DigiPorFlow

Natural Environment Research Council

HERIOT WATT UNIVERSITY

From micro-Ct images to OpenFOAM simulations

Dr Julien Maes

The DigiPorFlow group
Institute of GeoEnergy Engineering
Heriot-Watt University

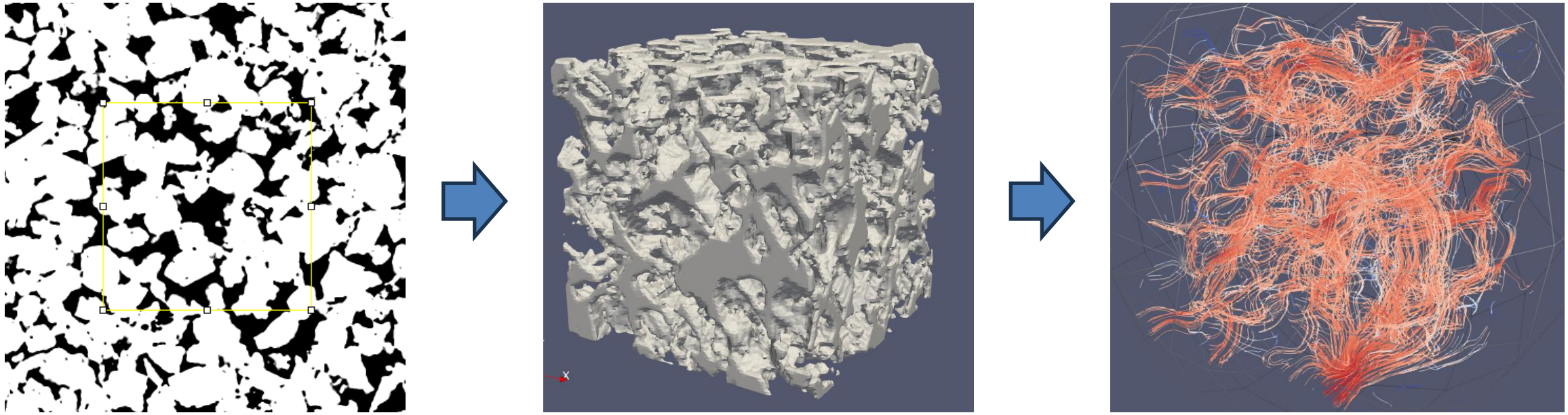
EPSRC
Engineering and Physical Sciences Research Council

Requirement

- Linux distribution (or virtual machines)
- OpenFOAM (preferably ESI but adaptable to any)
- python 3
- Fiji imageJ: https://imagej.net/software/fiji/downloads
- Download training folder: https://github.com/GeoChemFoam/GeoChemFoam/tree/main/Examples/FoamIberiaTraining
- Optional: install GeoChemFoam: https://github.com/GeoChemFoam

Summary

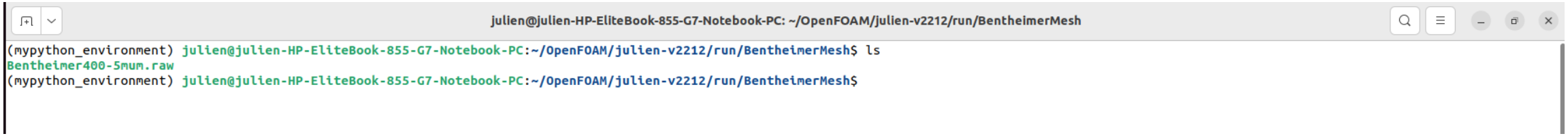
- Two approaches for:



- First approach: Image > stl > snappyHexMesh
- Second approach: Image > Volume-Of-Solid description > modified simpleFoam solver for immersed boundary condition (Darcy-Brinkman-Stokes)

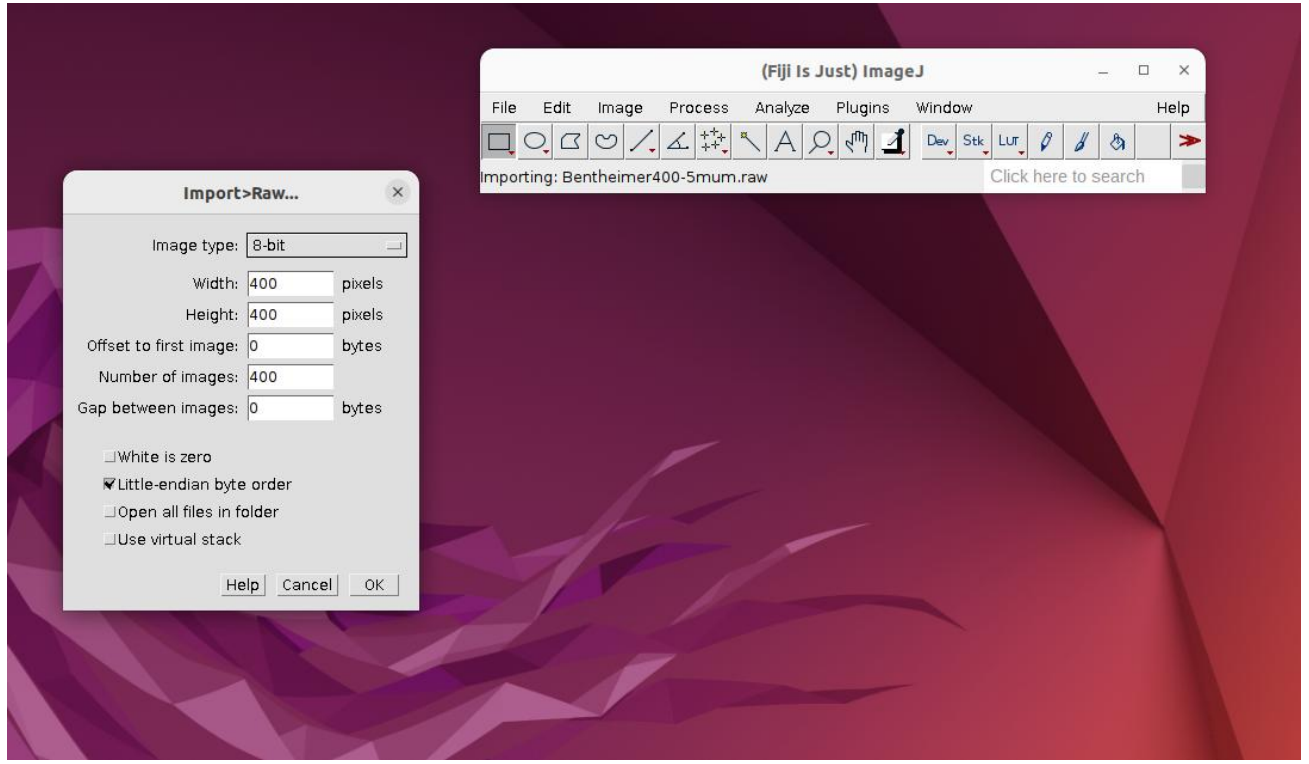
Standard approach (1/31)

- Dowload the Bentheimer image from: https://github.com/GeoChemFoam/GeoChemFoam/tree/main/Examples/FoamIberiaTraining
- Or clone the repostory
- git clone https://github.com/GeoChemFoam/GeoChemFoam.git
- Source OpenFOAM bashrc file
- Copy the image in a folder in your OpenFOAM run directory, in a folder called BentheimerMesh



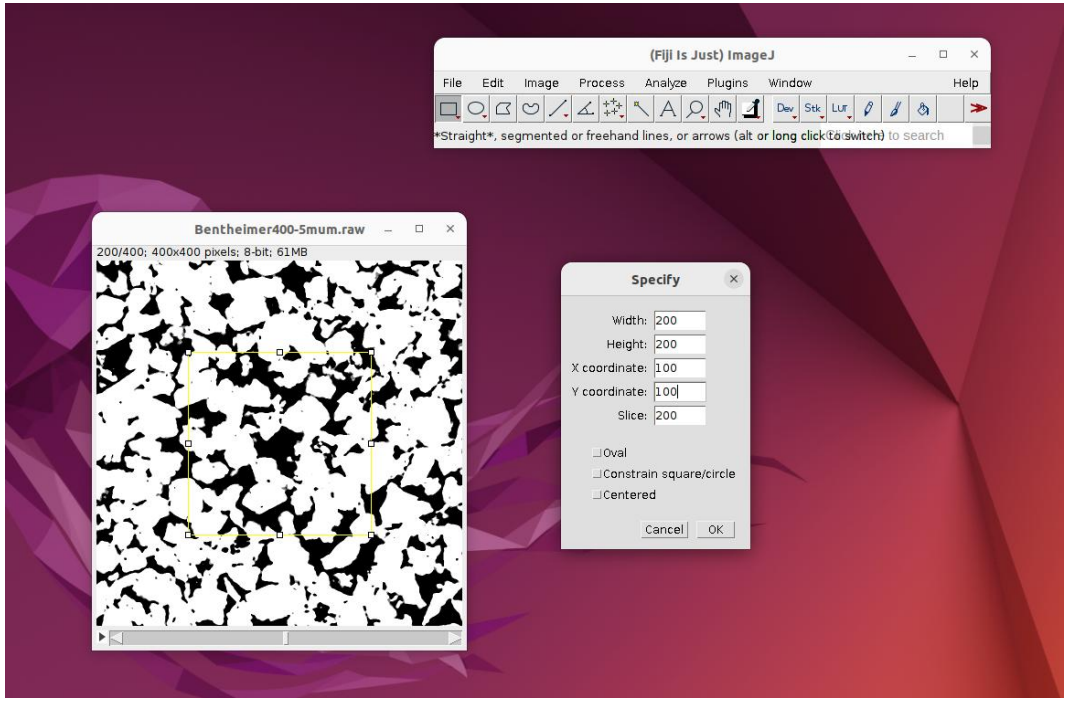
Standard approach (2/31)

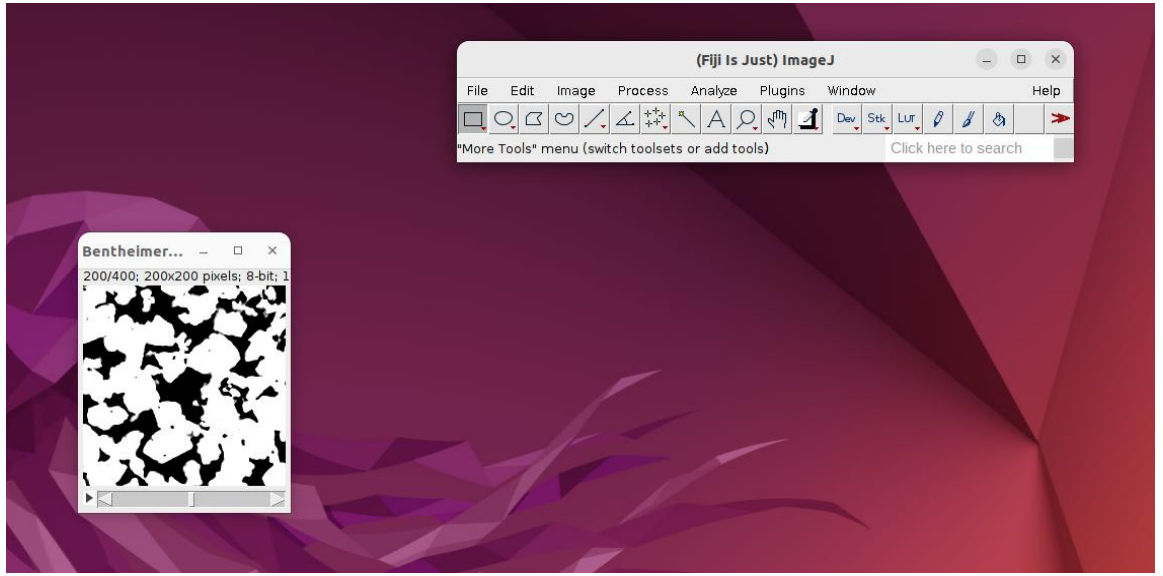
- Open Fiji imageJ-linux64
- Import the raw image. File > Import raw
- Select the image type (8 bit) and the dimension (400x400x400)



Standard approach (3/31)

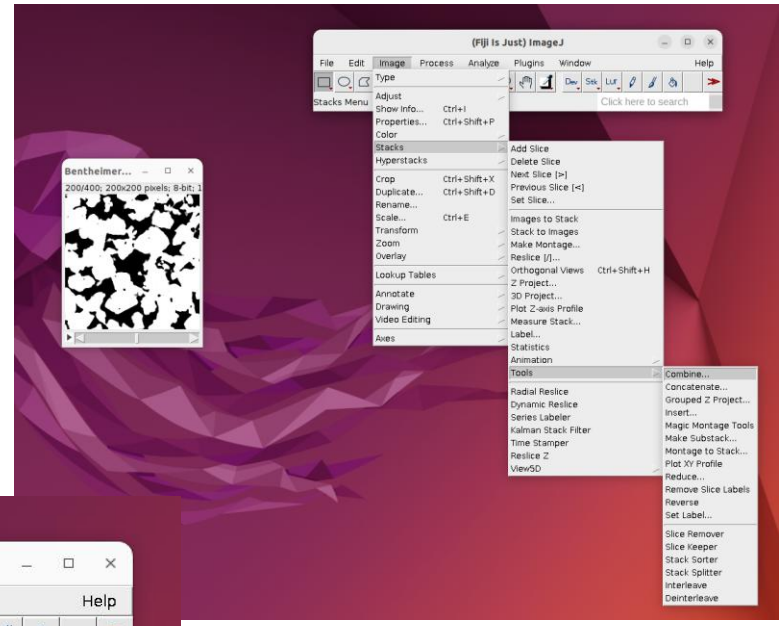
- In the image, white (255) are the solid grains and black (0) are the pores
- Select a smaller window of 200x200 at the center of the image
- Then apply Image>Crop. Now the image is 200x200x400

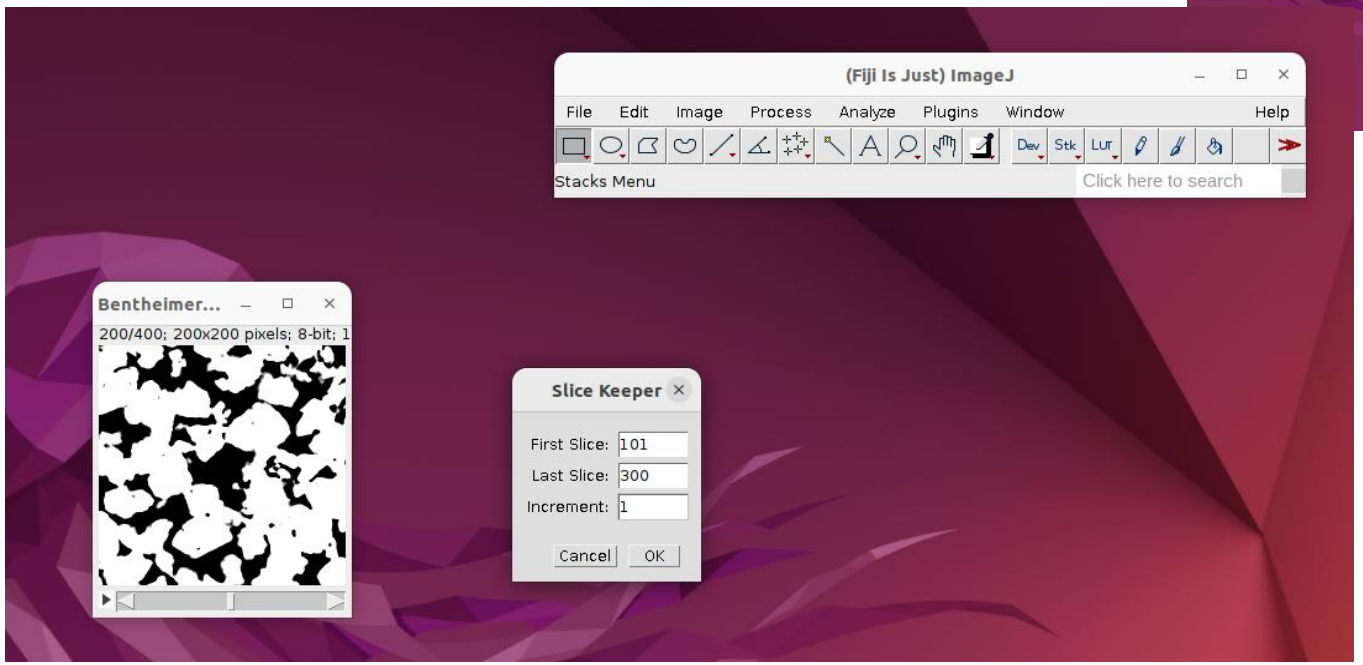




Standard approach (4/31)

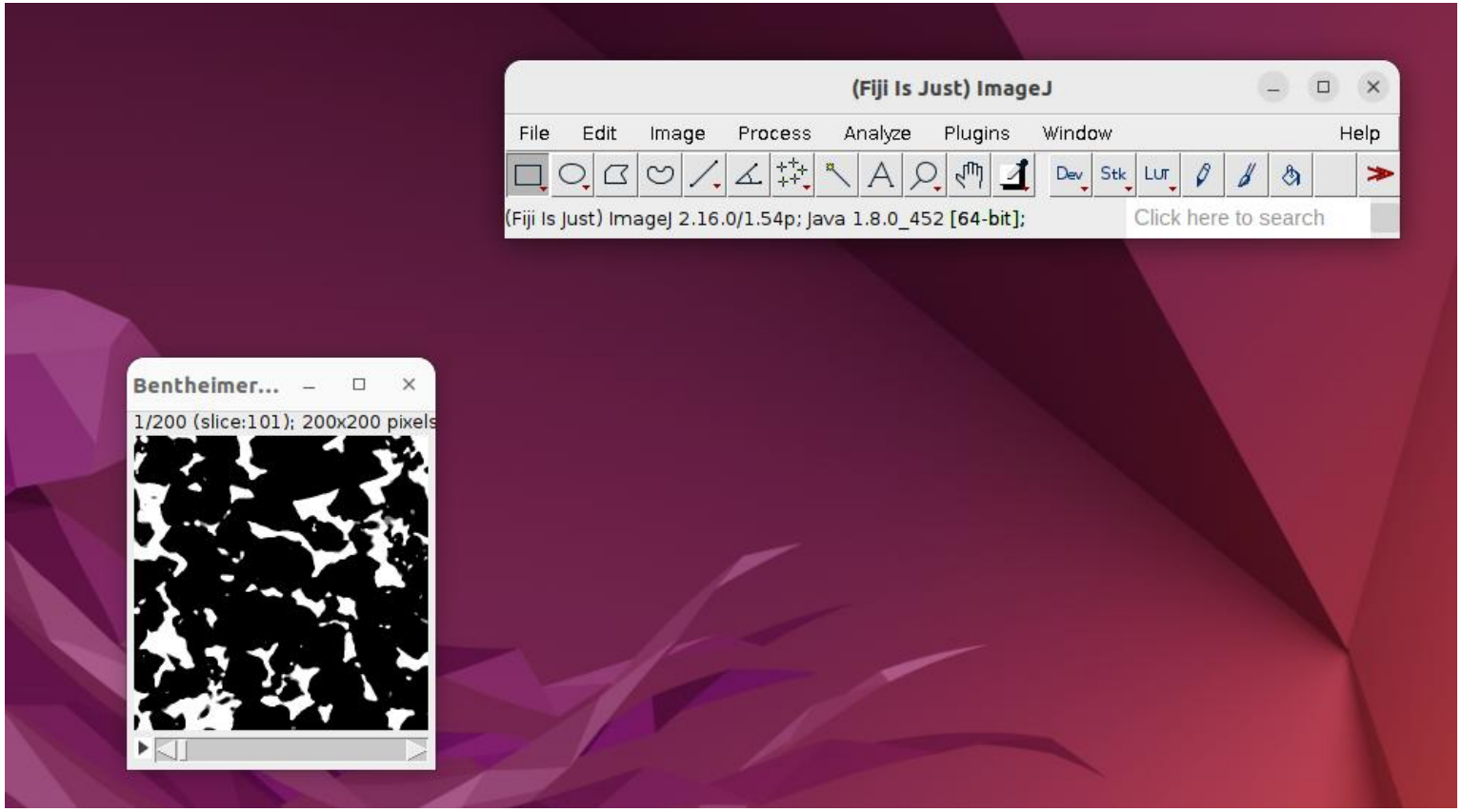
- Select Image>Staccks>Tools>Slice Keeper
- Select first slice 101, last slice 300, increment 1, taking the middle part of the image
- A new window appear. Close the old window. The imageis now 200x200x200





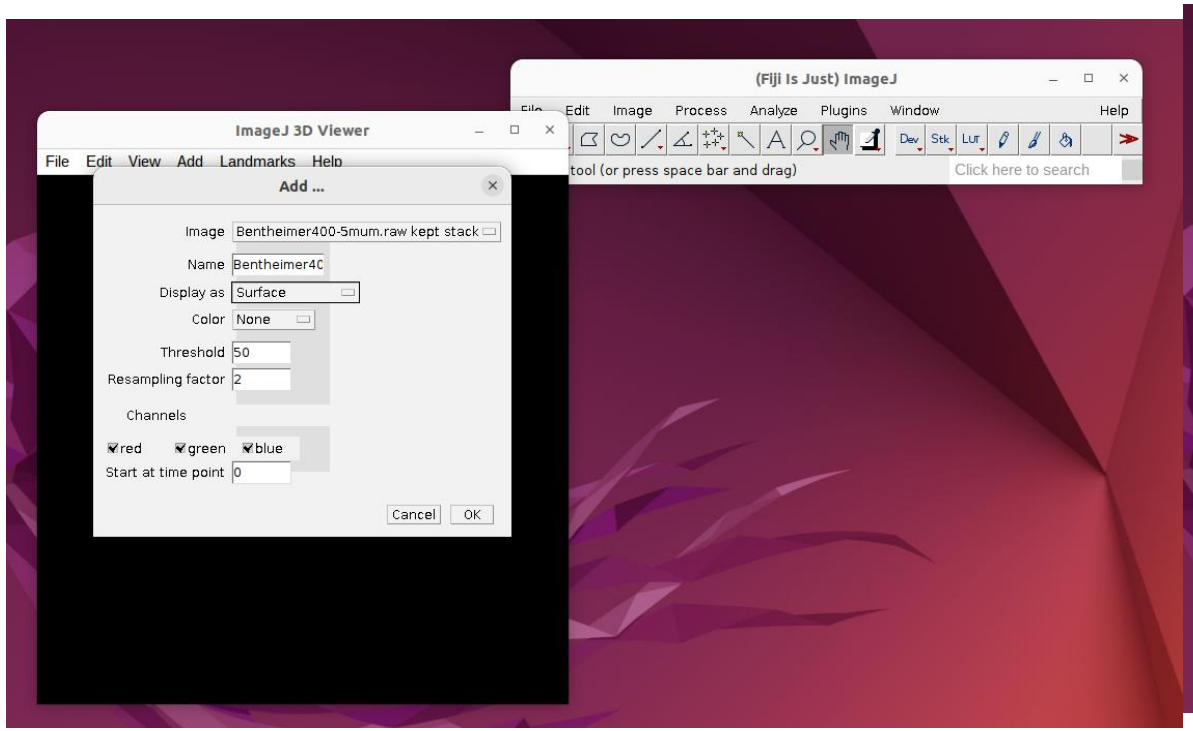
Standard approach (5/31)

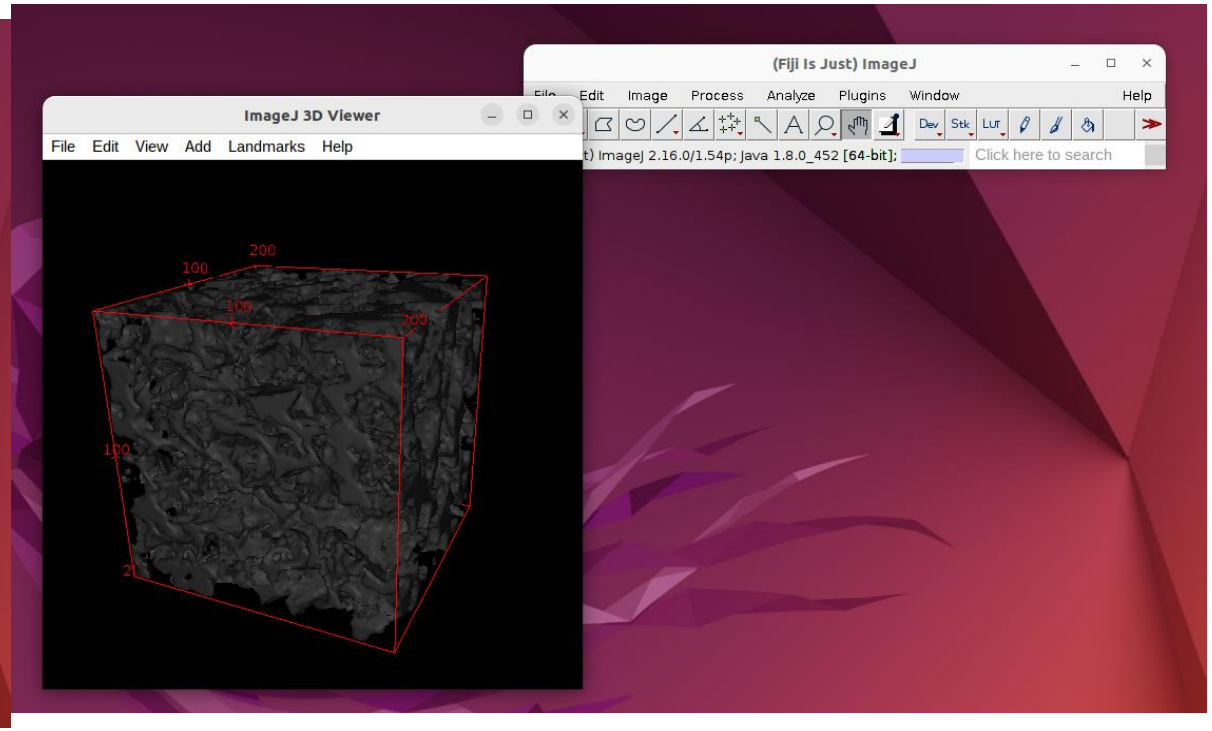
- Invert the image. Pores=255 and Solid=0 now



Standard approach (6/31)

- Open the 3D viewer Plugin>3D viewer
- Select: Display as Surface
- The pore surface will appear





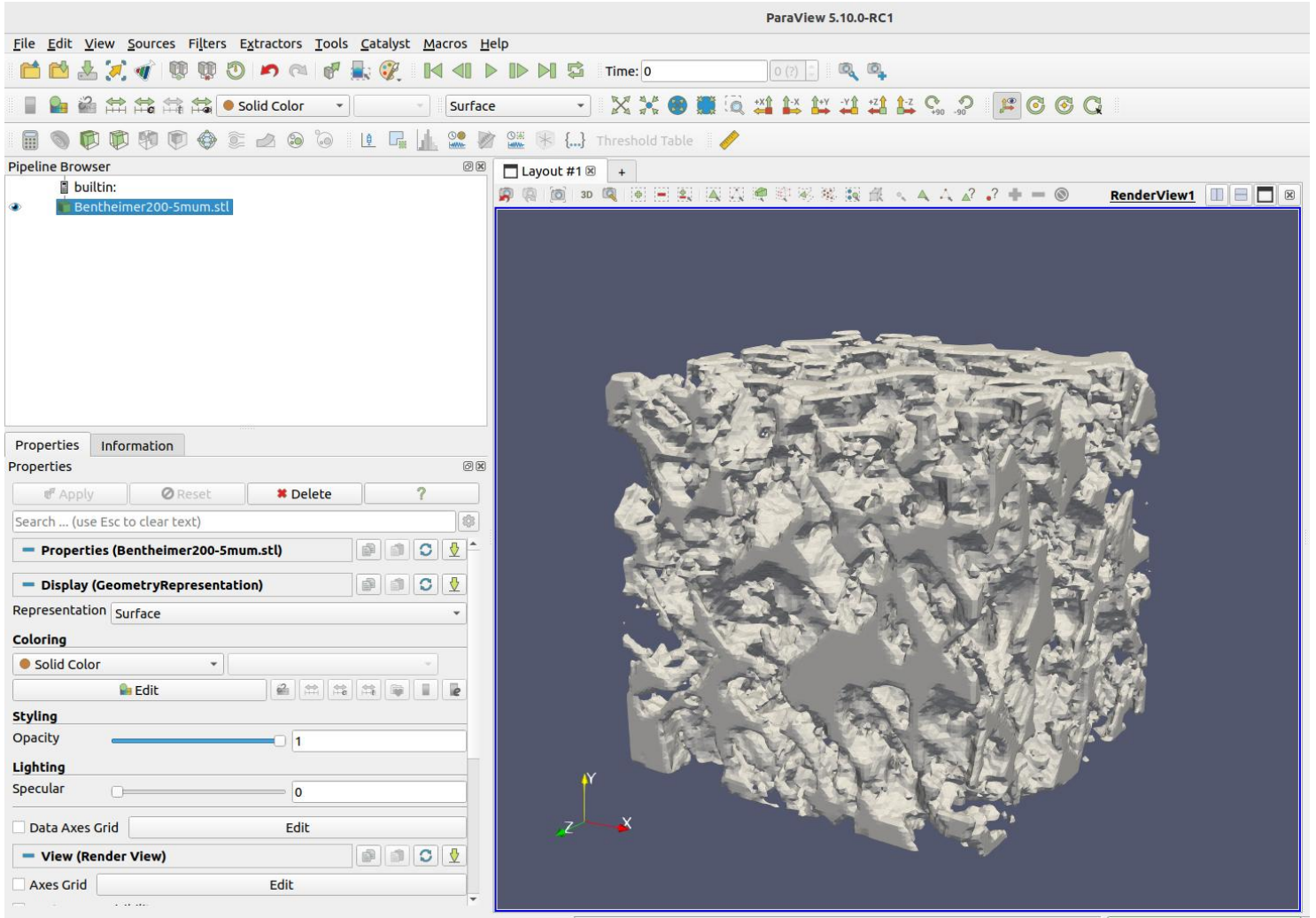
Standard approach (7/31)

- Export the image: File > Export Surface > STL (binary)
- Save it as bentheimer200-5mum.stl inside the OpenFOAM folder

```
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerMesh$ ls
Bentheimer400-5mum.raw
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerMesh$ ls
Bentheimer200-5mum.stl  Bentheimer400-5mum.raw
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerMesh$
```

Standard approach (8/31)

- The stl can be opened with paraview



Standard approach (9/31)

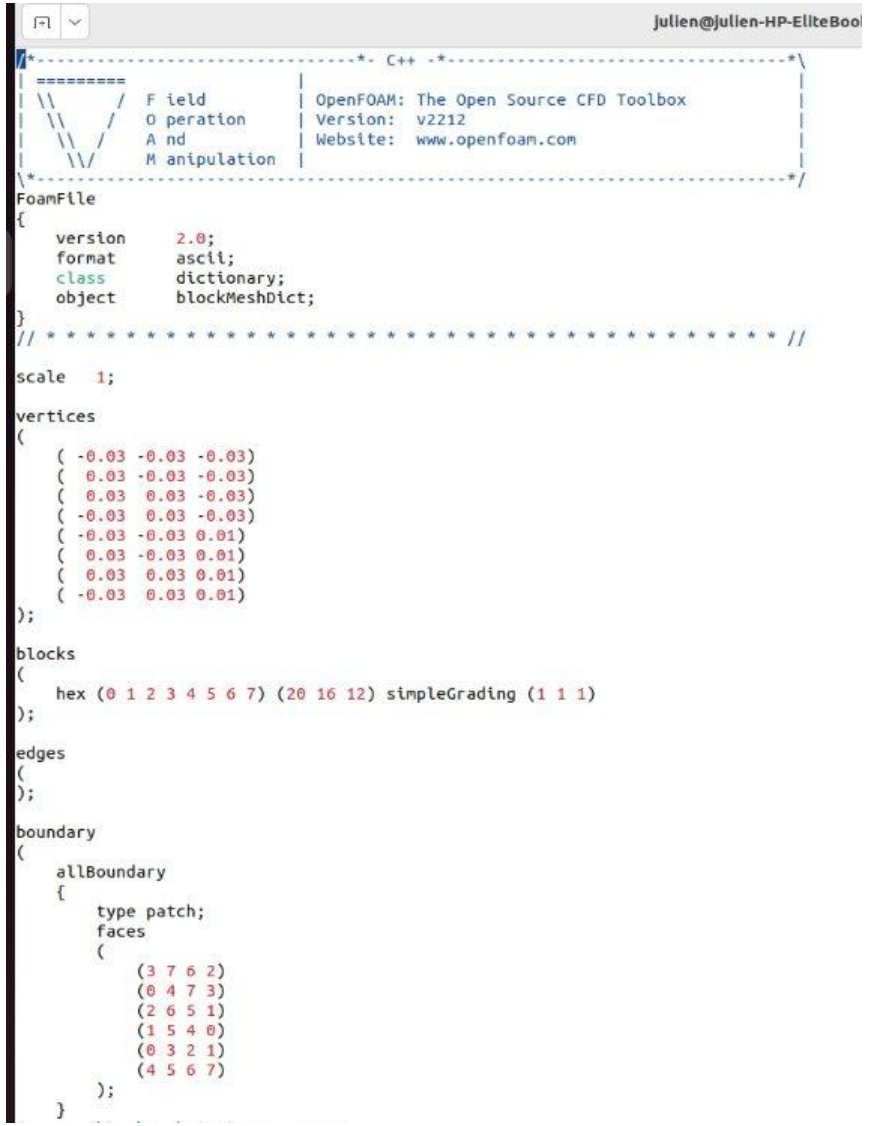
- Copy the content of the flange meshing tutorial inside your folder

```
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerMesh$ cp -r /usr/lib/openfoam/openfoam2212/tutorials/mesh/snappyHexMesh/flange/* .
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerMesh$ ls
Allclean  Allrun  Bentheimer200-5mum.stl  Bentheimer400-5mum.raw  system
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerMesh$
```

Standard approach (10/31)

- Open the system/blockMeshDict file. It looks like this:

- We're going to replace with the dimension of the image



Standard approach (11/31)

- We keep 1 pixel on the boundaries to avoid misalignment
- We start with a 50x50x50 mesh (so resolution is 4x5mum=20mum

julien@julien-HP-EliteBo

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2212                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

scale   1;

vertices
(
    ( 1 1 1)
    ( 199 1 1)
    ( 199 199 1)
    ( 1 199 1)
    ( 1 1 199)
    ( 199 1 199)
    ( 199 199 199)
    ( 1 199 199)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (50 50 50) simpleGrading (1 1 1)
);
```

Standard approach (12/31)

- We're going to inject at z=0 and produce at z=200, so we need to change the boundaries:

```
boundary
(
    walls
    {
        type patch;
        faces
        (
            (3 7 6 2)
            (0 4 7 3)
            (2 6 5 1)
            (1 5 4 0)
        );
    }

    inlet
    {
       type patch;
       faces
       (
            (0 3 2 1)
       );
    }

    outlet
    {
       type patch;
       faces
       (
            (4 5 6 7)
        );
    }
);
```

Standard approach (13/31)

- Next we need to change the system/snappyHexMeshDict file.
- First change the geometry

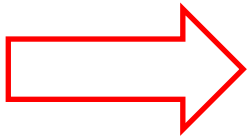
```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      snappyHexMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

// Which of the steps to run
castellatedMesh true;
snap            true;
addLayers       false;


// Geometry. Definition of all surfaces. All surfaces are of class
// searchableSurface.
// Surfaces are used
// - to specify refinement for any mesh cell intersecting it
// - to specify refinement for any mesh cell inside/outside/near
// - to 'snap' the mesh boundary to the surface
geometry
{
    flange.stl
    {
        type triSurfaceMesh;
        name flange;
    }

    //- Refine a bit extra around the small centre hole
    refineHole
    {
        type    sphere;
        origin  (0 0 -0.012);
        radius  0.003;
    }
}
```

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      snappyHexMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

// Which of the steps to run
castellatedMesh true;
snap            true;
addLayers       false;


// Geometry. Definition of all surfaces. All surfaces are of class
// searchableSurface.
// Surfaces are used
// - to specify refinement for any mesh cell intersecting it
// - to specify refinement for any mesh cell inside/outside/near
// - to 'snap' the mesh boundary to the surface
geometry
{
    Bentheimer200-5mum.stl
    {
        type triSurfaceMesh;
        name poreWalls;
    }

    //- Refine a bit extra around the small centre hole
    //refineHole
    //{
    //    type    sphere;
    //    origin  (0 0 -0.012);
    //    radius  0.003;
    //}
}
```

Standard approach (14/31)

- Refine around surface

```
// Explicit feature edge refinement
// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

// Specifies a level for any cell intersected by its edges.
// This is a featureEdgeMesh, read from constant/triSurface for now.
features
(
    {
        file "flange.extendedFeatureEdgeMesh";
        level 0;
    }
);



// Surface based refinement
// ~~~~~~~~~~~~~~~~~~~~~~~~

// Specifies two levels for every surface. The first is the minimum level,
// every cell intersecting a surface gets refined up to the minimum level.
// The second level is the maximum level. Cells that 'see' multiple
// intersections where the intersections make an
// angle > resolveFeatureAngle get refined up to the maximum level.

refinementSurfaces
{
    flange
    {
        // Surface-wise min and max refinement level
        level (2 2);
    }
}
```



```
// Explicit feature edge refinement
// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

// Specifies a level for any cell intersected by its edges.
// This is a featureEdgeMesh, read from constant/triSurface for now.
features
(
    //{
    //    file "flange.extendedFeatureEdgeMesh";
    //    level 0;
    //}
);



// Surface based refinement
// ~~~~~~~~~~~~~~~~~~~~~~~~

// Specifies two levels for every surface. The first is the minimum level,
// every cell intersecting a surface gets refined up to the minimum level.
// The second level is the maximum level. Cells that 'see' multiple
// intersections where the intersections make an
// angle > resolveFeatureAngle get refined up to the maximum level.

refinementSurfaces
{
    poreWalls
    {
        // Surface-wise min and max refinement level
        level (1 1);
    }
}
```

Standard approach (15/31)

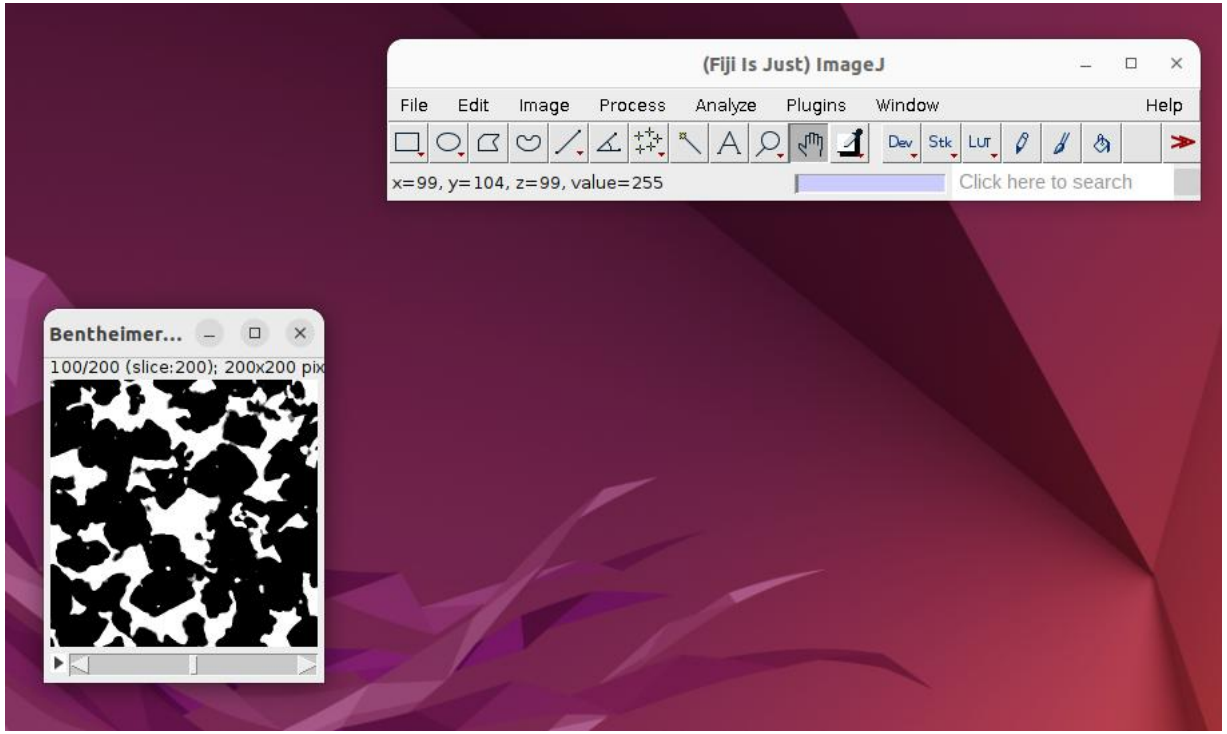
- Remove refine region

```
    refinementRegions
    {
        //refineHole
        //{
        //    mode inside;
        //    levels ((1E15 3));
        //}
    }


    // Mesh selection
    // ~~~~~~~~~~~~~~

    // After refinement patches get added for all refinementSurfaces and
    // all cells intersecting the surfaces get put into these patches. The
    // section reachable from the locationInMesh is kept.
    // NOTE: This point should never be on a face, always inside a cell, even
    // after refinement.
    // This is an outside point locationInMesh (-0.033 -0.033 0.0033);
    locationInMesh (-9.23149e-05 -0.0025 -0.0025); // Inside point

    // Whether any faceZones (as specified in the refinementSurfaces)
    // are only on the boundary of corresponding cellZones or also allow
    // free-standing zone faces. Not used if there are no faceZones.
    allowFreeStandingZoneFaces true;
}
```

Standard approach (16/31)

- For location in mesh, we need a point at the center of a pore
- Use imageJ for that
- Example (99,104,99)

```
// Mesh selection
// ~~~~~~~~~~~~~~

// After refinement patches get added for all refinementSurfaces and
// all cells intersecting the surfaces get put into these patches. The
// section reachable from the locationInMesh is kept.
// NOTE: This point should never be on a face, always inside a cell, even
// after refinement.
// This is an outside point locationInMesh (-0.033 -0.033 0.0033);
locationInMesh (99 104 99); // Inside point
```



Standard approach (17/31)

- Remove the layer

```
// Settings for the layer addition.
addLayersControls
{
    // Are the thickness parameters below relative to the undistorted
    // size of the refined cell outside layer (true) or absolute sizes (fa
    relativeSizes true;

    // Per final patch (so not geometry!) the layer information
    layers
    {
        //"flange_.*"
        //{
        //    nSurfaceLayers 1;
        //}
    }

    // Expansion factor for layer mesh
    expansionRatio 1.0;
```

Standard approach (18/31)

- Open the Allrun file

```
#!/bin/sh
cd "${0%/*}" || exit                                # Run from this directory
. ${WM_PROJECT_DIR:?}/bin/tools/RunFunctions        # Tutorial run functions
#------------------------------------------------------------------------------

mkdir -p constant/triSurface

cp -f \
    "$FOAM_TUTORIALS"/resources/geometry/flange.stl.gz \
    constant/triSurface

runApplication blockMesh

runApplication surfaceFeatureExtract

runApplication snappyHexMesh -overwrite

#------------------------------------------------------------------------------
```

Standard approach (19/31)

- Replace by this:

```
#!/bin/sh
cd "${0%/*}" || exit                                # Run from this directory
. ${WM_PROJECT_DIR:?}/bin/tools/RunFunctions        # Tutorial run functions
#------------------------------------------------------------------------------

mkdir -p constant/triSurface

cp -f \
    Bentheimer200-5mum.stl \
    constant/triSurface

runApplication blockMesh

#runApplication surfaceFeatureExtract

runApplication decomposePar

runParallel snappyHexMesh -overwrite

runParallel transformPoints -scale '(5e-6 5e-6 5e-6)'

runParallel checkMesh

runApplication reconstructParMesh -constant

#------------------------------------------------------------------------------
~
```

Copy image

Not used here

Parallel

Scale to resolution (5 microns)

Check mesh quality

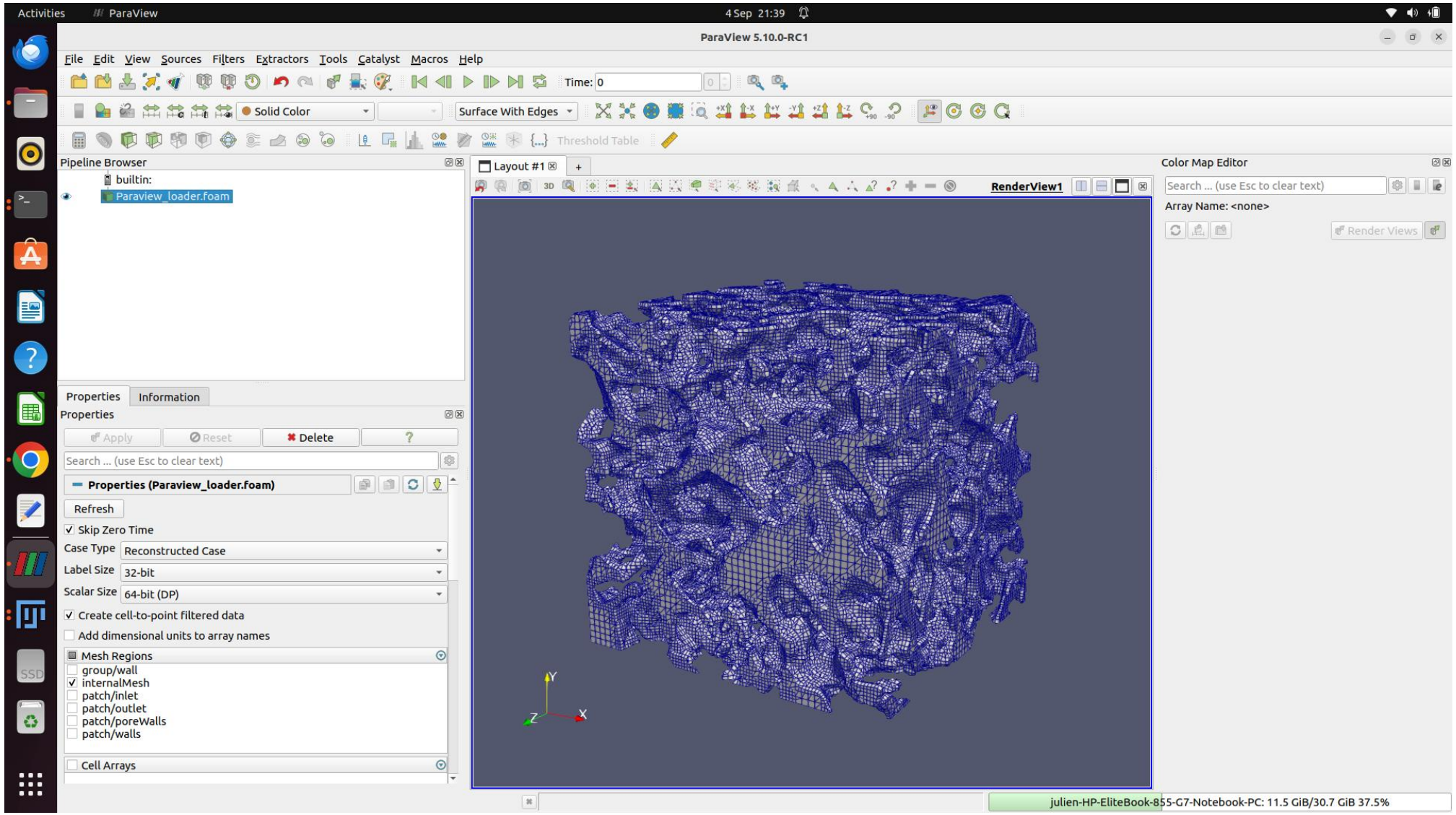
Reconstruct mesh

Standard approach (20/31)

- Open system/decomposeParDict
- Use as many processors as you have

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2212                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 8;

method          simple;

coeffs
{
    n           (2 2 2);
}


// ************************************************************************* //
```

Standard approach (21/31)



Standard approach (22/31)

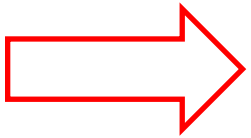
- Create a new folder called BentheimerFlow
- Copy the content of pitzDaily simpleFoam tutorial in

```
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerFlow$ cp -r /usr/lib/openfoam/openfoam2212/tutorials/incompressible/simpleFoam/pitzDaily/*
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerFlow$ ls
0  constant  system
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerFlow$
```

Standard approach (23/31)

- Correct the 0/p boundary conditions
- We inject at constant kinematic pressure 1 m^2/s^2
- Don’t forget the poreWalls!

```
boundaryField
{
    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    upperWall
    {
        type            zeroGradient;
    }

    lowerWall
    {
        type            zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }
}
```

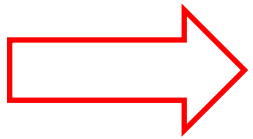


```
boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 1;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    walls
    {
        type            zeroGradient;
    }

    poreWalls
    {
        type            zeroGradient;
    }
}
```

Standard approach (24/31)

- And 0/U

```
boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (10 0 0);
    }

    outlet
    {
        type            zeroGradient;
    }

    upperWall
    {
        type            noSlip;
    }

    lowerWall
    {
        type            noSlip;
    }

    frontAndBack
    {
        type            empty;
    }
}
```



```
boundaryField
{
    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    walls
    {
        type            noSlip;
    }

    poreWalls
    {
        type            noSlip;
    }
}


// ************************************************
~
~
~
~
```

Standard approach (25/31)

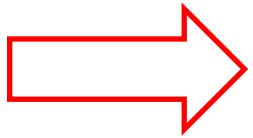
- Remove turbulence variables in 0

```
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerFlow$ rm -f 0/epsilon 0/k 0/nu* 0/omega
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerFlow$ ls 0
p  U
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerFlow$
```

Standard approach (26/31)

- Change constant/turbulenceProperties to laminar

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

simulationType      RAS;

RAS
{
    // Tested with kEpsilon, realizableKE, kOmega, kOmegaSST,
    // ShihQuadraticKE, LienCubicKE.
    RASModel        kEpsilon;

    turbulence      on;

    printCoeffs     on;
}
```



```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

simulationType      laminar;

//RAS
//{
    // Tested with kEpsilon, realizableKE, kOmega, kOmegaSST,
    // ShihQuadraticKE, LienCubicKE.
//    RASModel        kEpsilon;

//    turbulence      on;

//    printCoeffs     on;
//}
```

Standard approach (27/31)

- Copy the mesh from BentheimerMesh in constant

```
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerFlow$ cp -r ../BentheimerMesh/constant/polyMesh/ constant/.
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerFlow$ ls constant/
polyMesh  transportProperties  turbulenceProperties
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerFlow$
```

- Copy the decomposeParDict file from BentheimerMesh in system

```
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerFlow$ cp ../BentheimerMesh/system/decomposeParDict system/.
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerFlow$ ls system/
blockMeshDict  controlDict  decomposeParDict  fvSchemes  fvSolution  streamlines
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerFlow$
```

Standard approach (28/31)

- Remove streamlines from system/controlDict

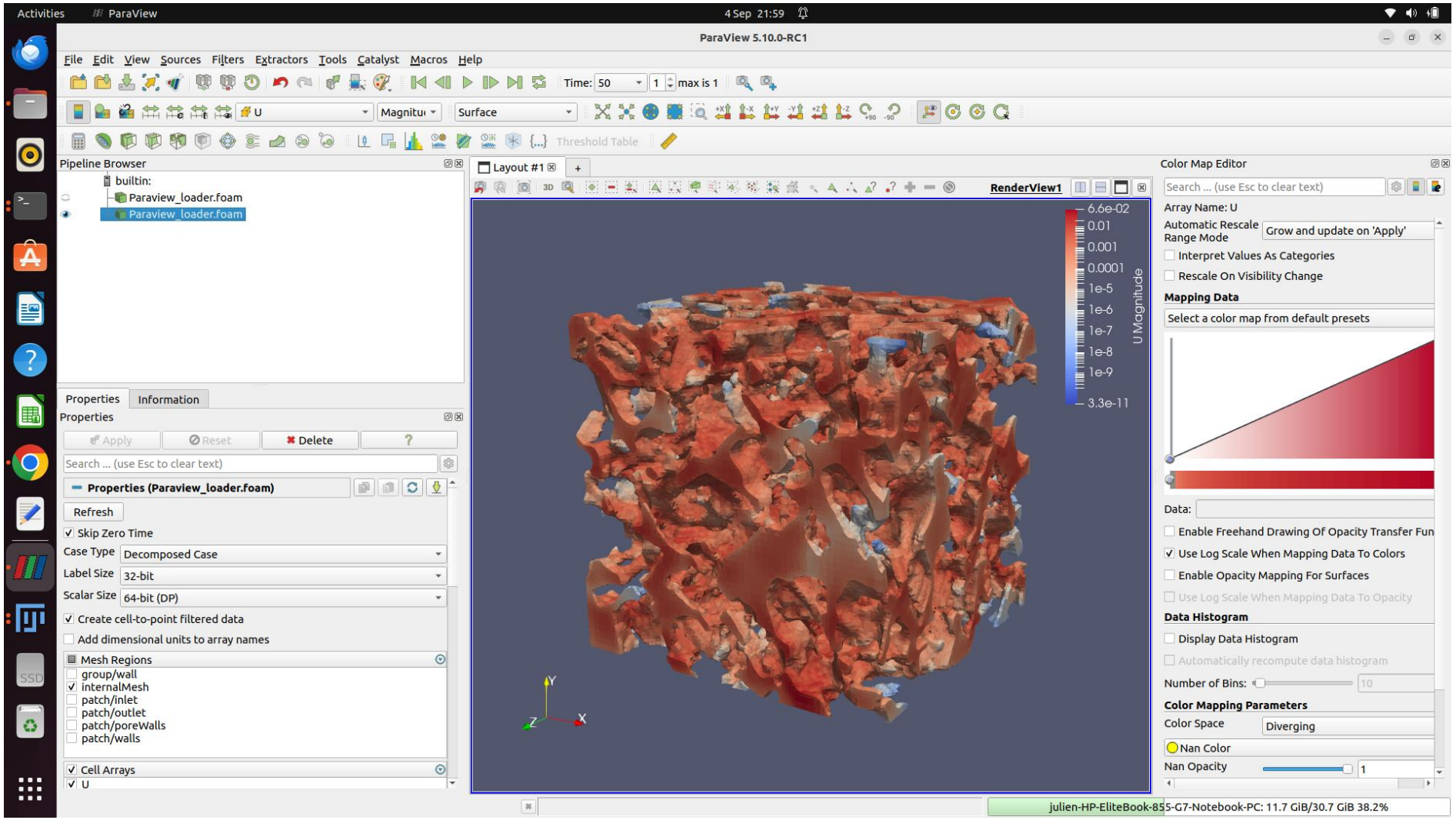
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2212                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     simpleFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         2000;

deltaT          1;

writeControl    timeStep;

writeInterval   100;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

//functions
//{
//    #includeFunc streamlines
//}
```

Standard approach (29/31)

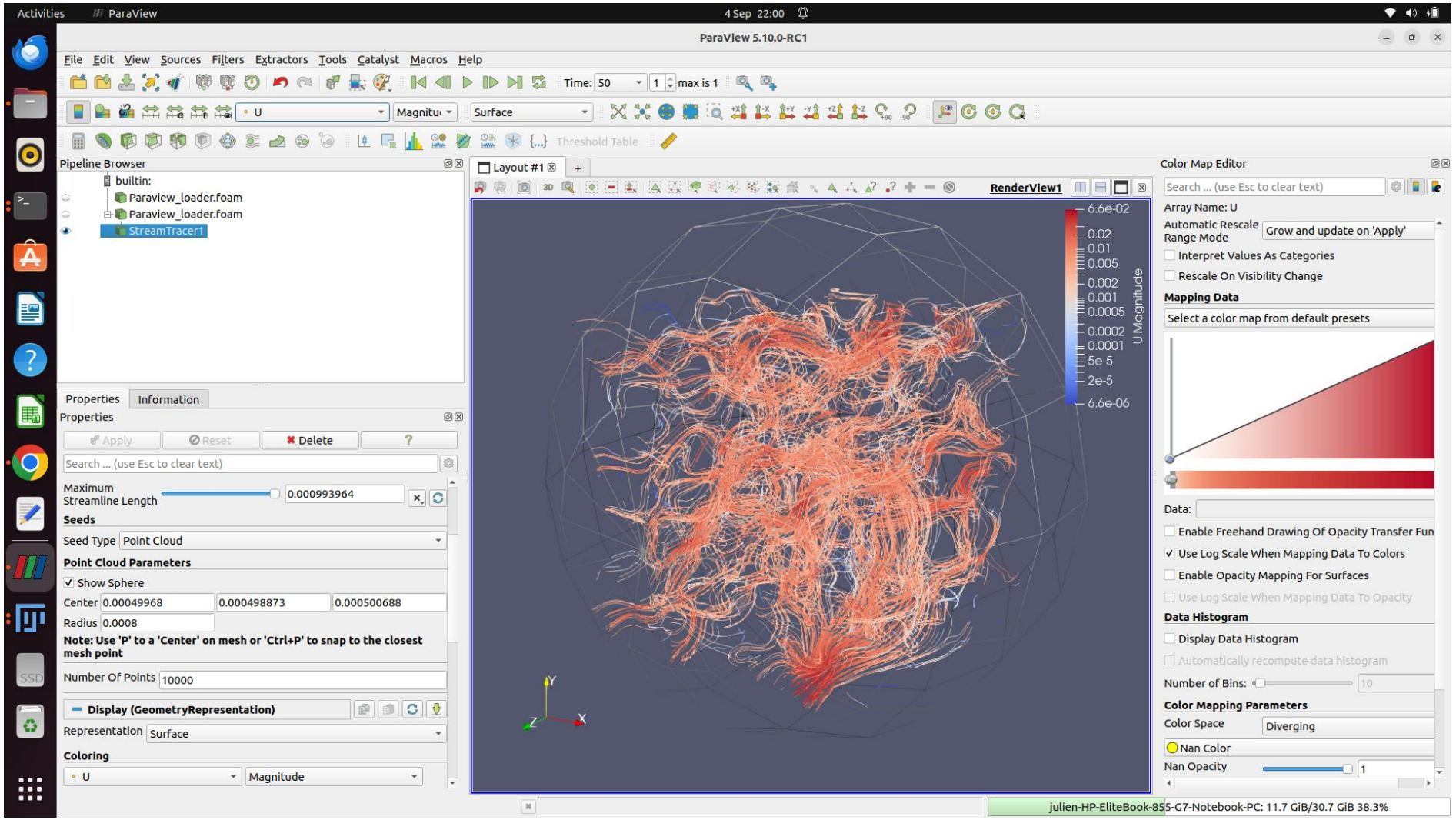
- decomposePar
- mpiexec –np 8 simpleFoam -parallel

```
ExecutionTime = 11.78 s  ClockTime = 12 s

Time = 46

smoothSolver:  Solving for Ux, Initial residual = 0.00185949, Final residual = 0.000155903, No Iterations 1
smoothSolver:  Solving for Uy, Initial residual = 0.00175707, Final residual = 0.000150139, No Iterations 1
smoothSolver:  Solving for Uz, Initial residual = 0.000810478, Final residual = 7.88186e-05, No Iterations 1
GAMG:  Solving for p, Initial residual = 0.000273325, Final residual = 2.35131e-05, No Iterations 1
time step continuity errors : sum local = 0.131317, global = 0.0954474, cumulative = 9.00385
ExecutionTime = 12.06 s  ClockTime = 12 s

Time = 47

smoothSolver:  Solving for Ux, Initial residual = 0.00125085, Final residual = 0.000110669, No Iterations 1
smoothSolver:  Solving for Uy, Initial residual = 0.00119028, Final residual = 0.000107948, No Iterations 1
smoothSolver:  Solving for Uz, Initial residual = 0.000605293, Final residual = 5.78944e-05, No Iterations 1
GAMG:  Solving for p, Initial residual = 0.00024159, Final residual = 1.48821e-05, No Iterations 2
time step continuity errors : sum local = 0.0831548, global = 0.0121817, cumulative = 9.01604
ExecutionTime = 12.35 s  ClockTime = 12 s

Time = 48

smoothSolver:  Solving for Ux, Initial residual = 0.00148284, Final residual = 0.000124325, No Iterations 1
smoothSolver:  Solving for Uy, Initial residual = 0.00140176, Final residual = 0.000119687, No Iterations 1
smoothSolver:  Solving for Uz, Initial residual = 0.000642854, Final residual = 6.2641e-05, No Iterations 1
GAMG:  Solving for p, Initial residual = 0.000218376, Final residual = 1.86615e-05, No Iterations 1
time step continuity errors : sum local = 0.104233, global = 0.0750497, cumulative = 9.09109
ExecutionTime = 12.62 s  ClockTime = 12 s

Time = 49

smoothSolver:  Solving for Ux, Initial residual = 0.00100345, Final residual = 8.86615e-05, No Iterations 1
smoothSolver:  Solving for Uy, Initial residual = 0.000954835, Final residual = 8.63789e-05, No Iterations 1
smoothSolver:  Solving for Uz, Initial residual = 0.000480583, Final residual = 4.61173e-05, No Iterations 1
GAMG:  Solving for p, Initial residual = 0.000193113, Final residual = 1.91613e-05, No Iterations 1
time step continuity errors : sum local = 0.107066, global = 0.0848023, cumulative = 9.17589
ExecutionTime = 12.9 s  ClockTime = 13 s

Time = 50

smoothSolver:  Solving for Ux, Initial residual = 0.000837304, Final residual = 7.24012e-05, No Iterations 1
smoothSolver:  Solving for Uy, Initial residual = 0.000795861, Final residual = 7.04525e-05, No Iterations 1
smoothSolver:  Solving for Uz, Initial residual = 0.000394767, Final residual = 3.70644e-05, No Iterations 1
GAMG:  Solving for p, Initial residual = 0.000171715, Final residual = 1.15312e-05, No Iterations 2
time step continuity errors : sum local = 0.0644399, global = 0.0125715, cumulative = 9.18846
ExecutionTime = 13.19 s  ClockTime = 13 s


SIMPLE solution converged in 50 iterations

End
```

Standard approach (30/31)



Standard approach (31/31)



About GeoChemFoam (standard method)

- GCFoam includes script that modify the relevant files for you.
- Check tutorial 1: https://github.com/GeoChemFoam/GeoChemFoam/wiki/Test-Case-01----Species-transport
- Image properties can be enteredin the createMesh script: no need to go directly change blockMeshDict, snappyHexMeshDict, 0/p,…

```
###### USERS INPUT ######################################################

#Define image name
Image_name="Ketton"

#Image directory location ($PWD if current directory)
dir="$GCFOAM_IMG/raw"

#Choose image format
format='raw'
#format='stl'

#Choose if the image is compressed or not
compressed='yes'

#if format='raw', Define image dimensions
x_dim=512
y_dim=512
z_dim=512

#if format='raw', define value of the pores and solid in the image
pores_value=255
solid_value=0

#if format= 'stl' enter the location of a pore
#pore_index_X=0
#pore_index_Y=0
#pore_index_Z=0

# define resolution (m)
res=0.0000053

# Define cropping parameters
x_min=0
x_max=300
y_min=0
y_max=300
z_min=0
z_max=300

# number of cells of initial mesh
# n*(2^nlevel) should be equal to image dimension when not binning
n_x=75
n_y=75
n_z=75

# Level of refinement - mesh is refined by n_level at the pore surfaces
n_level=1

# flow direction (0 is x, 1 is y and 2 is z)
direction=0
```

Volume-Of-Solid approach (1/12)

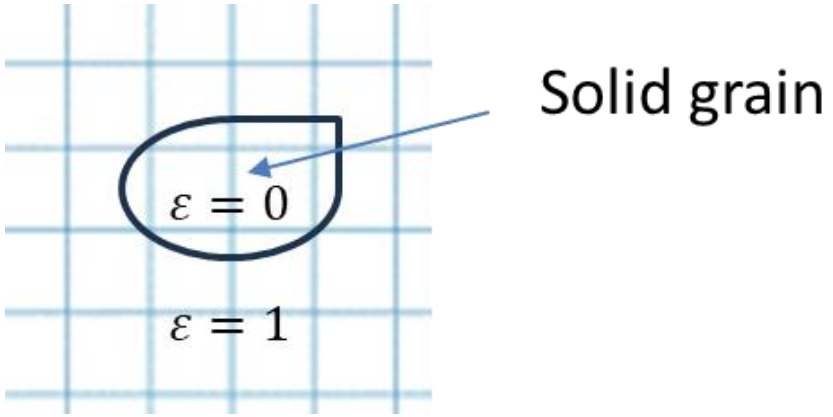
- Volume-of-Solid approach: solid surface represented by indicator function ε
- The flow is then solved using an immersed boundary condition
- E.g., Darcy-Brinkman-Stokes

$$0 = -\nabla p + \nabla.\left(\frac{\eta}{\varepsilon}\nabla \boldsymbol{u}\right) + \eta K^{-1}\boldsymbol{u}$$

- K^{-1}calculated for each voxel using Kozeny-Carman relationship

$$K^{-1} = A\frac{(1-\varepsilon)^2}{(\varepsilon + SMALL)^3}$$

- A calculated so that K^{-1} very large (e.g. 10^{21} when $\varepsilon = 0$
- We will implement this in a new solver simpleDBSFoam



Volume-Of-Solid approach (2/12)

- Copy BentheimerMesh folder into a new BentheimerVOS folder and clean it:

```
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run$ cp -r BentheimerMesh/ BentheimerVOS
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run$ cd BentheimerVOS/
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerVOS$ ./Allclean
Cleaning case /home/julien/OpenFOAM/julien-v2212/run/BentheimerVOS
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerVOS$ ls
Allclean  Allrun  Bentheimer200-5mum.stl  Bentheimer400-5mum.raw  system
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerVOS$
```

Bentheimer/system and Bentheimer/constant into our folder:

```
julien@julien-HP-EliteBook-855-G7-Notebook-PC: ~/OpenFOAM/julien-v2212/run/BentheimerVOS
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerVOS$ cp -r ../BentheimerFlow/0 .
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerVOS$ cp ../BentheimerFlow/constant/* constant/.
cp: -r not specified; omitting directory '../BentheimerFlow/constant/polyMesh'
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerVOS$ cp ../BentheimerFlow/system/* system/.
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerVOS$
```

Volume-Of-Solid approach (3/12)

- Modify the blockMeshDict file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2212                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

scale   5e-6;

vertices
(
    ( 1 1 1)
    ( 199 1 1)
    ( 199 199 1)
    ( 1 199 1)
    ( 1 1 199)
    ( 199 1 199)
    ( 199 199 199)
    ( 1 199 199)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (99 99 99) simpleGrading (1 1 1)
);
```

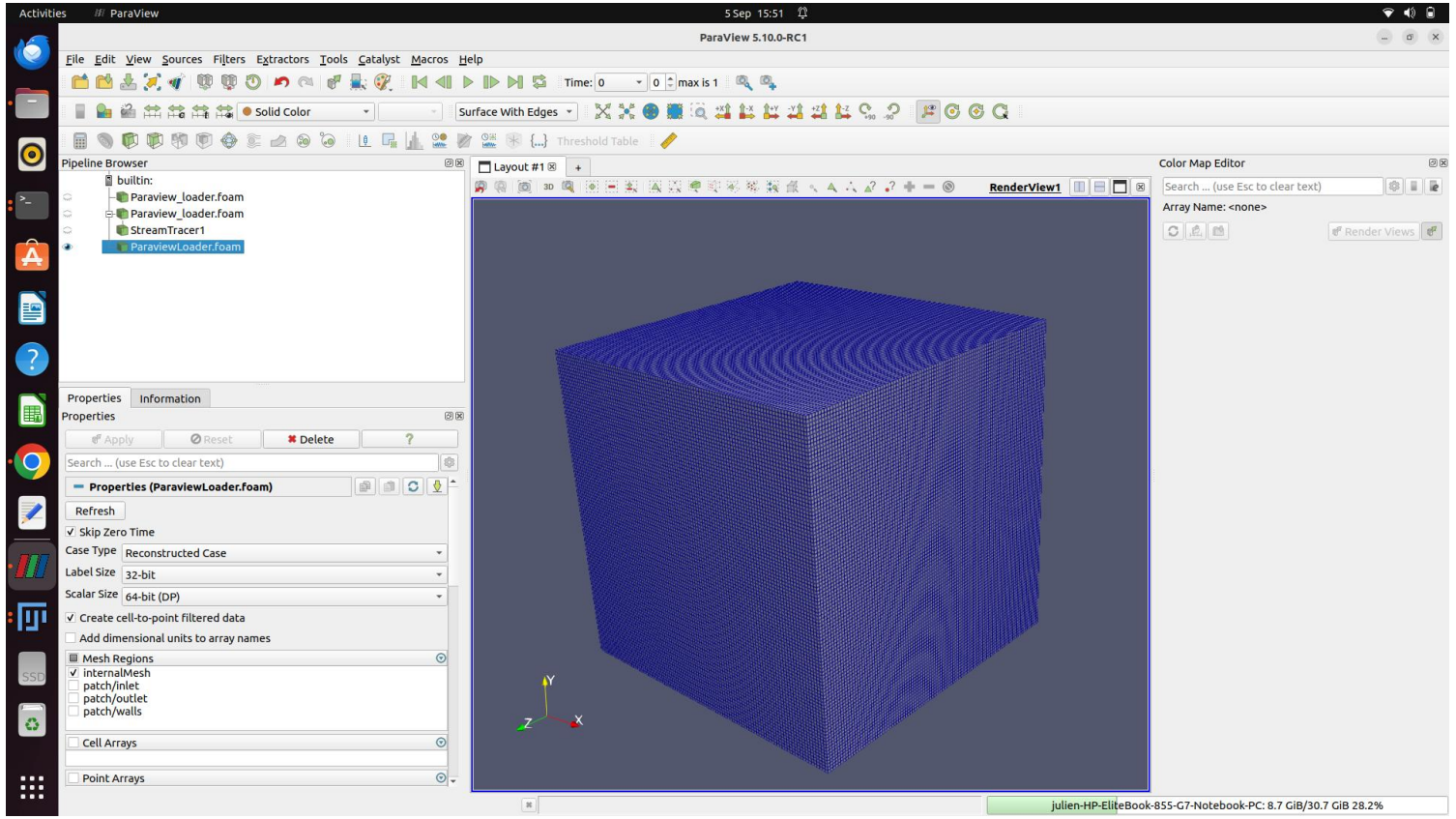
The resolution is now implemented here

Here it is important that the number of cells divide the number of voxels (199-1)/99=198/99=2

We are not using a mesh with levels (yet). It is a regular cartesian blockMesh

Volume-Of-Solid approach (4/12)

- Run blockMesh
- We have now a 99x99x99 regular grid



Volume-Of-Solid approach (5/12)

- Download the createEps.py script from the https://github.com/GeoChemFoam/GeoChemFoam/tree/main/Examples/FoamIberiaTraining repository and copy it in the folder

- You must modify the line to calculate what is the value of esp in the [i,j,k] grid block (1 if array=pores_value, 0 if array=solid_values)

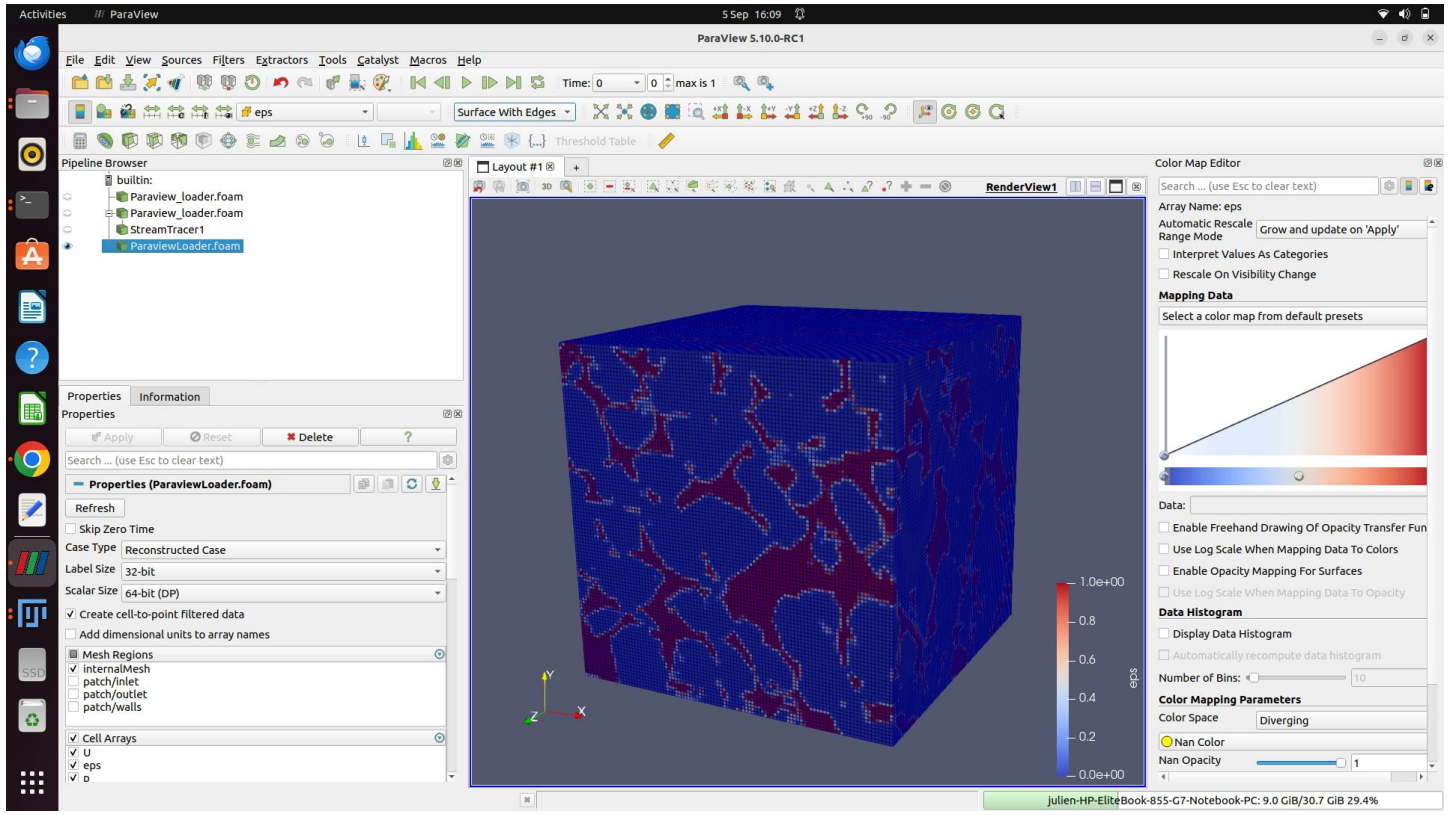
```
import os
import sys
import numpy as np


Image_name=sys.argv[1]
xDim = int(sys.argv[2])
yDim = int(sys.argv[3])
zDim = int(sys.argv[4])
xMin = int(sys.argv[5])
xMax = int(sys.argv[6])
yMin = int(sys.argv[7])
yMax = int(sys.argv[8])
zMin = int(sys.argv[9])
zMax = int(sys.argv[10])
nX   = int(sys.argv[11])
nY   = int(sys.argv[12])
nZ   = int(sys.argv[13])
pores_value = float(sys.argv[14])
solid_value = float(sys.argv[15])

# open image
f = open(Image_name+'.raw', 'rb')
img = np.fromfile(f, dtype=np.uint8)
img = np.reshape(img,(zDim,yDim,xDim))
my_array = img[zMin:zMax,yMin:yMax,xMin:xMax]


#number of cells
p=int((xMax-xMin)/nX)
q=int((yMax-yMin)/nY)
r=int((zMax-zMin)/nZ)

ncells = nX*nY*nZ

eps=np.zeros((nX,nY,nZ),dtype=float)
for i in range (0,nX):
    for ii in range (0,p):
        for j in range (0,nY):
            for jj in range (0,q):
                for k in range (0,nY):
                    for kk in range (0,r):
                        eps[i,j,k]+=//*FILL VALUE OF EPS HERE*//

data = [
'/*---------------------------------*- C++ -*----------------------------------*\\\\n',
'| =========                 |                                                 |\\n',
'| \\\\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |\\n',
'|  \\\\    /   O peration     | Version:  2212                                  |\\n',
'|   \\\\  /    A nd           | Website:  www.openfoam.com                      |\\n',
'|    \\\\/     M anipulation  |                                                 |\\n',
'\\*---------------------------------------------------------------------------*/\\n',
"FoamFile\n"
```

Volume-Of-Solid approach (6/12)

- Run the python script with the correct arguments

```
julien@julien-HP-EliteBook-855-G7-Notebook-PC: ~/OpenFOAM/julien-v2212/run/BentheimerVOS
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212/run/BentheimerVOS$ python createEps.py 'Bentheimer400-5mum' 400 400 400 101 299 101 299 101 299 99 99 99 0 255
```

- We have now a Volume-Of-Solid indicator function eps



Volume-Of-Solid approach (7/12)

- Copy the simpleFoam folder into your OpenFOAM folder:

julien@julien-HP-EliteBook-855-G7-Notebook-PC: ~/OpenFOAM/julien-v2212

```
(mypython_environment) julien@julien-HP-EliteBook-855-G7-Notebook-PC:~/OpenFOAM/julien-v2212$ cp -r /usr/lib/openfoam/openfoam2212/applications/solvers/incompressible/simpleFoam/ .
```

- Rename the folder simpleDBSFoam, and the simpleFoam.C file simpleDBSFoam.C

Volume-Of-Solid approach (8/12)

- Modify the Make/file:

```
simpleDBSFoam.C

EXE = $(FOAM_USER_APPBIN)/simpleDBSFoam
~
```

- Compile. We have now a copy of simpleFoam called simpleDBSFoam

Volume-Of-Solid approach (9/12)

- Modify the createFields.H file to read the eps field
- We also need to create the volScalarField nu

```
label pRefCell = 0;
scalar pRefValue = 0.0;
setRefCell(p, simple.dict(), pRefCell, pRefValue);
mesh.setFluxRequired(p.name());


singlePhaseTransportModel laminarTransport(U, phi);

volScalarField nu
(
    laminarTransport.nu()
);
```

```
Info<< "Reading field p\n" << endl;
volScalarField p
(
    IOobject
    (
        "p",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

Info<< "Reading field eps\n" << endl;
volScalarField eps
(
    IOobject
    (
        "eps",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

Info<< "Reading field U\n" << endl;
volVectorField U
(
    IOobject
    (
        "U",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

Volume-Of-Solid approach (10/12)

- ko (permeability of the solid field) will be read from constant/transportProperties, and the inverse of permeability field will be created using Kozeny-Carman

```
singlePhaseTransportModel laminarTransport(U, phi);

volScalarField nu
(
    laminarTransport.nu()
);


dimensionedScalar k0(dimensionedScalar::getOrDefault("k0",laminarTransport,sqr(dimLength),Zero));

volScalarField Kinv = 1/k0*pow(SMALL/(eps+SMALL),3)*pow(1-eps,2);
```

- And in constant/transportProperties:

```
transportModel  Newtonian;

nu              1e-05;

k0 k0 [0 2 0 0 0 0 0] 1e-21;

// ************************************************************************
```

Volume-Of-Solid approach (11/12)

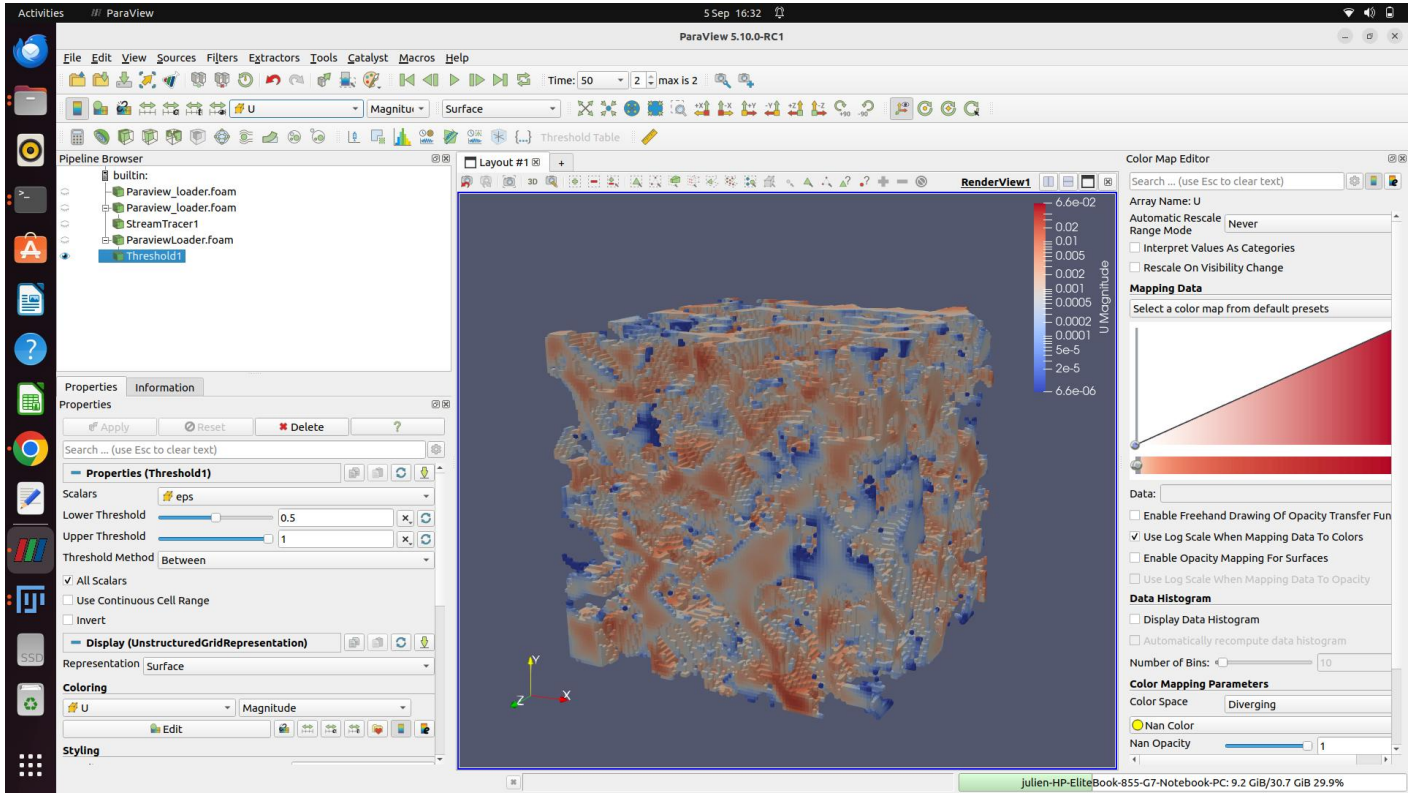
- Modify the UEqn.H file:

```
MRF.correctBoundaryVelocity(U);

tmp<fvVectorMatrix> tUEqn
(
    fvm::div(phi, U)
  + MRF.DDt(U)
  //+ turbulence->divDevReff(U)
  - fvm::laplacian(fvc::interpolate(nu)/fvc::interpolate(eps+SMALL),U)
  + fvm::Sp(nu*Kinv,U)
 ==
    fvOptions(U)
);
fvVectorMatrix& UEqn = tUEqn.ref();

UEqn.relax();

fvOptions.constrain(UEqn);

if (simple.momentumPredictor())
{
    solve(UEqn == -fvc::grad(p));

    fvOptions.correct(U);
}
```

- Compile your code

Volume-Of-Solid approach (12/12)

- decomposePar
- mpiexec –np 8 simpleDBSFoam -parallel



Here we have used threshold to remove the solid cells

About GeoChemFoam (VOS method)

- GCFoam includes script that modify the relevant files for you.
- Check tutorial 5: [Test Case 05 Bentheimer heat transfer · GeoChemFoam/GeoChemFoam Wiki]
- Image properties can be enteredin the createMesh script: no need to go directly change blockMeshDict, topoSetDict, 0/p,…

```
###### USERS INPUT ########################################################

#Define image name
Image_name="Bentheimer400-5mum"
#Image_name="HM12_2_4"
#Image_name="4StraightFractures"
#Image_name="Est_3phase500cubed4micron"

#Image directory location ($PWD if current directory)
dir="$GCFOAM_IMG/raw"

#Choose image format
format='raw'

#Choose if the image is compressed or not
compressed='yes'

# Define image dimensions
x_dim=400
y_dim=400
z_dim=400

segmentation='grayscale'
#Values of solid, pore, and minimum porosity value for the solid phase (
pores_value=0
solid_value=255
eps_min=0.0001

#segmentation='phases'
#Values of solid and pore
#pores_value=1
#solid_value=3

#define the labels of the phases
#phases=(1,2,3)

#define the porosity of each phase, note that the porosity of the solid
#micro_por=(1,0.35,0.0001)

# Define cropping parameters
x_min=0
x_max=200
y_min=0
y_max=200
z_min=0
z_max=200

#padding for inlet/outlet
#This adds voxels in the image directly
padWidth=2

# define resolution (m)
```