ang.m
- MATLAB version used for this example: R2022b
- Input data: input_ang_file.ang (Aluminium)

* Only the segments marked as input should be altered



```
function extract_euler_from_ang()
%% Extracting euler angles from EBSD (.ang) format data
    clc;clear;
    % the header end line should be checked by opening the ang file ...
    % in a text editor before running this script
    header_line_end = 126;
    imp_data = importdata('input_ang_file.ang',' ',header_line_end);
    data = imp_data.data;

    % convert to degrees
    phi1 = data(:,1) * 180/pi;
    Phi = data(:,2) * 180/pi;
    phi2 = data(:,3) * 180/pi;

    % write data into a text file
    file = fopen('full_data_euler.txt','wt');
    for ia = 1:size(data,1)
        fprintf(file,'%e%s%e%s%e\n',phi1(ia),' ', ...
            Phi(ia),' ',phi2(ia));
    end
```

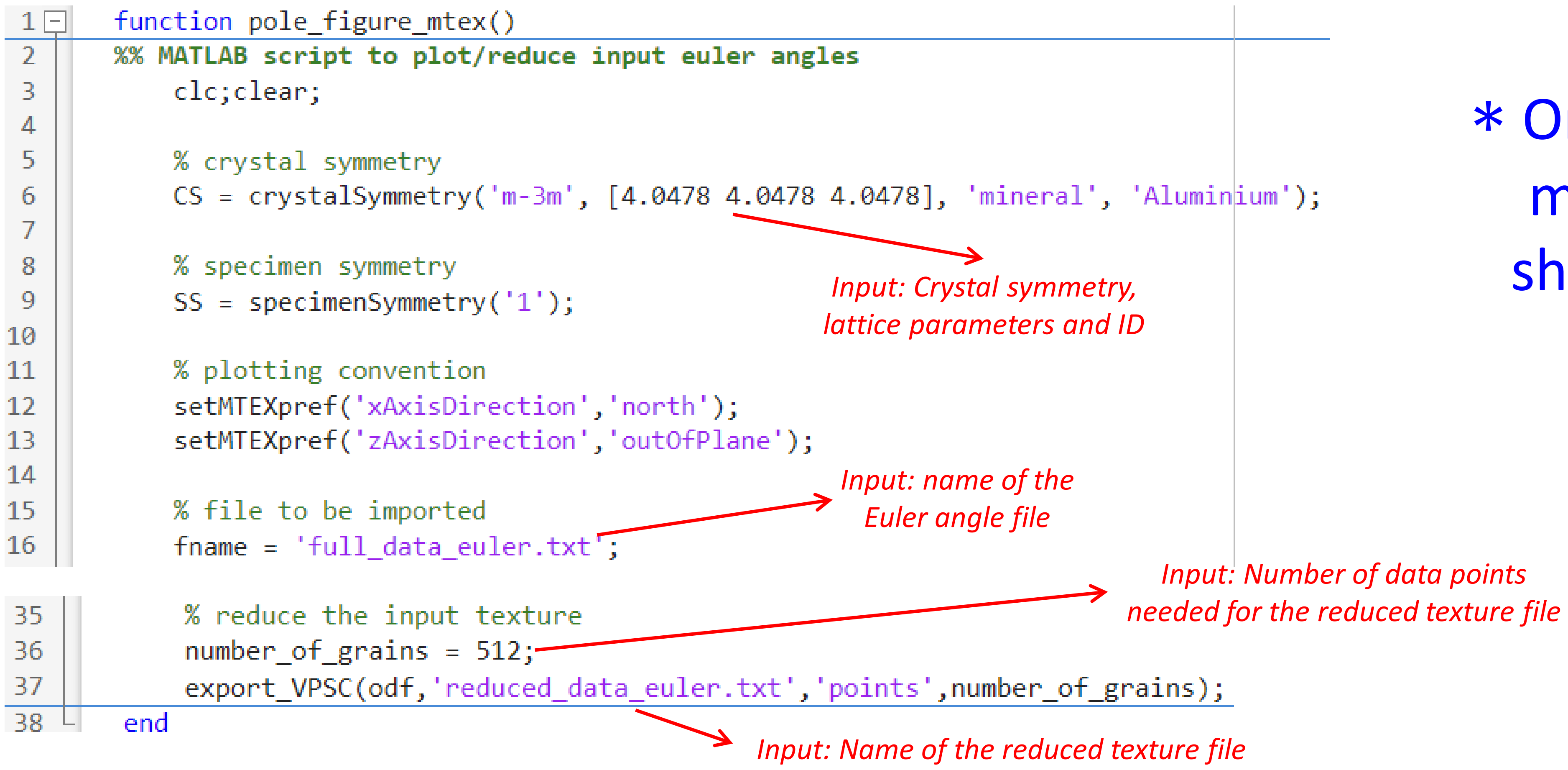
Input: name of the ang file

Input: The header line end can vary for different machines. This should be cross-checked by opening the ang file in a text editor

Input: name of the output file

Steps to Reduce the Extracted Data-1

- MATLAB script used: tools/euler_angle_extractor_ang_file/pole_figure_mtex.m
- MATLAB version used for this example: R2022b
- MTex version: 5.4.0 (*https://mtex-toolbox.github.io/download*)
- Input: Euler angle data generated in the previous slide



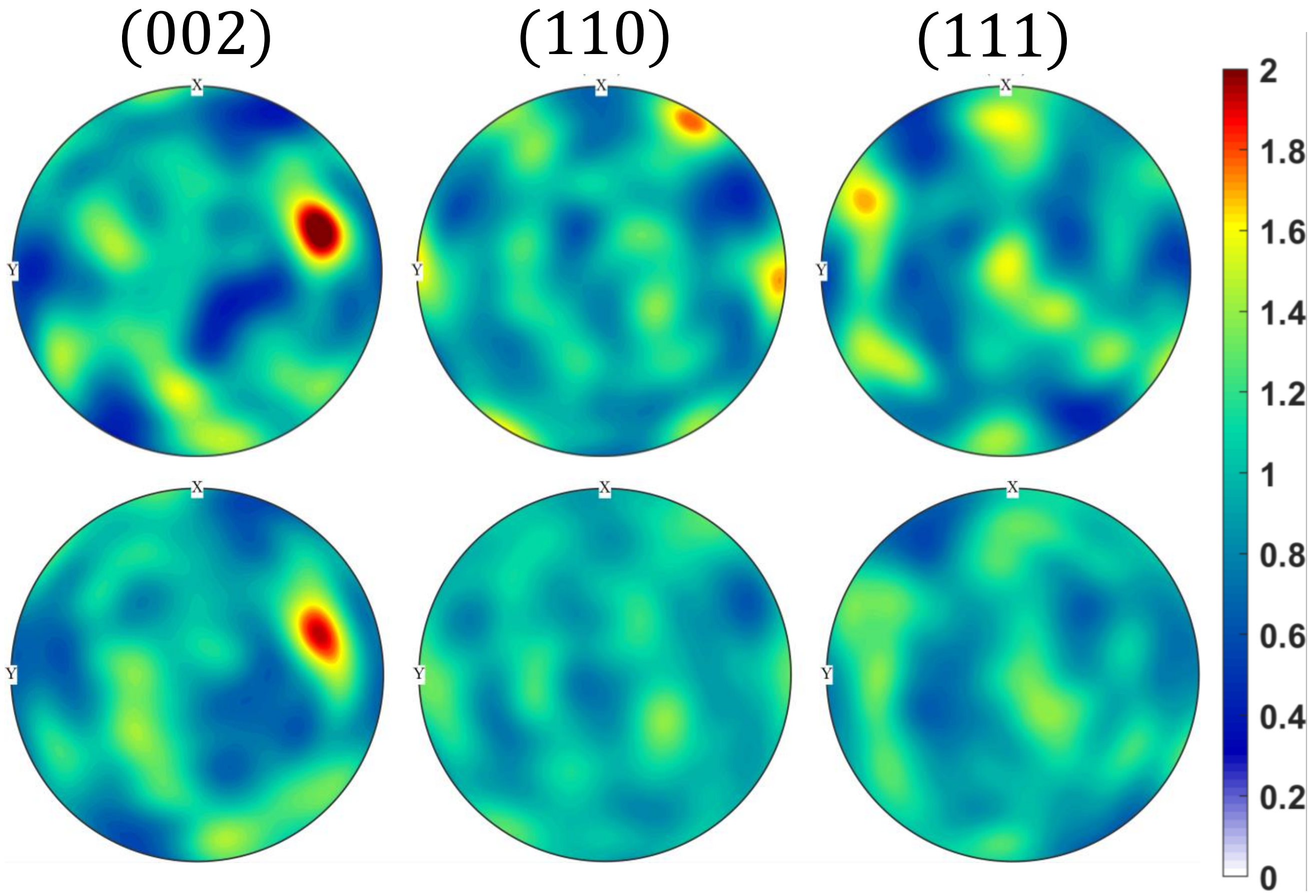
```
function pole_figure_mtex()
%% MATLAB script to plot/reduce input euler angles
    clc;clear;

    % crystal symmetry
    CS = crystalSymmetry('m-3m', [4.0478 4.0478 4.0478], 'mineral', 'Aluminium');

    % specimen symmetry
    SS = specimenSymmetry('1');

    % plotting convention
    setMTEXpref('xAxisDirection','north');
    setMTEXpref('zAxisDirection','outOfPlane');

    % file to be imported
    fname = 'full_data_euler.txt';

    % reduce the input texture
    number_of_grains = 512;
    export_VPSC(odf,'reduced_data_euler.txt','points',number_of_grains);
end
```

Input: Crystal symmetry, lattice parameters and ID
Input: name of the Euler angle file
Input: Number of data points needed for the reduced texture file
Input: Name of the reduced texture file

* Only the segments marked as input should be altered

Steps to Reduce the Extracted Data-2

- MATLAB version used for this example: R2022b
- MTex version: 5.11.1 (https://mtex-toolbox.github.io/download)
- Original data file: tools/euler_angle_extractor_ang_file_mtex_5p11/full_data_euler.txt



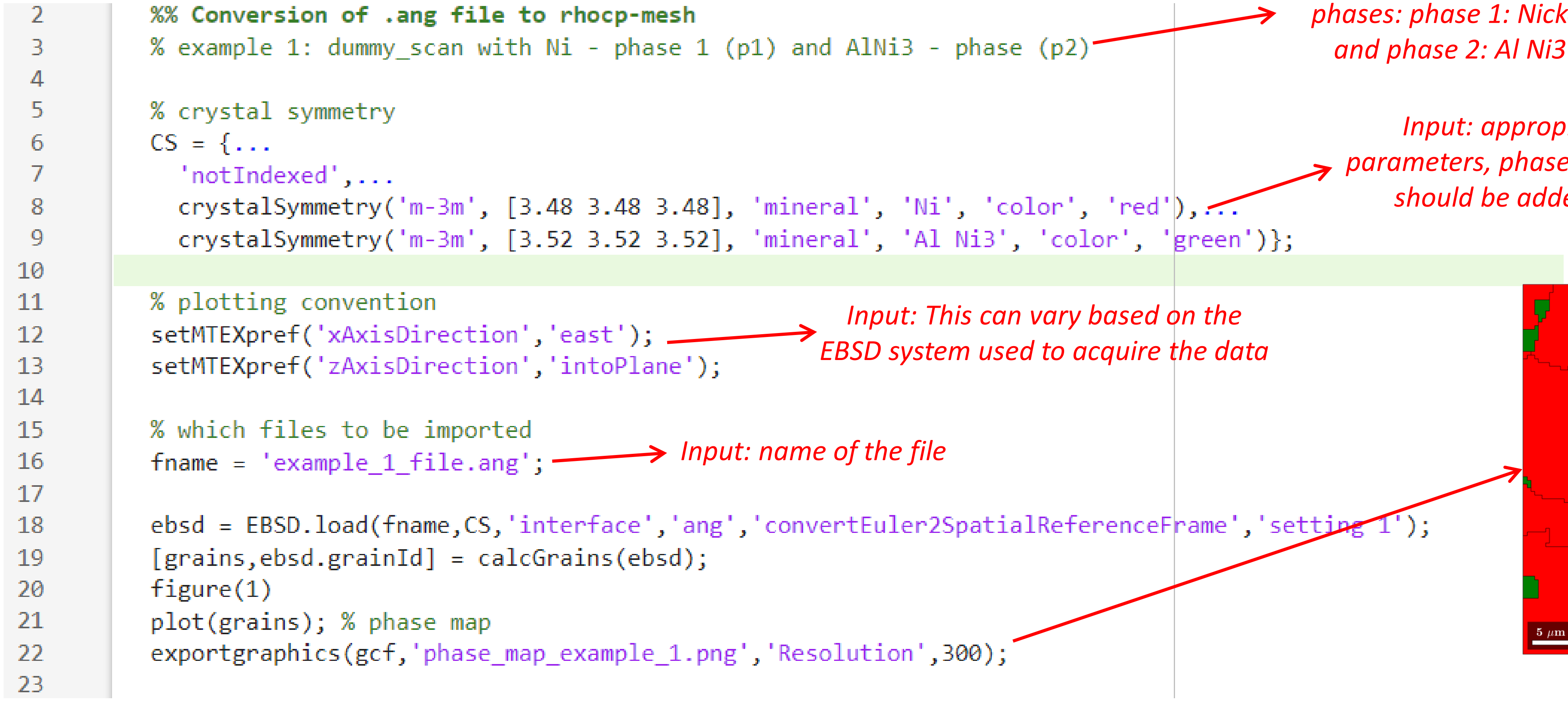
- Alternatively, users can also try the *shuf* command in Linux to reduce the original data file.

Using EBSD Mesh (.ang) in ρ-CP

Steps to Use EBSD Mesh in ρ-CP-1

- MATLAB version used for this example: R2022b
- MTex version: 5.11.1 (https://mtex-toolbox.github.io/download)
- Name of MATLAB script: mtex_ang_to_rhocp_e1.m
- Original data file: tools/ebsd_mesh/example_1_file.ang

* Only the segments marked as input should be altered



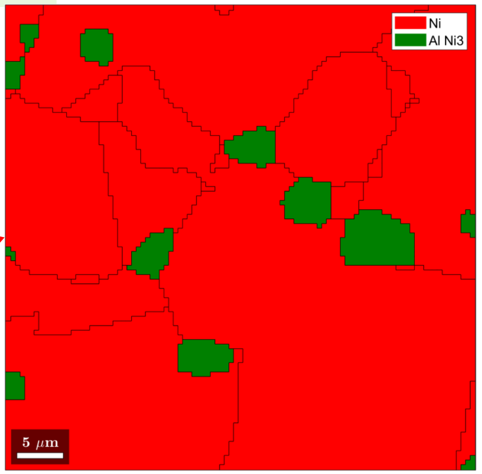
```
%% Conversion of .ang file to rhocp-mesh
% example 1: dummy_scan with Ni - phase 1 (p1) and AlNi3 - phase (p2)

% crystal symmetry
CS = {...
  'notIndexed',...
  crystalSymmetry('m-3m', [3.48 3.48 3.48], 'mineral', 'Ni', 'color', 'red'),...
  crystalSymmetry('m-3m', [3.52 3.52 3.52], 'mineral', 'Al Ni3', 'color', 'green')};

% plotting convention
setMTEXpref('xAxisDirection','east');
setMTEXpref('zAxisDirection','intoPlane');

% which files to be imported
fname = 'example_1_file.ang';

ebsd = EBSD.load(fname,CS,'interface','ang','convertEuler2SpatialReferenceFrame','setting 1');
[grains,ebsd.grainId] = calcGrains(ebsd);
figure(1)
plot(grains); % phase map
exportgraphics(gcf,'phase_map_example_1.png','Resolution',300);
```

The scan file has 2 phases: phase 1: Nickel and phase 2: Al Ni3

Input: appropriate lattice parameters, phases and symmetry should be added over here

Input: This can vary based on the EBSD system used to acquire the data

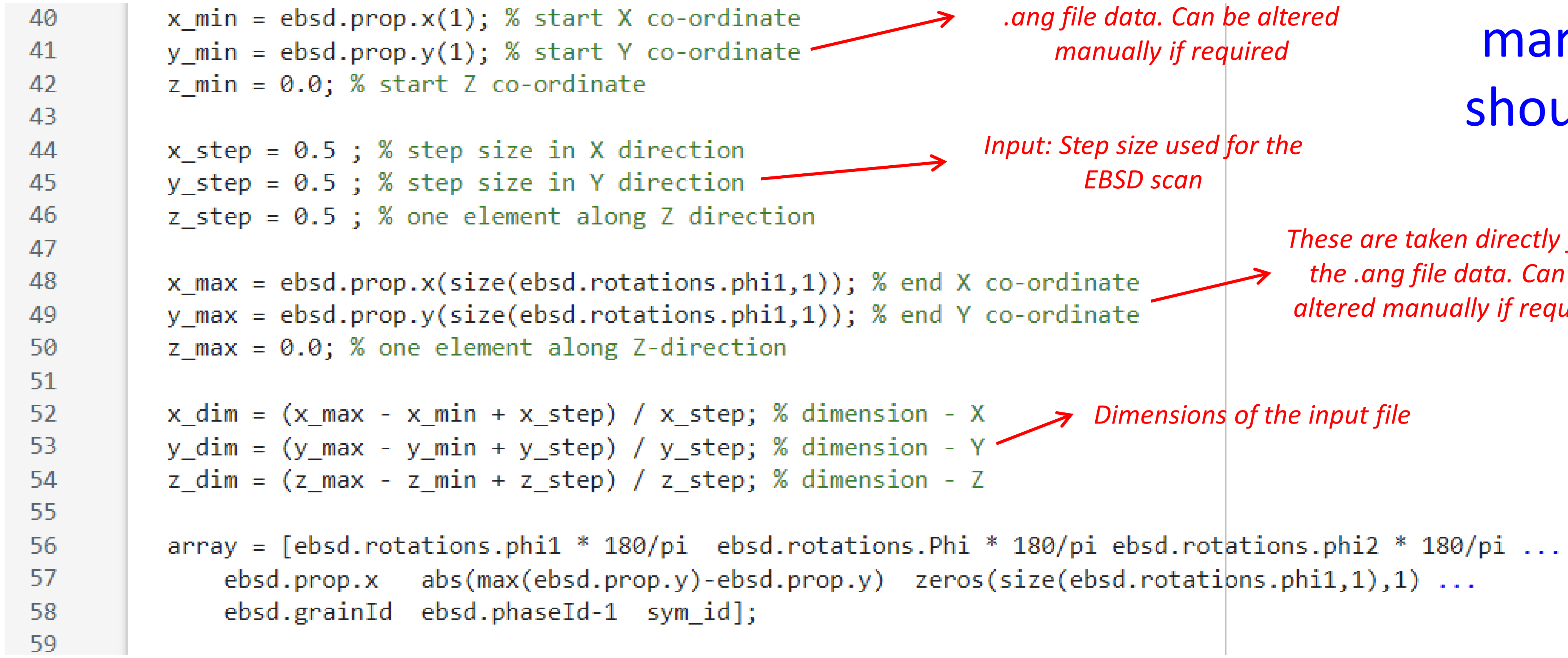
Input: name of the file



Steps to Use EBSD Mesh in ρ-CP-2

- The script generates a 3D dimensional mesh (1 element along the Z direction) from the input EBSD data.
- It is assumed that the input .ang file contains **Euler angles in Bunge format and in radians**.

* Only the segments marked as input should be altered



```
x_min = ebsd.prop.x(1); % start X co-ordinate
y_min = ebsd.prop.y(1); % start Y co-ordinate
z_min = 0.0; % start Z co-ordinate

x_step = 0.5 ; % step size in X direction
y_step = 0.5 ; % step size in Y direction
z_step = 0.5 ; % one element along Z direction

x_max = ebsd.prop.x(size(ebsd.rotations.phi1,1)); % end X co-ordinate
y_max = ebsd.prop.y(size(ebsd.rotations.phi1,1)); % end Y co-ordinate
z_max = 0.0; % one element along Z-direction

x_dim = (x_max - x_min + x_step) / x_step; % dimension - X
y_dim = (y_max - y_min + y_step) / y_step; % dimension - Y
z_dim = (z_max - z_min + z_step) / z_step; % dimension - Z

array = [ebsd.rotations.phi1 * 180/pi  ebsd.rotations.Phi * 180/pi ebsd.rotations.phi2 * 180/pi ...
    ebsd.prop.x   abs(max(ebsd.prop.y)-ebsd.prop.y)  zeros(size(ebsd.rotations.phi1,1),1) ...
    ebsd.grainId  ebsd.phaseId-1  sym_id];

```

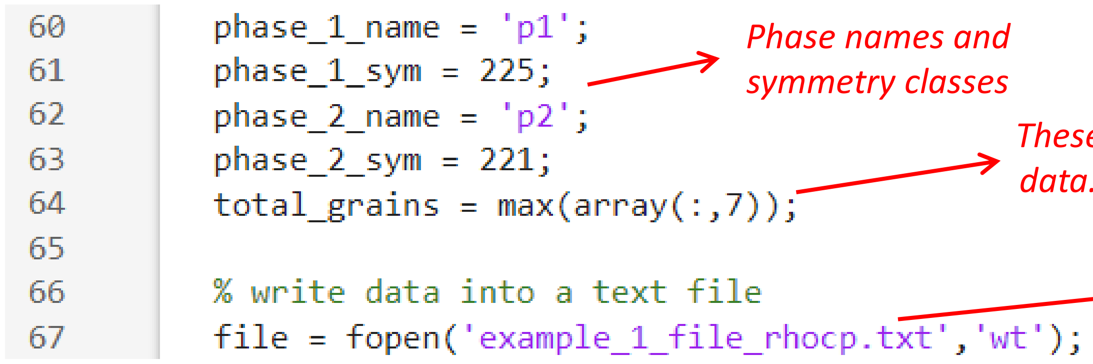
These are taken directly from the .ang file data. Can be altered manually if required

Input: Step size used for the EBSD scan

These are taken directly from the .ang file data. Can be altered manually if required

Dimensions of the input file

Steps to Use EBSD Mesh in ρ-CP-3



```
phase_1_name = 'p1';
phase_1_sym = 225;
phase_2_name = 'p2';
phase_2_sym = 221;
total_grains = max(array(:,7));

% write data into a text file
file = fopen('example_1_file_rhocp.txt','wt');
```

Phase names and symmetry classes

These are taken directly from the .ang file data. Can be altered manually if required

Input: Name of the output file (to be used in ρ-CP)

* Only the segments marked as input should be altered

Structure of the output mesh format (to be used in ρ-CP):

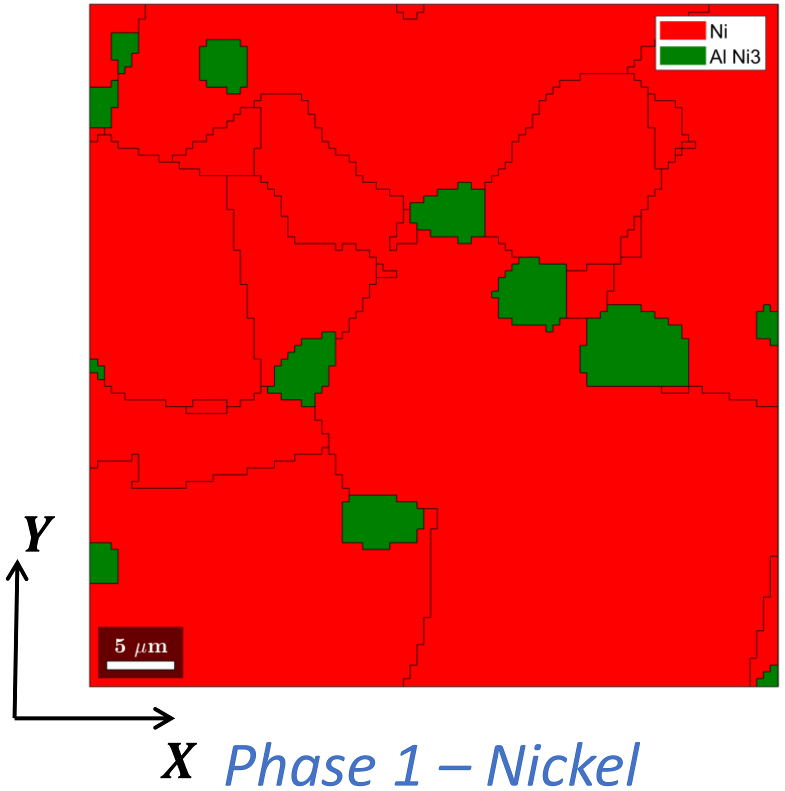
- Column 1: Euler angle "phi1"
- Column 2: Euler angle "PHI"
- Column 3: Euler angle "phi2"
- Column 4: x-coordinate (in microns)
- Column 5: y-coordinate (in microns)
- Column 6: z-coordinate (in microns)
- Column 7: grain number (integer)
- Column 8: phase number (integer)
- Column 9: Symmetry class (from TSL)

Structure of the output file

Steps to Use EBSD Mesh in ρ-CP-4

1 element along Z direction

Example 1

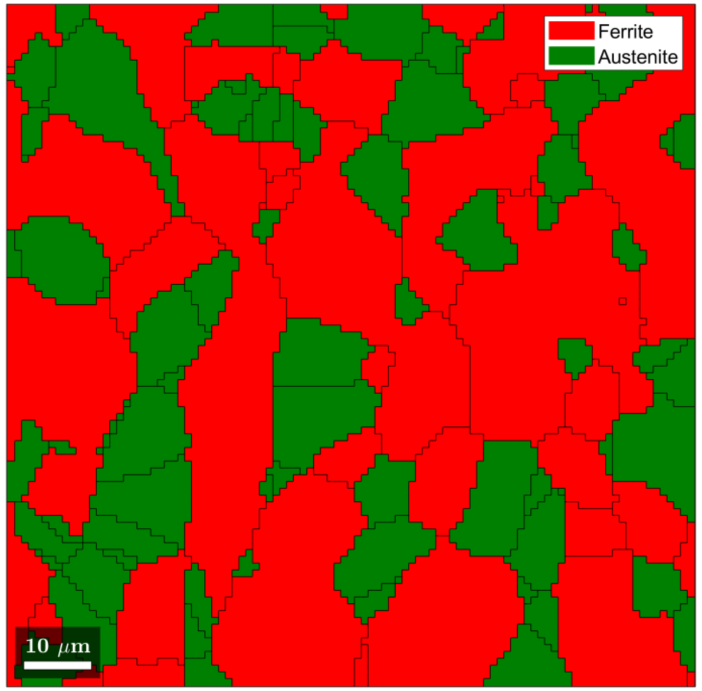


Phase 1 – Nickel

Phase 2 – AlNi3

Output file:
/tools/ebsd_mesh/example_1_file_rhocp.txt

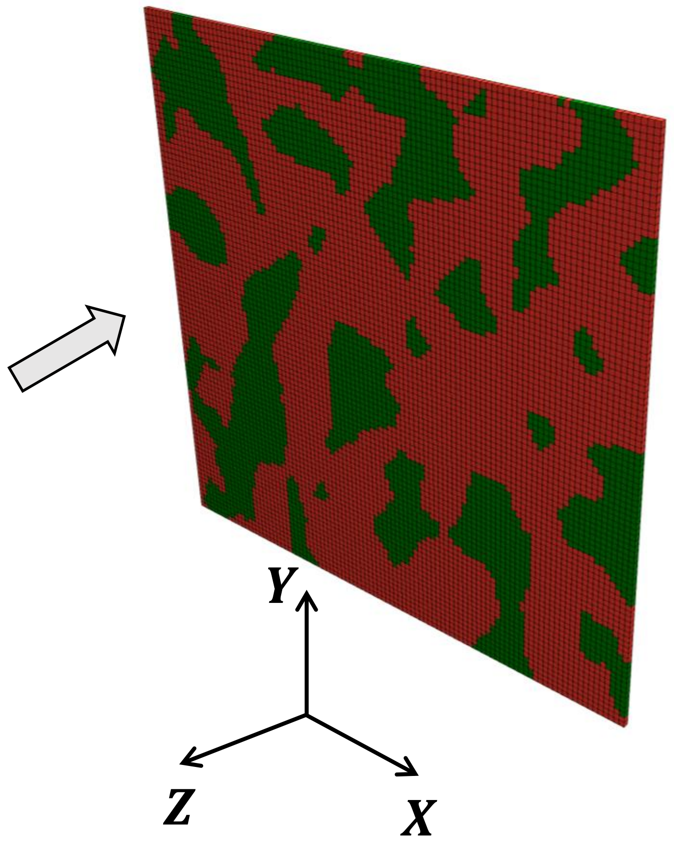
Example 2



Phase 1 – Ferrite

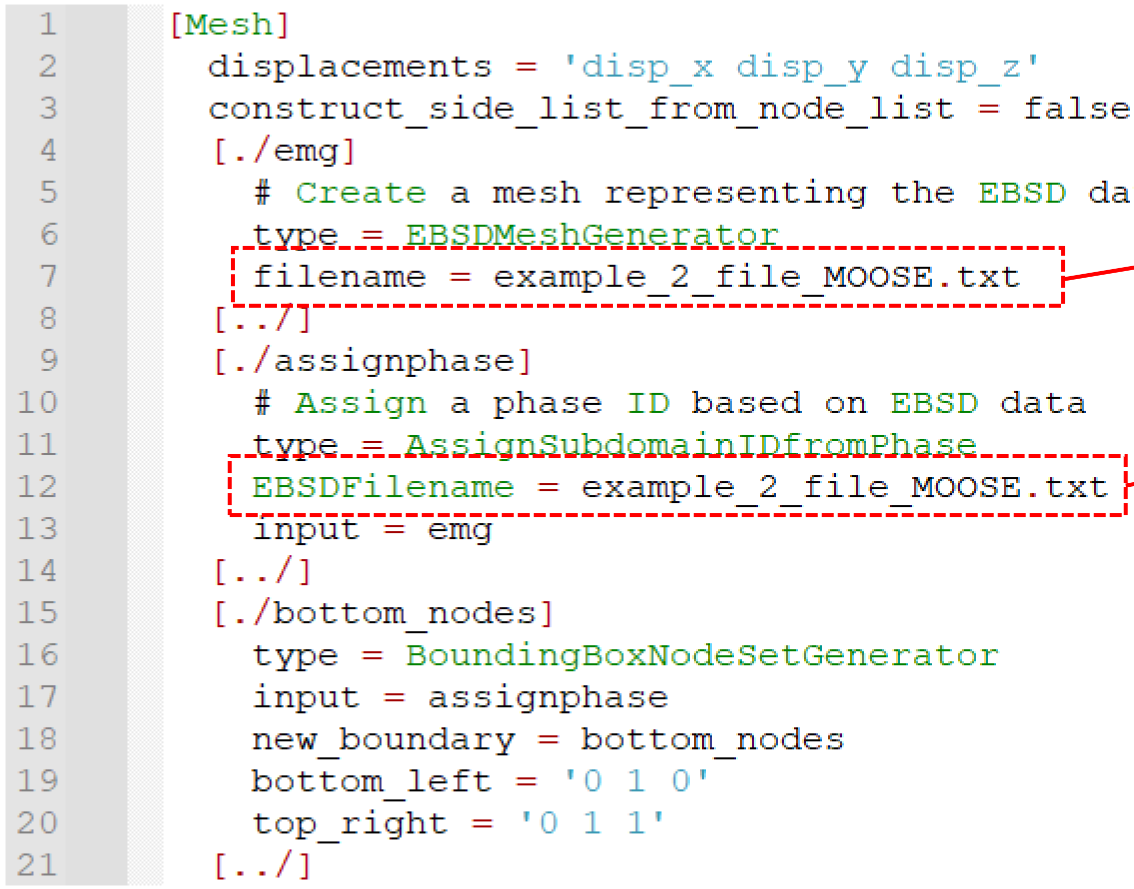
Phase 2 – Austenite

Output file:
/tools/ebsd_mesh/example_2_file_rhocp.txt



Steps to Use EBSD Mesh in ρ-CP-5

- Importing into ρ-CP (/tools/ebsd_mesh_rhocp/example_2/ebsd_sim.i)



```
[Mesh]
  displacements = 'disp_x disp_y disp_z'
  construct_side_list_from_node_list = false
  [./emg]
    # Create a mesh representing the EBSD data
    type = EBSDMeshGenerator
    filename = example_2_file_MOOSE.txt
  [../]
  [./assignphase]
    # Assign a phase ID based on EBSD data
    type = AssignSubdomainIDfromPhase
    EBSDFilename = example_2_file_MOOSE.txt
    input = emg
  [../]
  [./bottom_nodes]
    type = BoundingBoxNodeSetGenerator
    input = assignphase
    new_boundary = bottom_nodes
    bottom_left = '0 1 0'
    top_right = '0 1 1'
  [../]
```

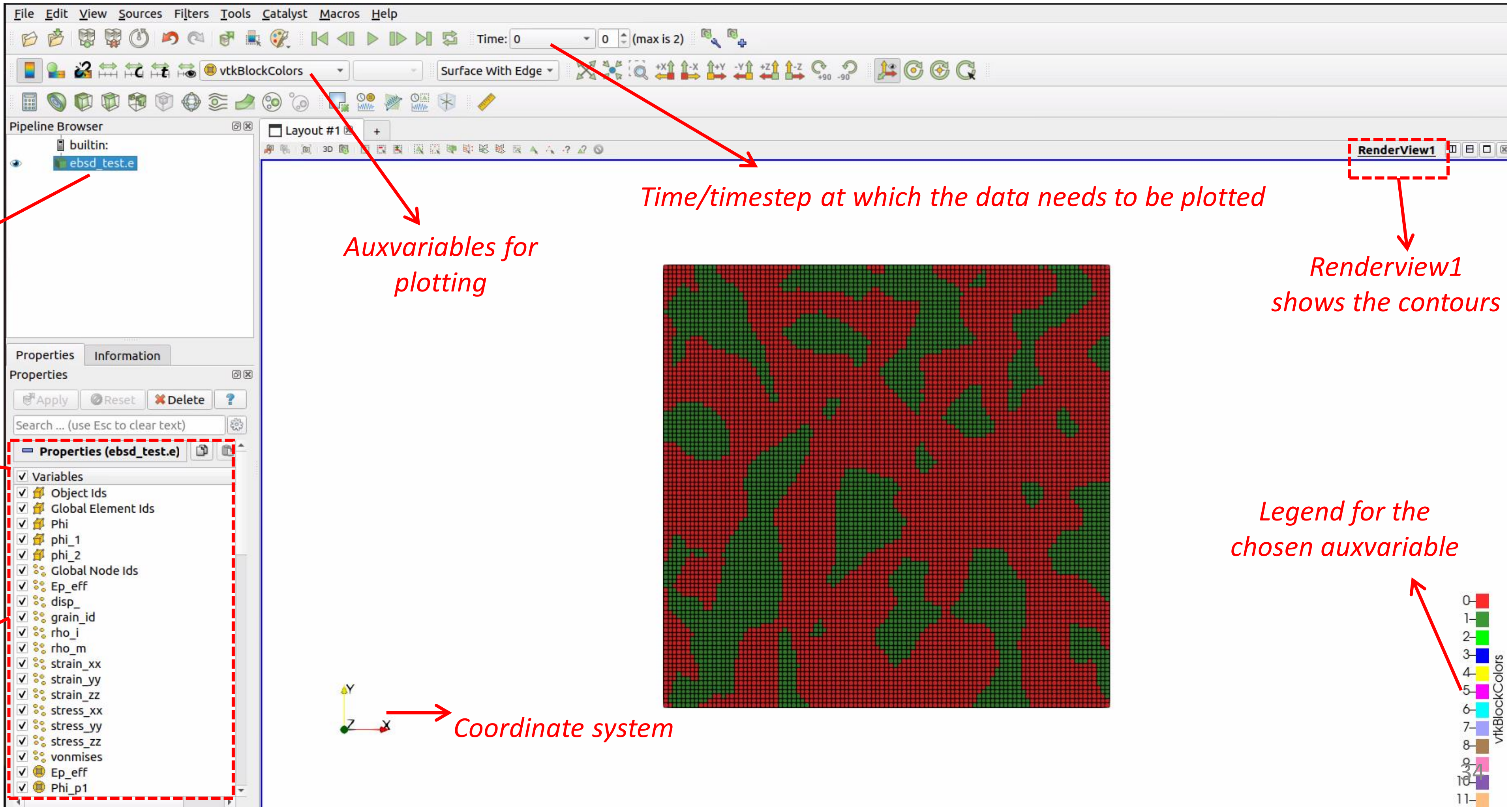
Name of the mesh file along with the extension

Assign phase ID based on the mesh file

Rebuilding .ang File From Exodus Output

Steps to Generate .ang File From Exodus-1

- Paraview version: 5.7.0 (https://www.paraview.org/download/)
- File used for this exercise: /tools/ebsd_mesh_rhocp/example_2/ebsd_test.e



Properties (ebsd_test.e)

- ✓ Variables
- ✓ Object Ids
- ✓ Global Element Ids
- ✓ Phi
- ✓ phi_1
- ✓ phi_2
- ✓ Global Node Ids
- ✓ Ep_eff
- ✓ disp_
- ✓ grain_id
- ✓ rho_i
- ✓ rho_m
- ✓ strain_xx
- ✓ strain_yy
- ✓ strain_zz
- ✓ stress_xx
- ✓ stress_yy
- ✓ stress_zz
- ✓ vonmises
- ✓ Ep_eff
- ✓ Phi_p1

filename

Auxvariables for plotting

Time/timestep at which the data needs to be plotted

Renderview1 shows the contours

Select the desired variables by checking the respective boxes under 'Properties' bar

Legend for the chosen auxvariable

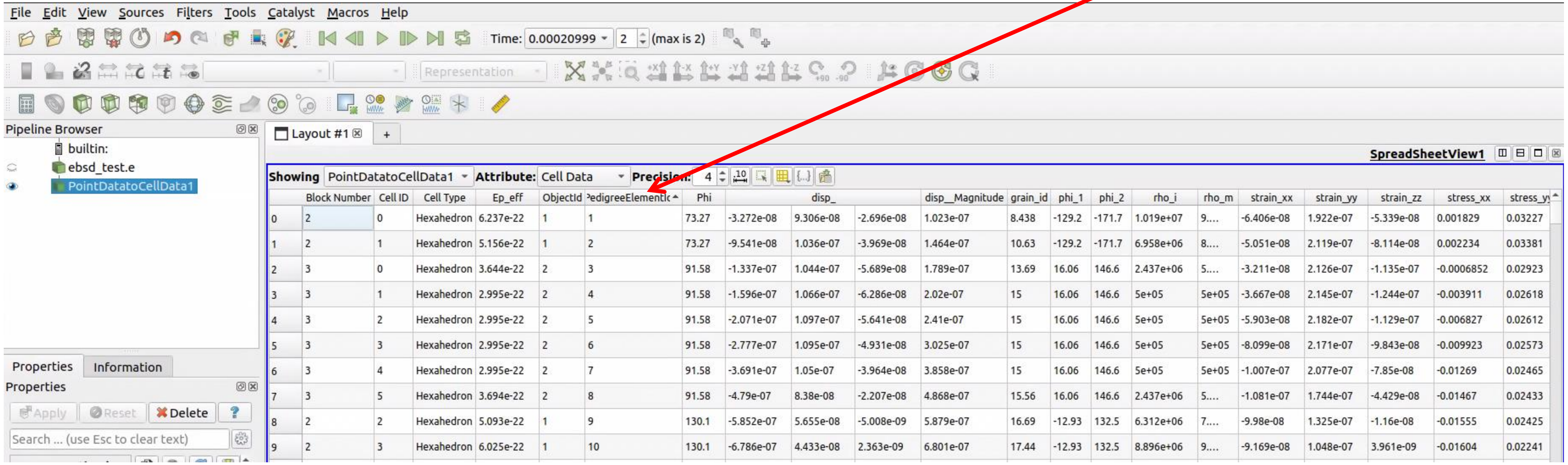
Auxvariables from input file appear over here

Coordinate system

Steps to Generate .ang File From Exodus-2

(Follow instructions given in slide 14-18)

1. Select the desired time/timestep
2. Go to Filters→Alphabetical→Cell Data to Point Data
3. Select Apply
4. The highlighted block should be placed on the CellDatatoPointData.
5. Right click on Renderview1 and select spreadsheet view.
6. Before exporting the spreadsheet, sort the data by clicking on "*PedigreeElementID*" column.

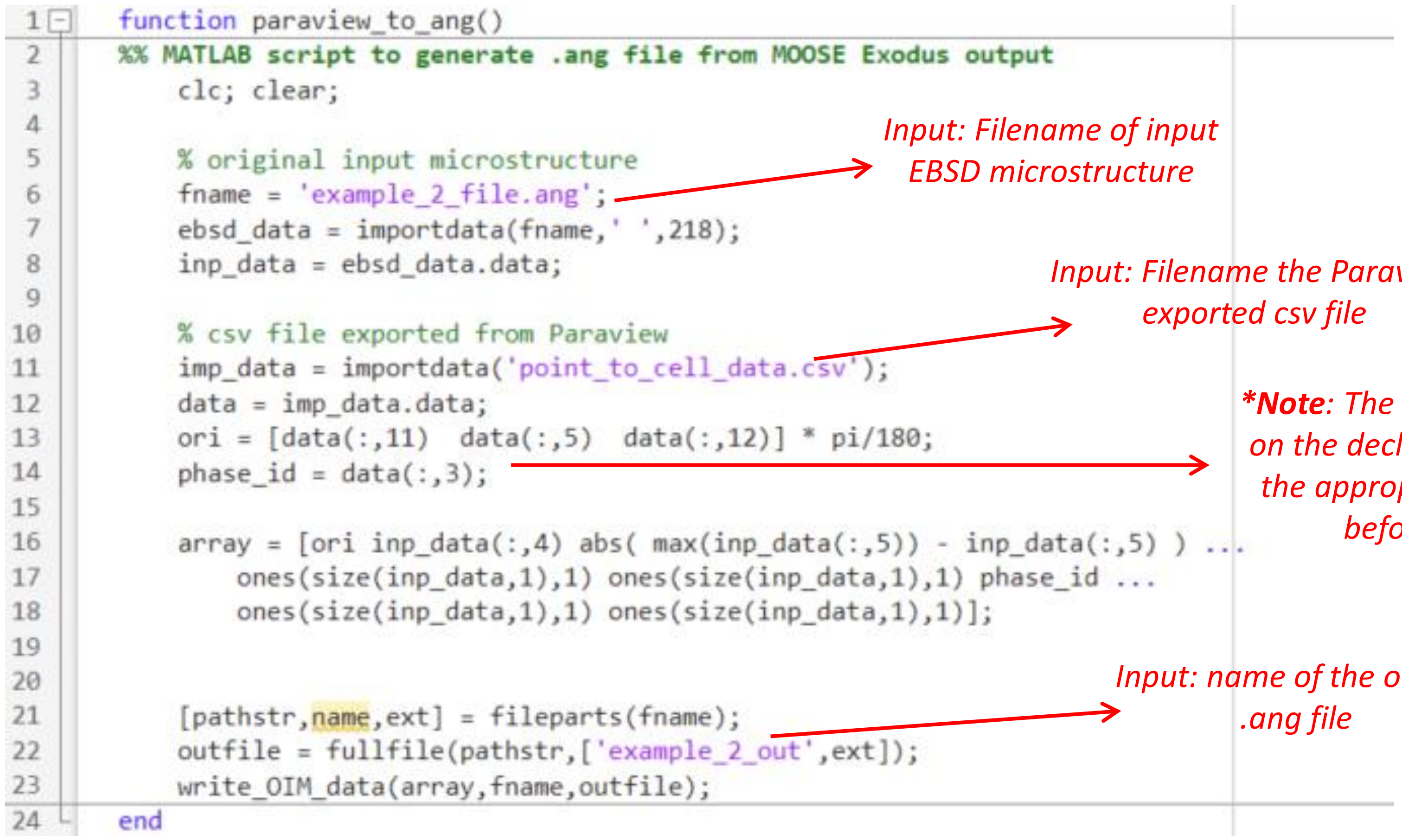


Showing PointDatatoCellData1 Attribute: Cell Data Precision: 4

	Block Number	Cell ID	Cell Type	Ep_eff	ObjectId	PedigreeElementIc	Phi	disp_			disp__Magnitude	grain_id	phi_1	phi_2	rho_i	rho_m	strain_xx	strain_yy	strain_zz	stress_xx	stress_yy
0	2	0	Hexahedron	6.237e-22	1	1	73.27	-3.272e-08	9.306e-08	-2.696e-08	1.023e-07	8.438	-129.2	-171.7	1.019e+07	9....	-6.406e-08	1.922e-07	-5.339e-08	0.001829	0.03227
1	2	1	Hexahedron	5.156e-22	1	2	73.27	-9.541e-08	1.036e-07	-3.969e-08	1.464e-07	10.63	-129.2	-171.7	6.958e+06	8....	-5.051e-08	2.119e-07	-8.114e-08	0.002234	0.03381
2	3	0	Hexahedron	3.644e-22	2	3	91.58	-1.337e-07	1.044e-07	-5.689e-08	1.789e-07	13.69	16.06	146.6	2.437e+06	5....	-3.211e-08	2.126e-07	-1.135e-07	-0.0006852	0.02923
3	3	1	Hexahedron	2.995e-22	2	4	91.58	-1.596e-07	1.066e-07	-6.286e-08	2.02e-07	15	16.06	146.6	5e+05	5e+05	-3.667e-08	2.145e-07	-1.244e-07	-0.003911	0.02618
4	3	2	Hexahedron	2.995e-22	2	5	91.58	-2.071e-07	1.097e-07	-5.641e-08	2.41e-07	15	16.06	146.6	5e+05	5e+05	-5.903e-08	2.182e-07	-1.129e-07	-0.006827	0.02612
5	3	3	Hexahedron	2.995e-22	2	6	91.58	-2.777e-07	1.095e-07	-4.931e-08	3.025e-07	15	16.06	146.6	5e+05	5e+05	-8.099e-08	2.171e-07	-9.843e-08	-0.009923	0.02573
6	3	4	Hexahedron	2.995e-22	2	7	91.58	-3.691e-07	1.05e-07	-3.964e-08	3.858e-07	15	16.06	146.6	5e+05	5e+05	-1.007e-07	2.077e-07	-7.85e-08	-0.01269	0.02465
7	3	5	Hexahedron	3.694e-22	2	8	91.58	-4.79e-07	8.38e-08	-2.207e-08	4.868e-07	15.56	16.06	146.6	2.437e+06	5....	-1.081e-07	1.744e-07	-4.429e-08	-0.01467	0.02433
8	2	2	Hexahedron	5.093e-22	1	9	130.1	-5.852e-07	5.655e-08	-5.008e-09	5.879e-07	16.69	-12.93	132.5	6.312e+06	7....	-9.98e-08	1.325e-07	-1.16e-08	-0.01555	0.02425
9	2	3	Hexahedron	6.025e-22	1	10	130.1	-6.786e-07	4.433e-08	2.363e-09	6.801e-07	17.44	-12.93	132.5	8.896e+06	9....	-9.169e-08	1.048e-07	3.961e-09	-0.01604	0.02241

Steps to Generate .ang File From Exodus-3

7. Use the MATLAB script to generate .ang file from the Paraview exported csv file.
8. MATLAB script used: /tools/ebsd_mesh_data_extraction/paraview_to_ang.m



```
function paraview_to_ang()
%% MATLAB script to generate .ang file from MOOSE Exodus output
    clc; clear;

    % original input microstructure
    fname = 'example_2_file.ang';
    ebsd_data = importdata(fname,' ',218);
    inp_data = ebsd_data.data;

    % csv file exported from Paraview
    imp_data = importdata('point_to_cell_data.csv');
    data = imp_data.data;
    ori = [data(:,11)  data(:,5)  data(:,12)] * pi/180;
    phase_id = data(:,3);

    array = [ori inp_data(:,4) abs( max(inp_data(:,5)) - inp_data(:,5) ) ...
        ones(size(inp_data,1),1) ones(size(inp_data,1),1) phase_id ...
        ones(size(inp_data,1),1) ones(size(inp_data,1),1)];


    [pathstr,name,ext] = fileparts(fname);
    outfile = fullfile(pathstr,['example_2_out',ext]);
    write_OIM_data(array,fname,outfile);
end
```

Input: Filename of input EBSD microstructure

Input: Filename the Paraview exported csv file

***Note**: The column ID numbers can change based on the declaration of Auxvariables. Please check the appropriate column numbers in the csv file before running this MATLAB script.*

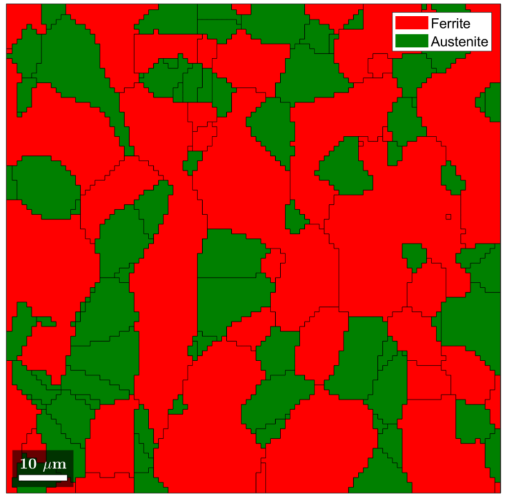
Input: name of the output .ang file

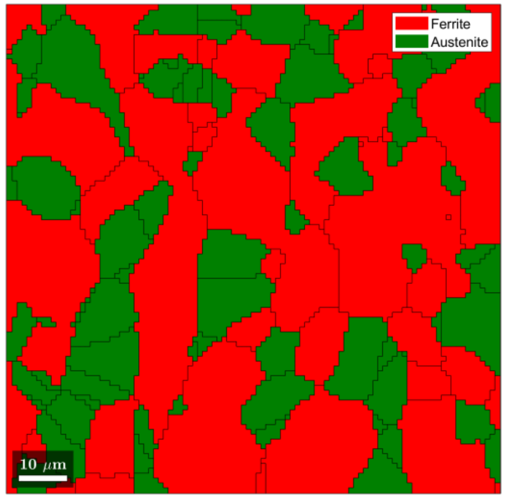
Steps to Generate .ang File From Exodus-4

- Verify the output .ang file by comparing with the initial microstructure.

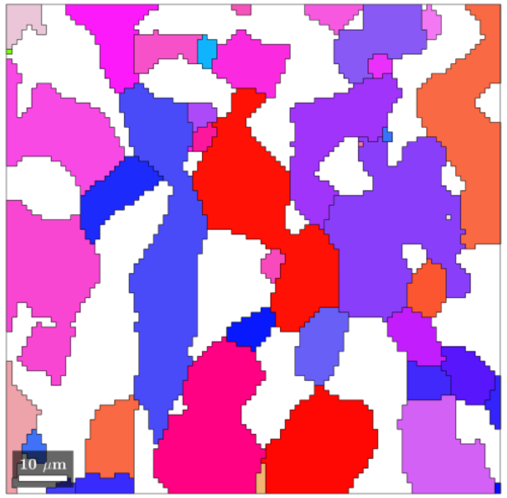
Example 2-Input

Example 2-Output at t=0.0002 s





Phase 1 – Ferrite
Phase 2 – Austenite



Ferrite