PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

## Introduction to OpenFOAM® Computational Library and Viscoelastic Fluid Flow Simulation

# P2 - Mesh Generation and Post-Processing

This presentation was adapted from Wagner Galuppo's and Gabriel Wagner's Foam@Iberia 2023 Beginner Course C2
https://github.com/Computational-Rheology/Foam_Iberia_2023/blob/main/Beginner/C2/


**Mohammadreza Aali & J. Miguel Nórbega**

mohammadreza.aali@jku.at & mnobrega@dep.uminho.pt

University of Minho
School of Engineering

Computational Rheology @IPC

iPC INSTITUTE FOR POLYMERS AND COMPOSITES

IPPD Institute of Polymer Processing and Digital Transformation

JOHANNES KEPLER UNIVERSITY LINZ

# Outline

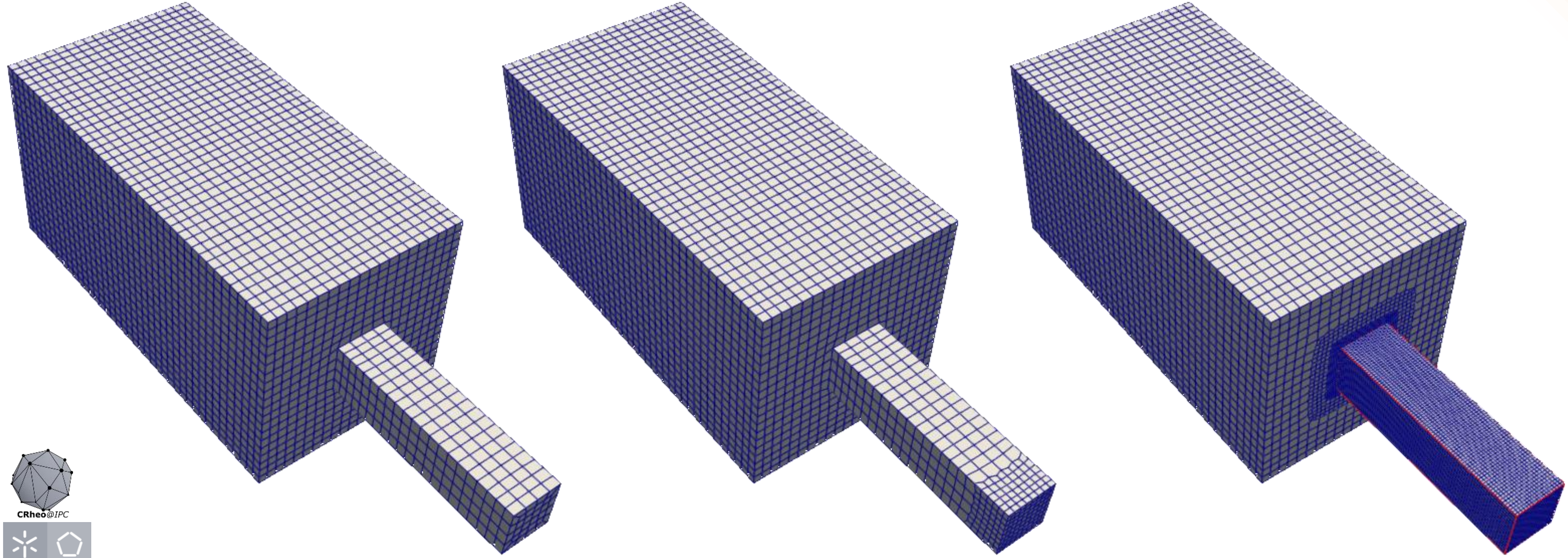
| | |
|---|---|
| 9:00 – 10:30 | **Introduction to OpenFOAM (P1)** |
| 10:30 – 12:00 | **Mesh generation and post-processing (P2)** |
| 12:00 – 13:00 | Lunch break |
| 13:00 – 14:30 | **Case studies: Single- and two-phase flow solvers (P3)** |
| 14:30 – 16:00 | **Case studies: Viscoelastic fluid flow solvers (P4)** |

CRheo@IPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

IPPD Institute of Polymer Processing and Digital Transformation

# Mesh Generation



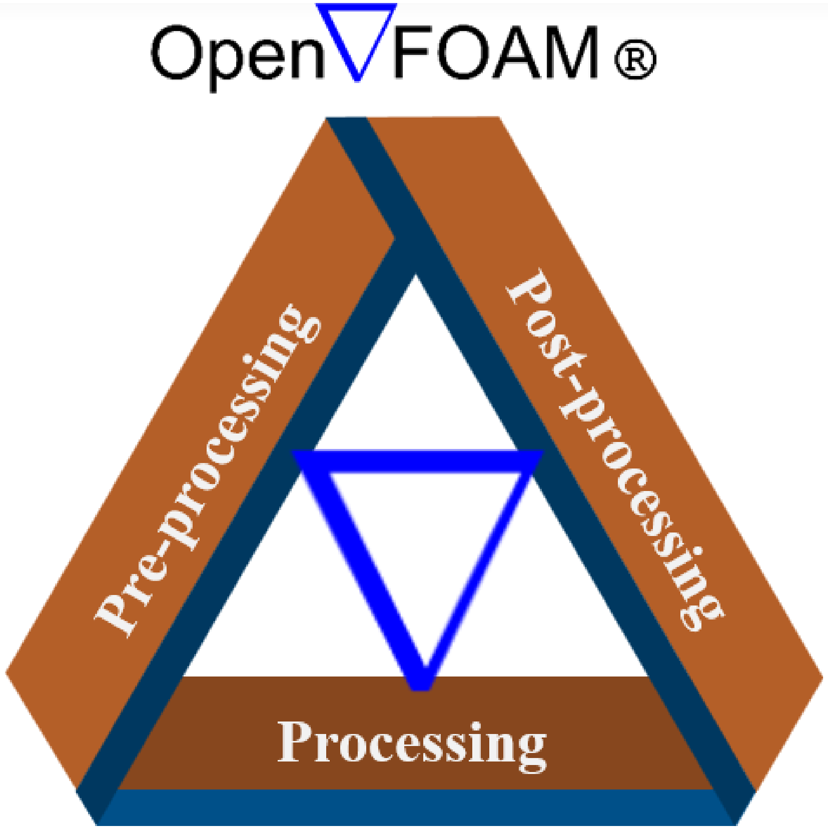
CRheo@IPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

IPPD Institute of
Polymer Processing and
Digital Transformation

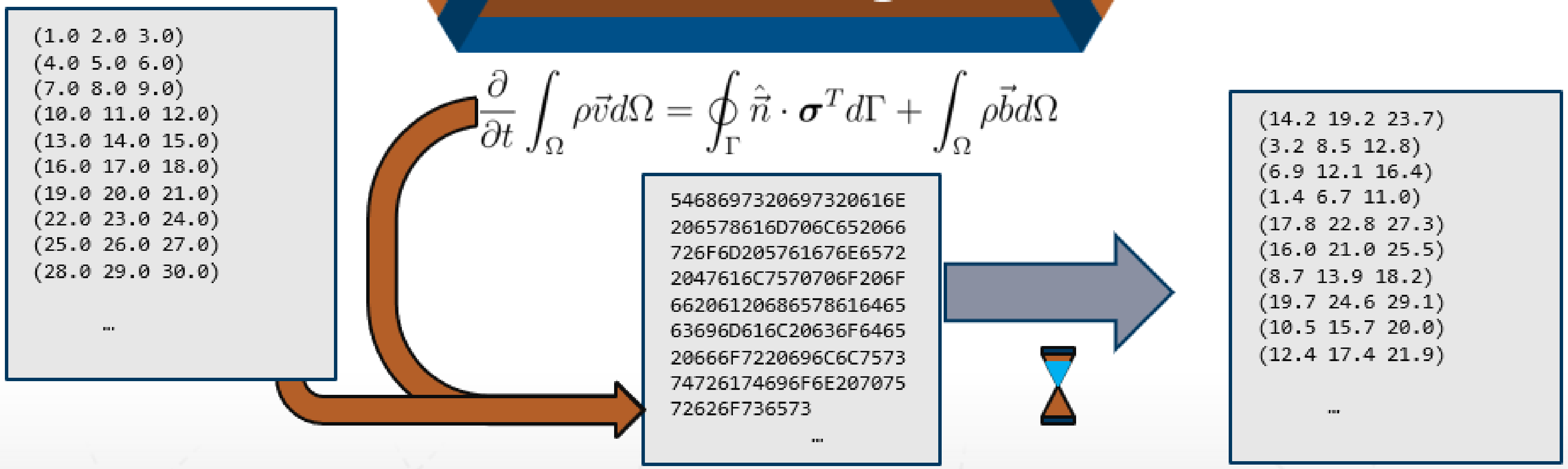
# Introduction

- Geometry Creation
- Meshing Creation
- Meshing Tools
- Material Properties Setup
- Boundaries Definitions
- Convergence Checks
- …




- Data Retrieval and Visualization
- Derived Data Generation
- Trend Analysis
- Improving user's readability of calculated numerical data.
- A wise linkage that leads to enhance productive science for decision support and communication

$$\frac{\partial}{\partial t}\int_{\Omega}\rho\vec{v}d\Omega = \oint_{\Gamma}\hat{\vec{n}}\cdot\boldsymbol{\sigma}^{T}d\Gamma + \int_{\Omega}\rho\vec{b}d\Omega$$



# Introduction

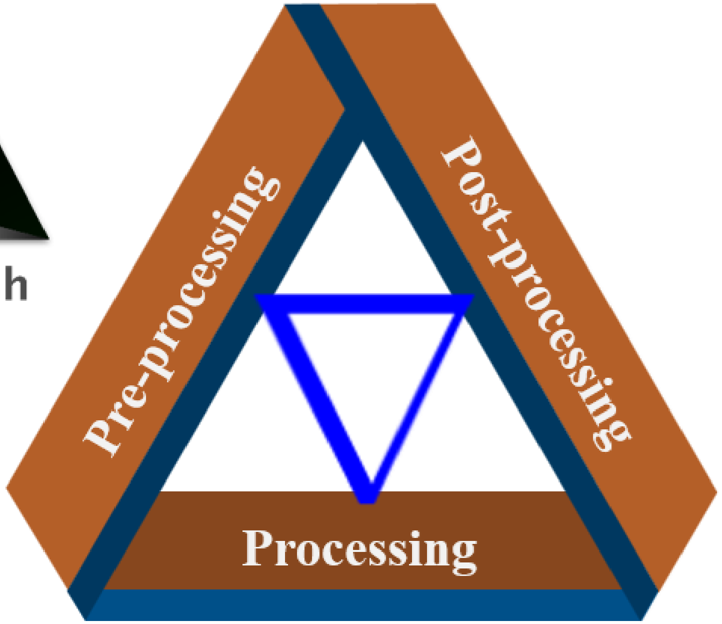
OpenFOAM®

blender®

python™

Gmsh

SALOME




OpenFOAM®

OpenFOAM®

ParaView

tecplot®

python™

blender®

# blockMesh – Case 21 (cavity)

1. Open WSL
2. >> of2206
3. >> run
4. >> cd  case21
5. >> code . ## Open Vscode
6. Visualize blockMeshDict file ##in the System folder

CRheo@IPC

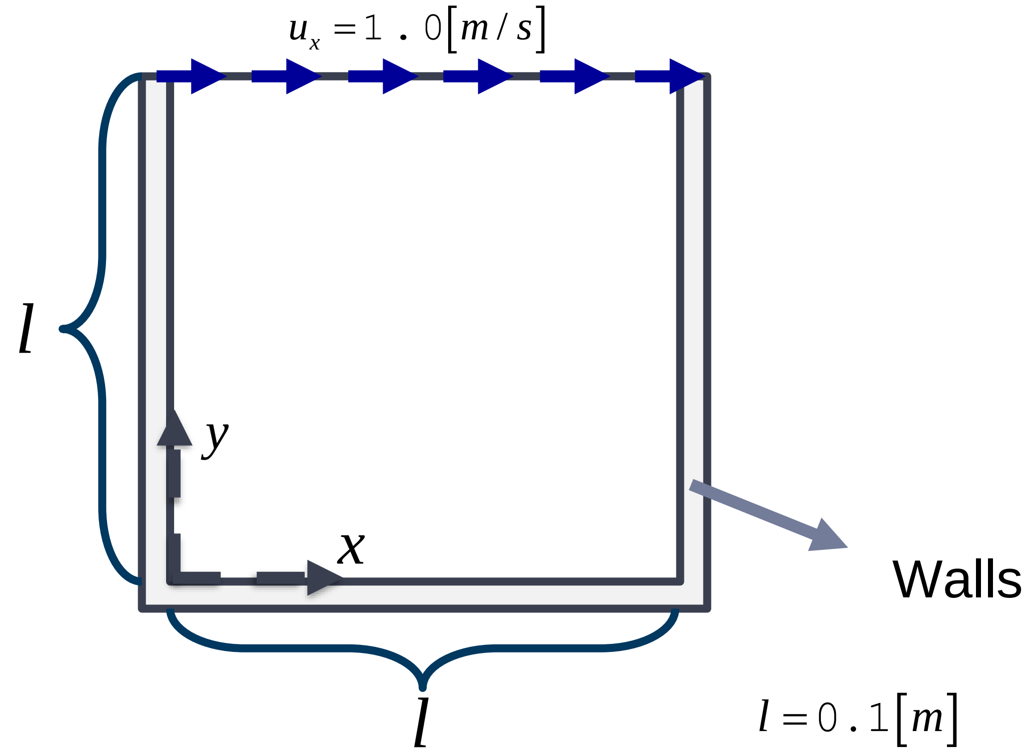
iPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

IPPD Institute of Polymer Processing and Digital Transformation

# blockMesh – Case 21 (cavity)

- Physical Problem: (Cavity tutorial)



# blockMesh – Case 21 (cavity)

BlockMesh Dictionary case21/`system/blockMeshDict`

../C2/tutorials/case01/system/blockMeshDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// convertToMeters 0.1;
scale 0.1;

vertices
(
    (0 0 0)
    (1 0 0)
    (1 1 0)
    (0 1 0)
    (0 0 0.1)
    (1 0 0.1)
    (1 1 0.1)
    (0 1 0.1)
);
```

CRheo@IPC

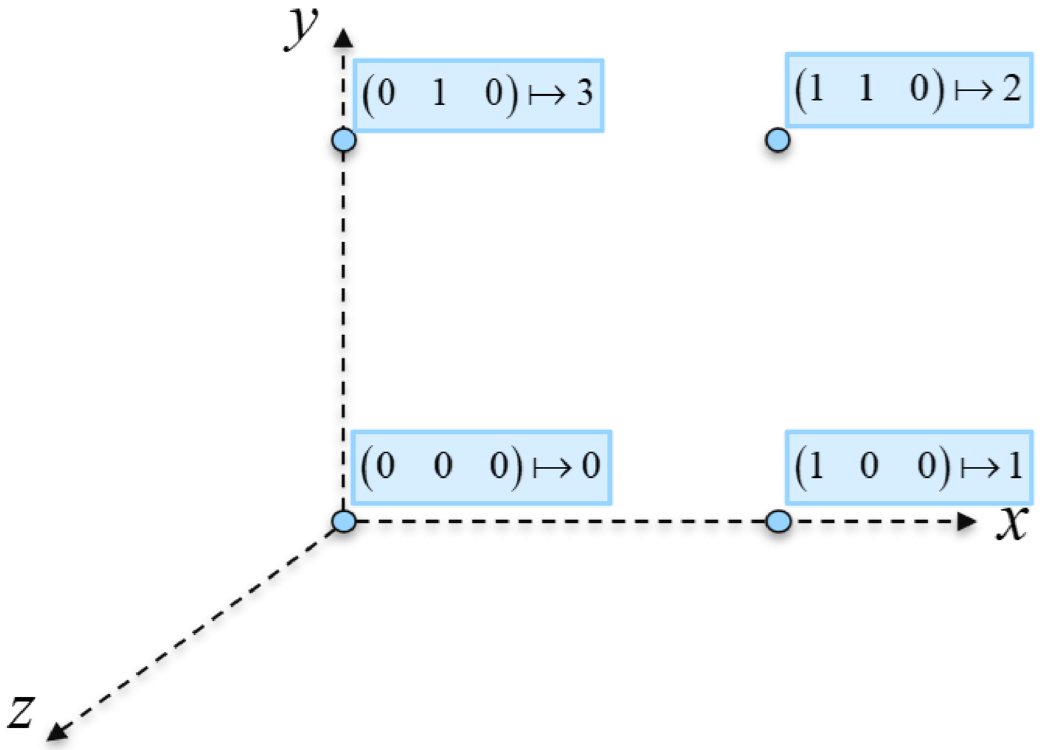
PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

IPPD Institute of Polymer Processing and Digital Transformation

# blockMesh – Case 21 (cavity)

- Generating Vertices

```
../case01/system/blockMeshDict
scale 0.1;

vertices
(
    (0 0 0)     // id: 0
    (1 0 0)     // id: 1
    (1 1 0)     // id: 2
    (0 1 0)     // id: 3
    (0 0 0.1)   // id: 4
    (1 0 0.1)   // id: 5
    (1 1 0.1)   // id: 6
    (0 1 0.1)   // id: 7
);
```

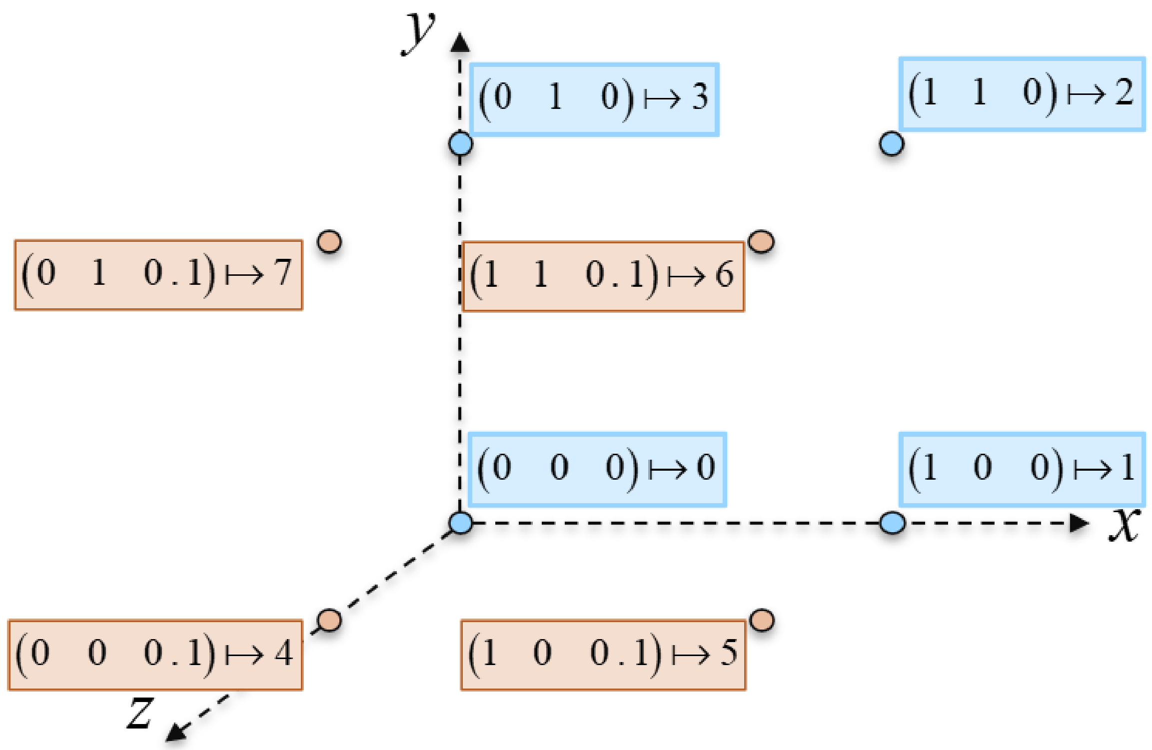


# blockMesh – Case 21 (cavity)

- Generating Vertices

../case01/system/blockMeshDict

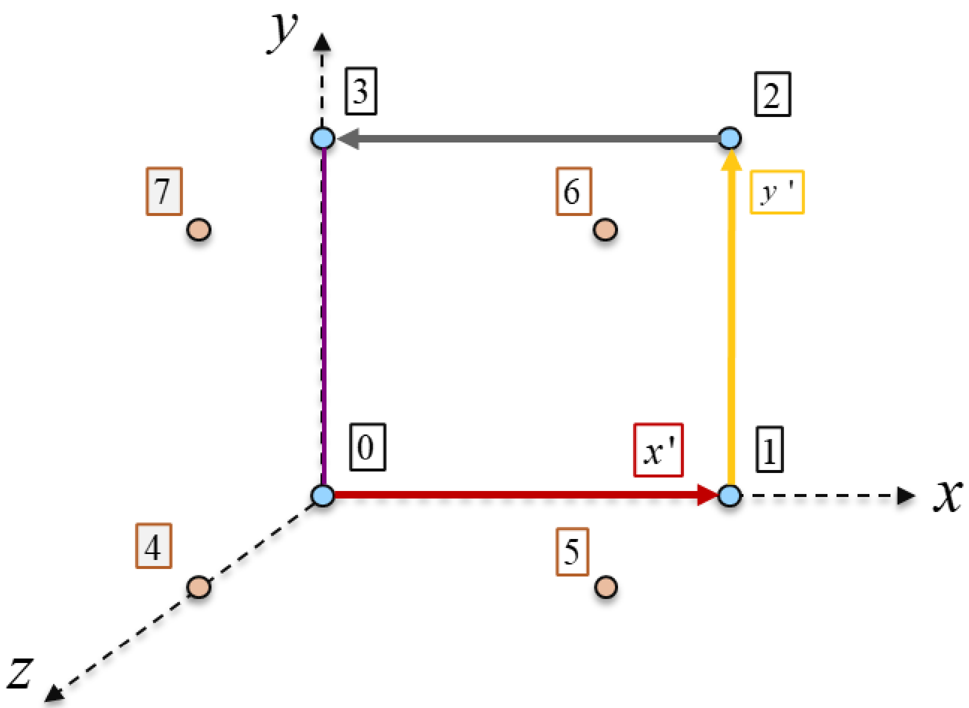
```
scale 0.1;

vertices
(
    (0 0 0)       // id: 0
    (1 0 0)       // id: 1
    (1 1 0)       // id: 2
    (0 1 0)       // id: 3
    (0 0 0.1)     // id: 4
    (1 0 0.1)     // id: 5
    (1 1 0.1)     // id: 6
    (0 1 0.1)     // id: 7
);
```



# blockMesh – Case 21 (cavity)

- Generating Vertices

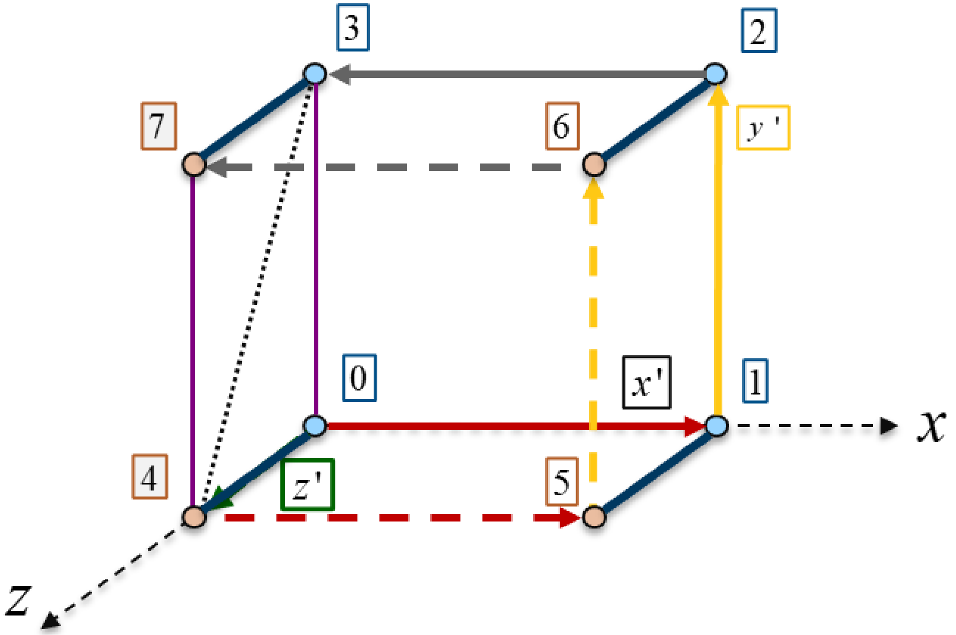
```
../case01/system/blockMeshDict
blocks
(
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
);
```

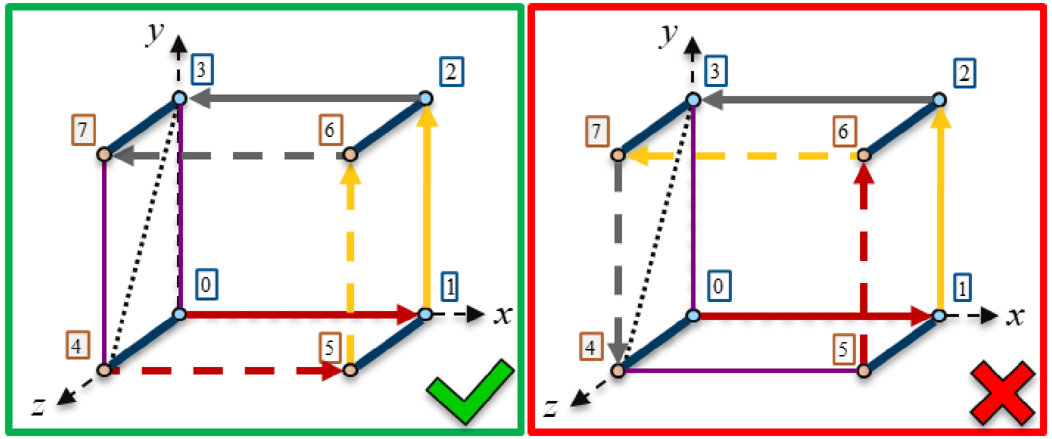


# blockMesh – Case 21 (cavity)

- Generating Vertices

```
../case01/system/blockMeshDict
blocks
(
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
);
```



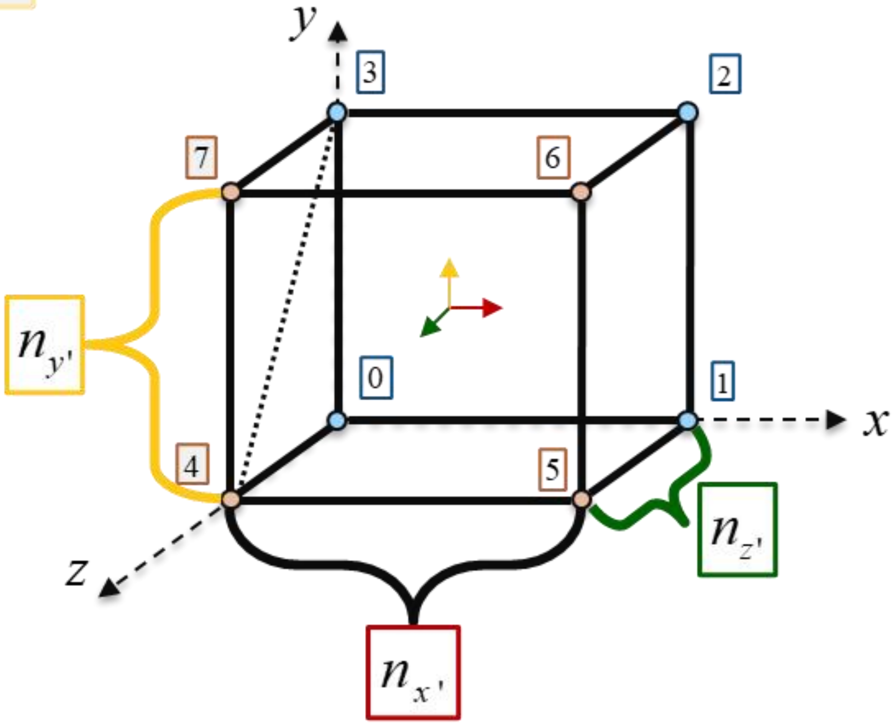



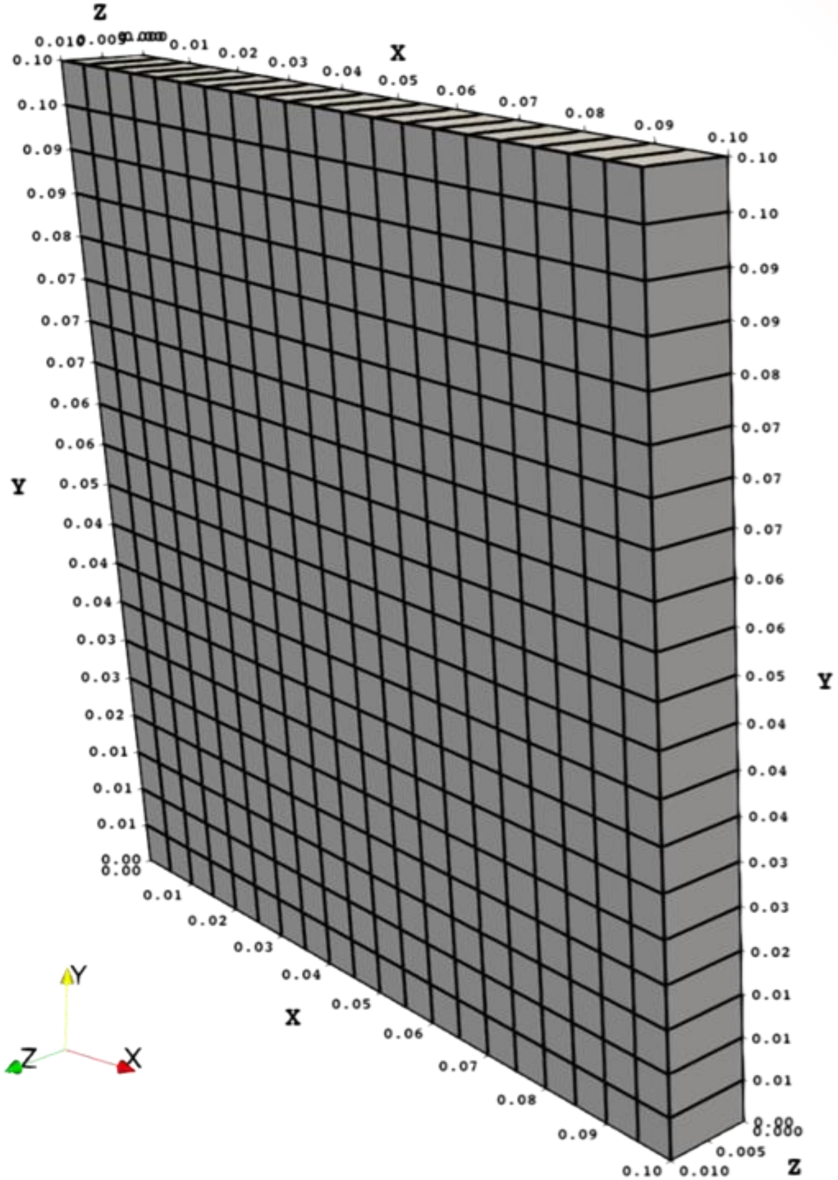
# blockMesh – Case 21 (cavity)

- Slicing block

```
../case01/system/blockMeshDict
blocks
(
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
);
```

$x'$ $y'$ $z'$ $n_{x'}$ $n_{y'}$ $n_{z'}$



# blockMesh – Case 21 (cavity)

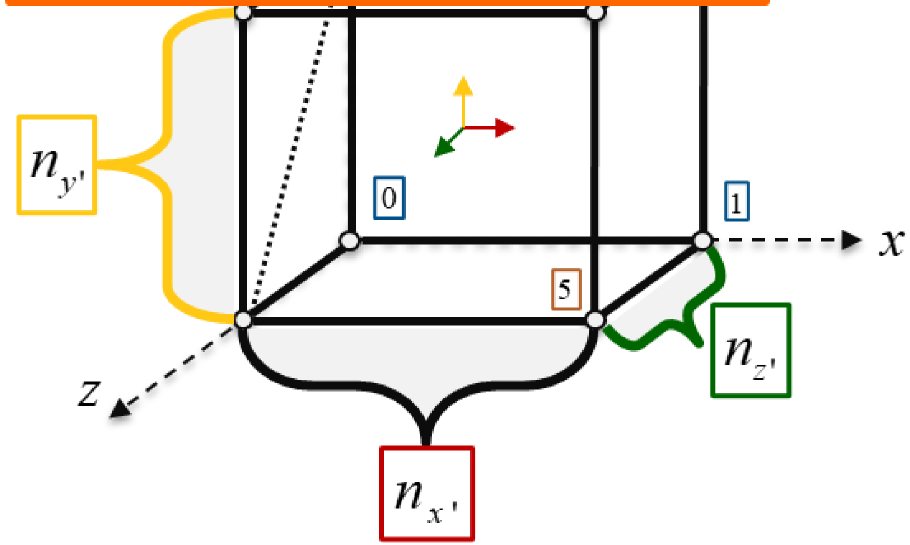
- Extra Functionalities

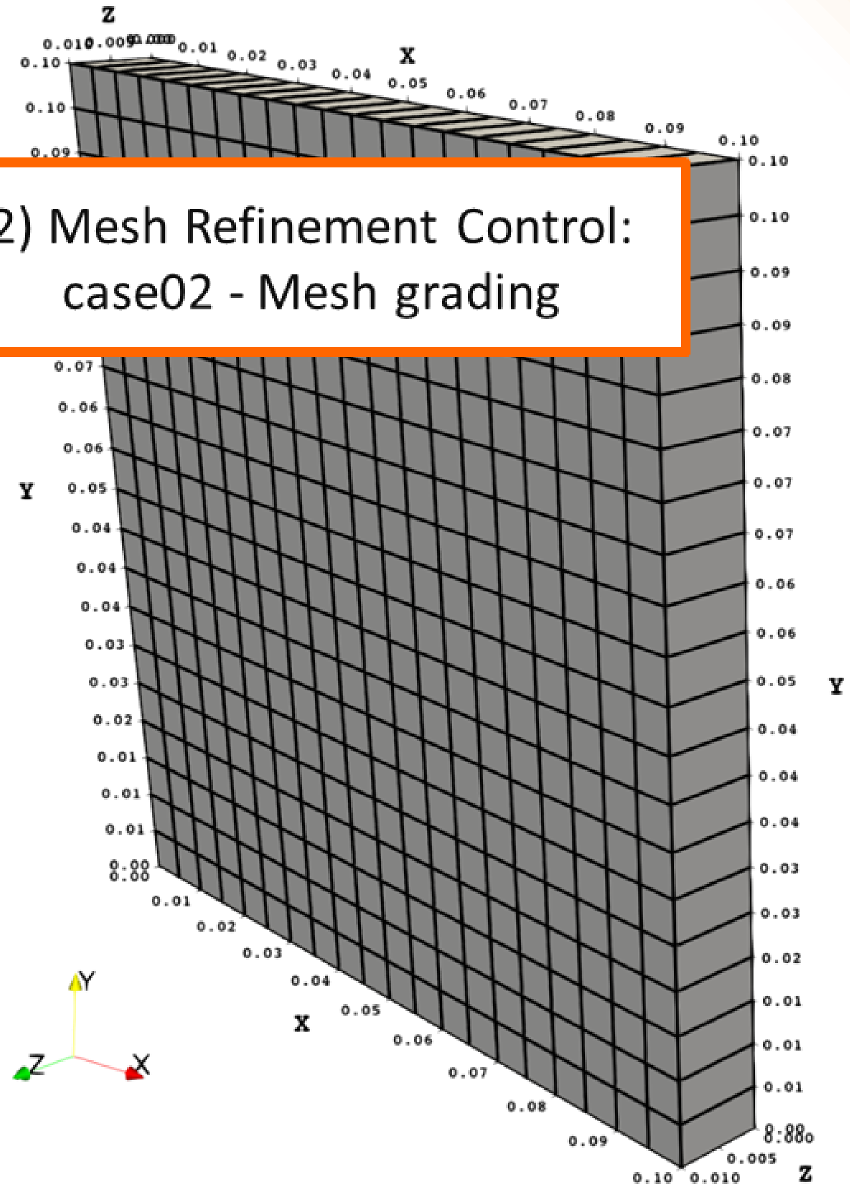
```
../case01/system/blockMeshDict
blocks
(
      hex (0 1 2 3 4 5 6 7) (20 20  1) simpleGrading (1 1 1)
);

edges
(
);
```

$z'$ $n_{x'}$ $n_{y'}$ $n_{z'}$ $x'$ $y'$

2) Mesh Refinement Control: case02 - Mesh grading

4) Curved Surfaces: Designing curvilinear regions




CRheo@IPC

iPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

IPPD Institute of Polymer Processing and Digital Transformation

# blockMesh – Case 21 (cavity)

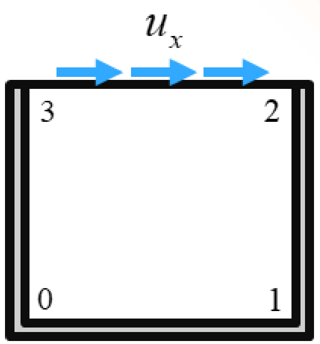
- Naming and Predefining Boundaries

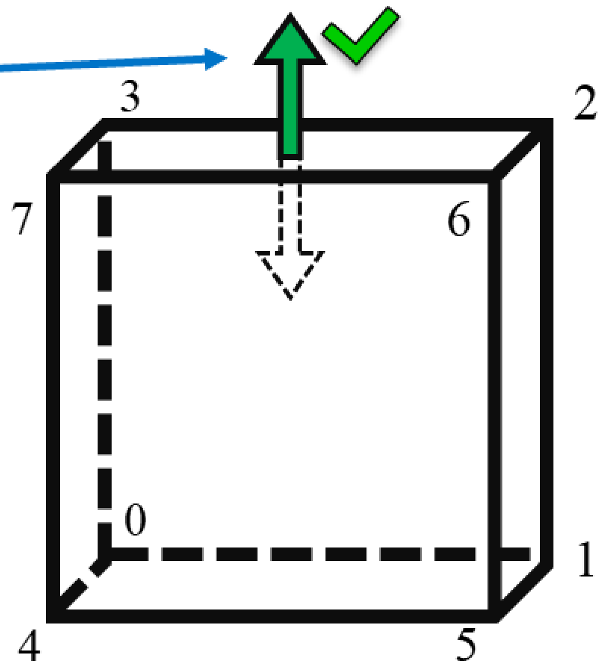
../case01/system/blockMeshDict

```
boundary
(
    movingWall
    {
        type wall;
        faces
        (
            (3 7 6 2)
        );
    }
    fixedWalls
    {
        type wall;
        faces
        (
            (0 4 7 3)
            (2 6 5 1)
            (1 5 4 0)
        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);
```

User input name for the followed face(s)

Base type boundary condition




CRheo@IPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

IPPD Institute of Polymer Processing and Digital Transformation

# blockMesh – Case 21 (cavity)

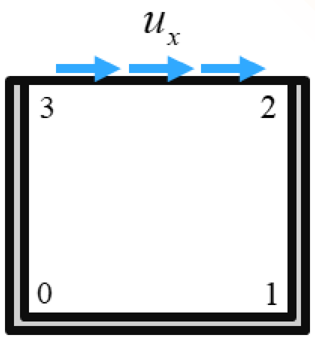
- Naming and Predefining Boundaries

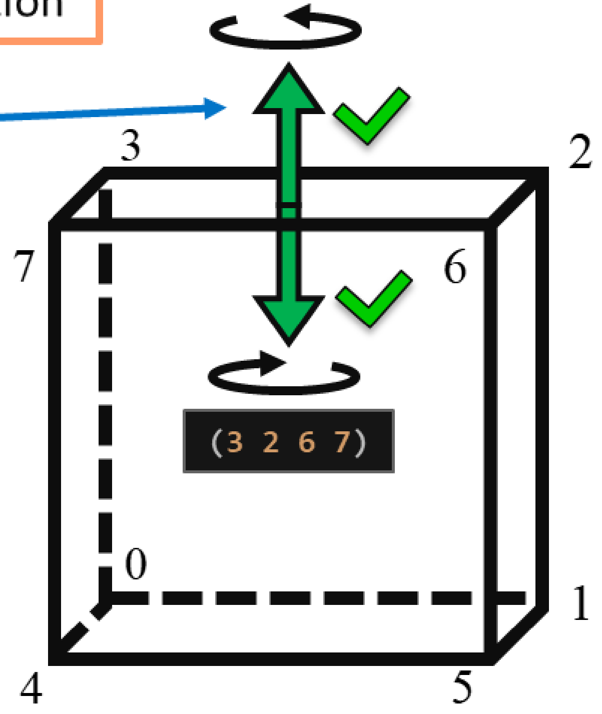
../case01/system/blockMeshDict

```
boundary
(
    movingWall
    {
        type wall;
        faces
        (
            (3 7 6 2)
        );
    }
    fixedWalls
    {
        type wall;
        faces
        (
            (0 4 7 3)
            (2 6 5 1)
            (1 5 4 0)
        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);
```

User input name for the followed face(s)

Base type boundary condition



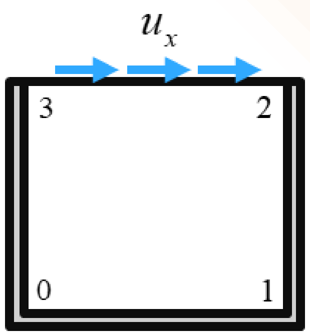



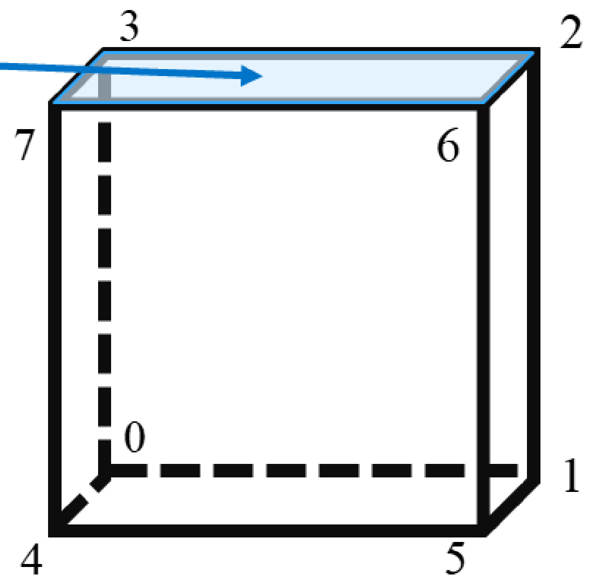
# blockMesh – Case 21 (cavity)

- Naming and Predefining Boundaries

../case01/system/blockMeshDict

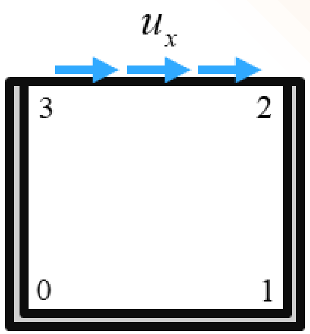
```
boundary
(
    movingWall
    {
        type wall;
        faces
        (
            (3 7 6 2)
        );
    }
    fixedWalls
    {
        type wall;
        faces
        (
            (0 4 7 3)
            (2 6 5 1)
            (1 5 4 0)
        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);
```



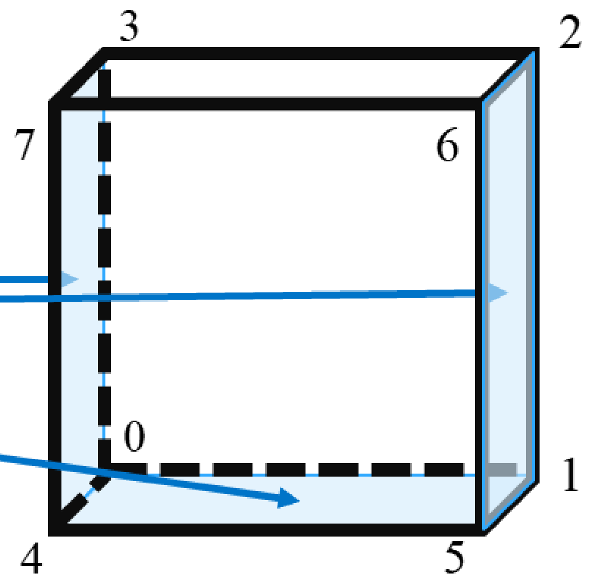
# blockMesh – Case 21 (cavity)

- Naming and Predefining Boundaries



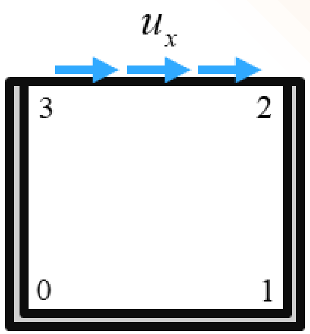

../case01/system/blockMeshDict

```
boundary
(
    movingWall
    {
        type wall;
        faces
        (
            (3 7 6 2)
        );
    }
    fixedWalls
    {
        type wall;
        faces
        (
            (0 4 7 3)
            (2 6 5 1)
            (1 5 4 0)
        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);
```



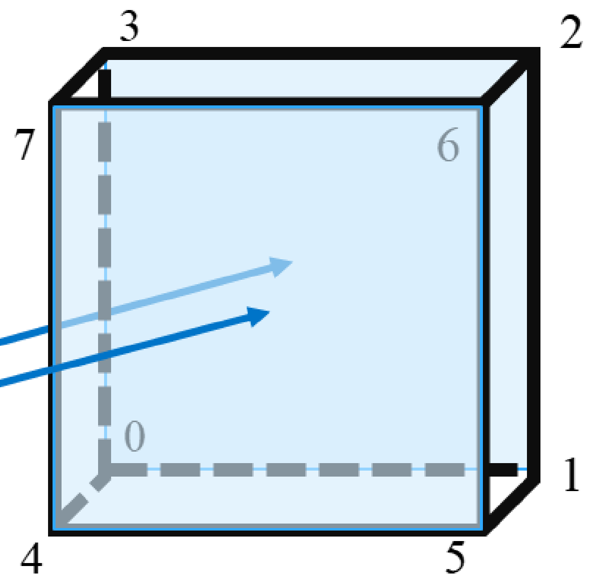
# blockMesh – Case 21 (cavity)

- Naming and Predefining Boundaries




../case01/system/blockMeshDict

```
boundary
(
    movingWall
    {
        type wall;
        faces
        (
            (3 7 6 2)
        );
    }
    fixedWalls
    {
        type wall;
        faces
        (
            (0 4 7 3)
            (2 6 5 1)
            (1 5 4 0)
        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);
```




CRheo@IPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

IPPD Institute of Polymer Processing and Digital Transformation

# blockMesh – Case 21 (cavity - Mesh Grading)

1. >> blockMesh
2. Check results in Paraview

**Hands on (change and visualize)**

1. Number of cells
2. Block height from 1 dm to 2 dm

CRheo@IPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

IPPD Institute of Polymer Processing and Digital Transformation

# blockMesh – Case 21 (cavity - Mesh Grading)

1. >> run
2. >> cd case21/grading/
3. `Visualize blockMeshDict file in VSCode`

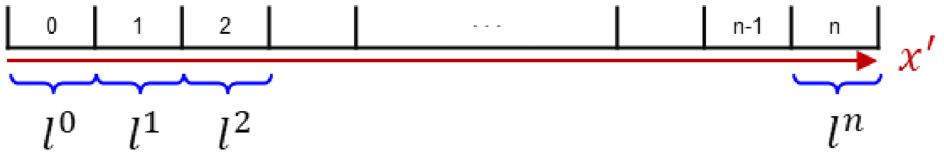
# blockMesh – Case 21 (cavity - Mesh Grading)

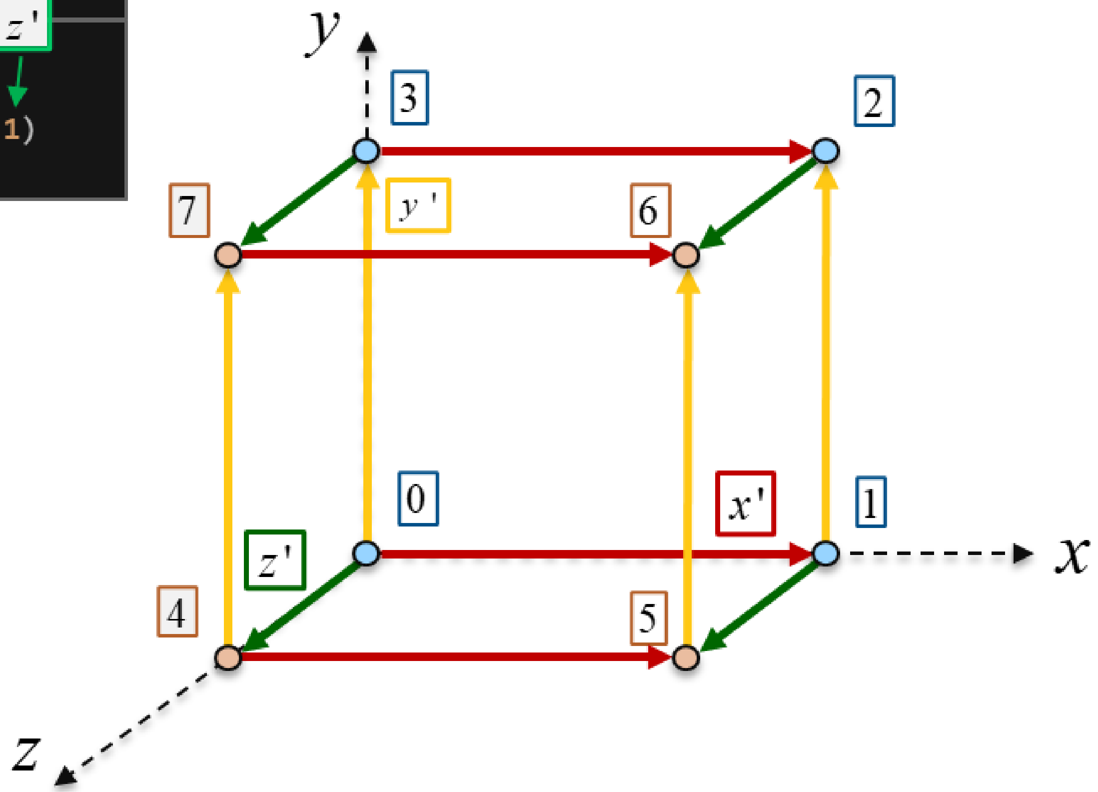
- Mesh grading for stretching the mesh towards one or more Planes.
- cavity Cavity case

```
../case01/system/blockMeshDict
31.  blocks
32.  (
33.      hex (0 1 2 3 4 5 6 7) (20 20  1) simpleGrading ( 1  1  1)
34.  );
```

simpleGrading = edges' cells expansion ratios

$$f_i = \frac{l_i^n}{l_i^0} \qquad f_{x'} = \frac{l_{x'}^n}{l_{x'}^0}$$



# blockMesh – Case 21 (cavity - Mesh Grading)

```
../case02/system/blockMeshDict

blocks
(
  …
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading ( 1.0  1.0  1.0 ) // original
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (10.0  1.0  1.0 ) // Fig. A
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading ( 0.1  1.0  1.0 ) // Fig. B
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading ( 0.1  0.1  1.0 ) // Fig. C
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading ( 0.1 10.0  1.0 ) // Fig. D

  // Compare the setup of the following block with Fig. A

    hex (1 2 3 0 5 6 7 4) (20 20 1) simpleGrading ( 1.0  10.0  1.0 ) // Fig. E
```

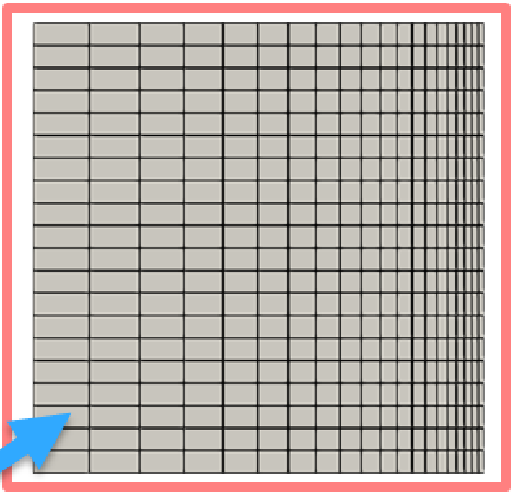


Fig.E

=

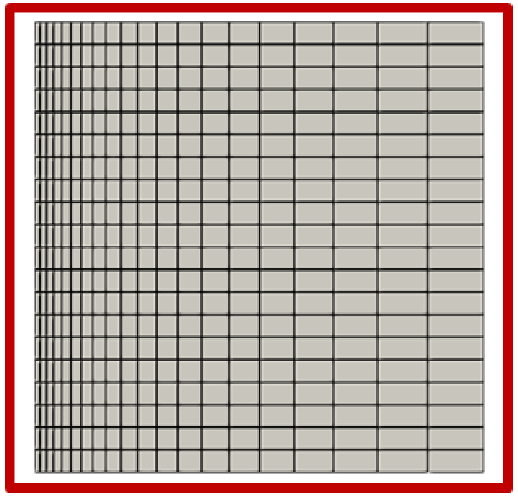


Fig.A

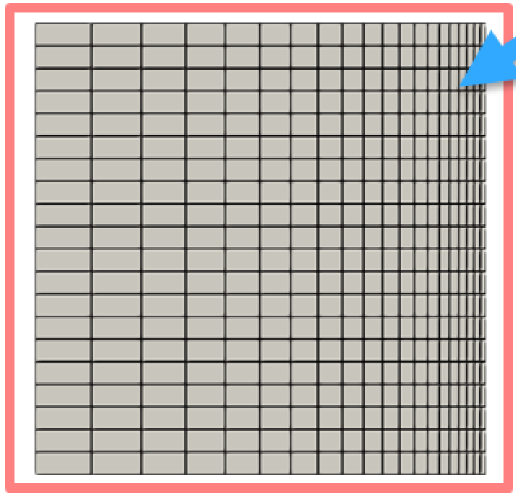


Fig.B

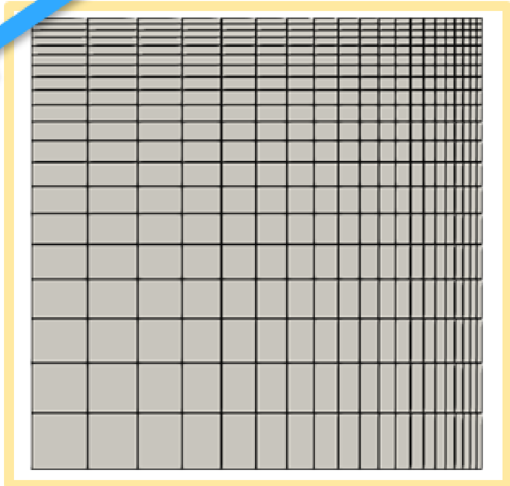


Fig.C

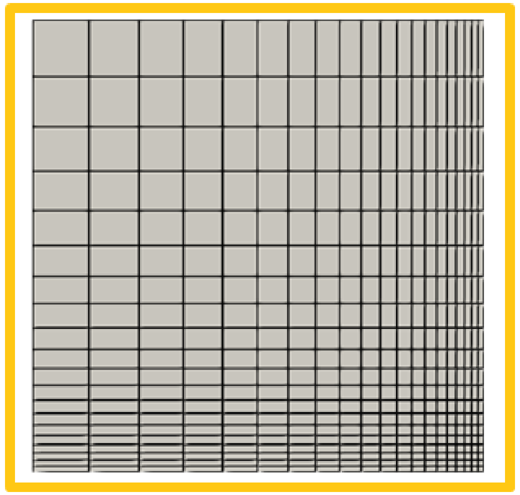


Fig.D

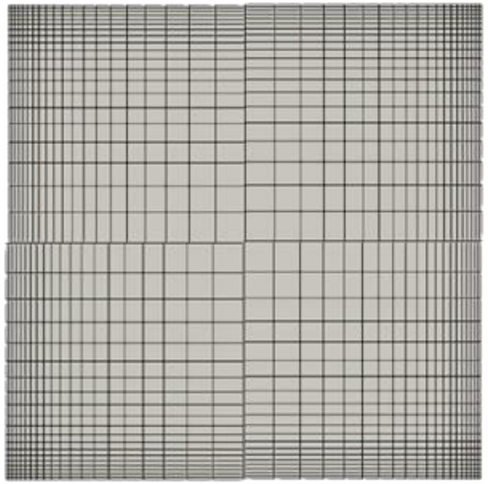
CRheo@IPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

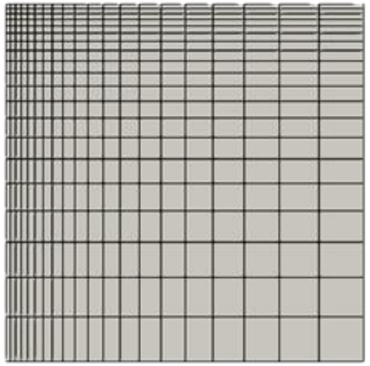
IPPD Institute of Polymer Processing and Digital Transformation

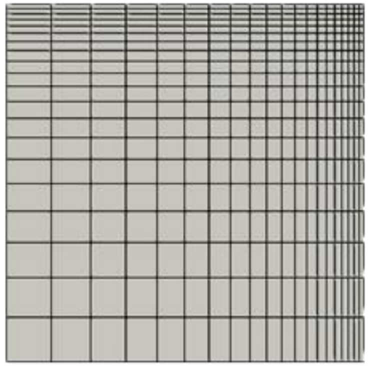
# blockMesh – Case 21 (cavity - Mesh Grading)

```
../case02/system/blockMeshDict

blocks
(
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (10.0  0.1  1.0 ) // Block 0
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading ( 0.1  0.1  1.0 ) // Block 1
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (10.0 10.0  1.0 ) // Block 2
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading ( 0.1 10.0  1.0 ) // Block 3

```



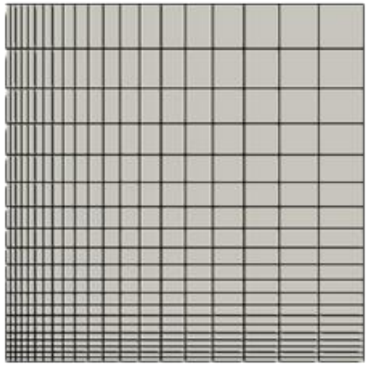
Block 0

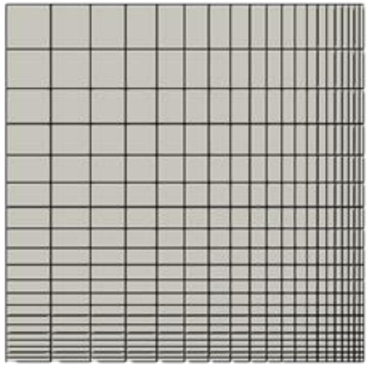




Block 1

Block 2





Block 3

# blockMesh – Case 21 (cavity)

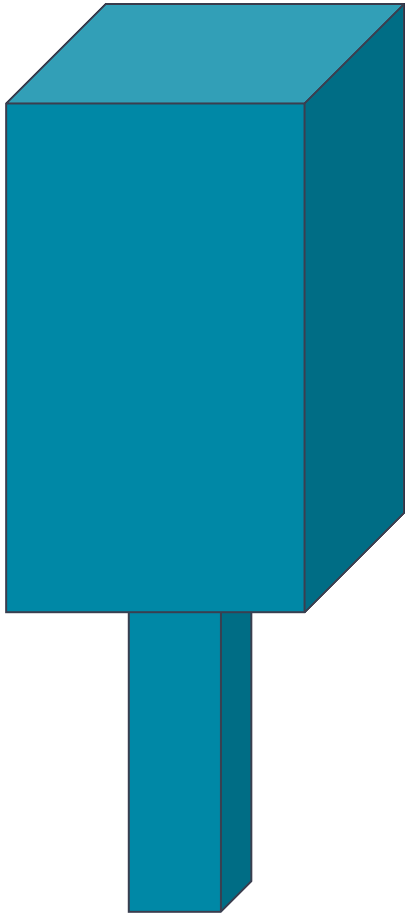
1. >> blockMesh
2. Check results in Paraview

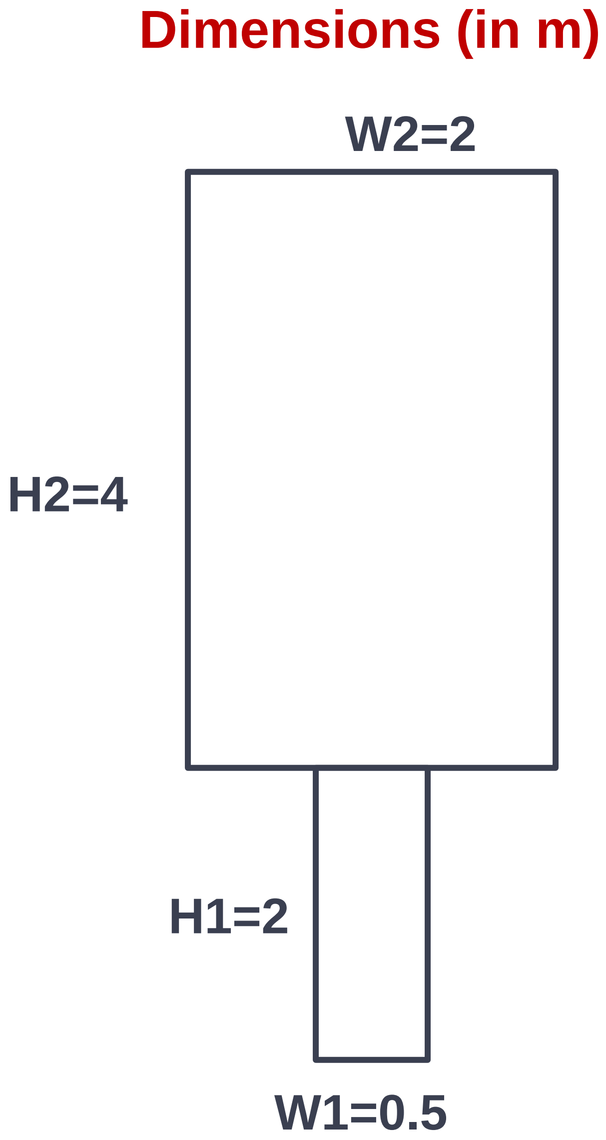
**Hands on (change and visualize)**

1. Test the different mesh grading configurations

# blockMesh – Case 22 (General)

## Geometry

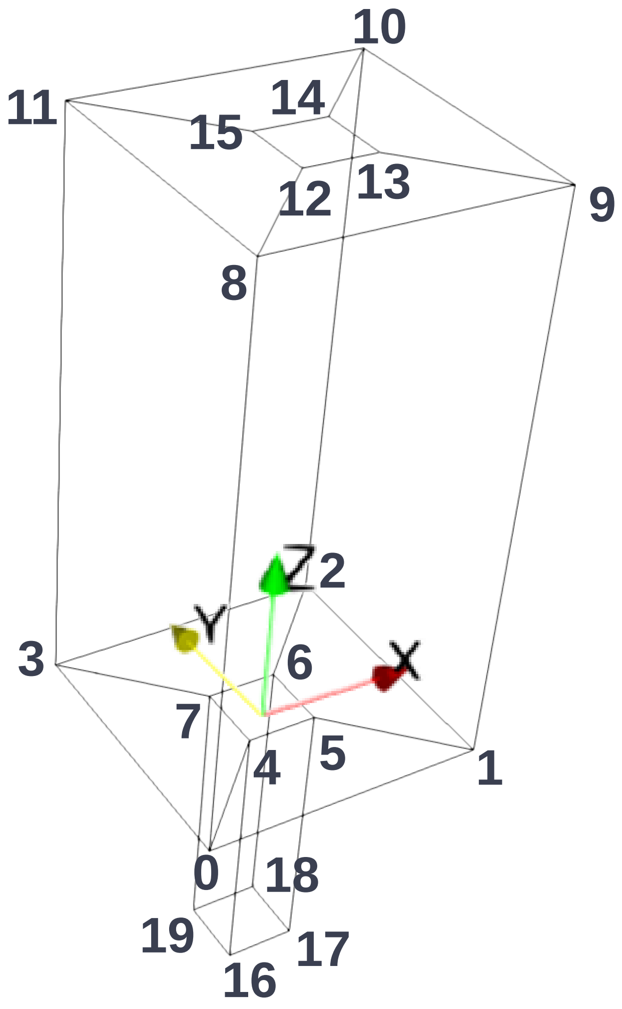




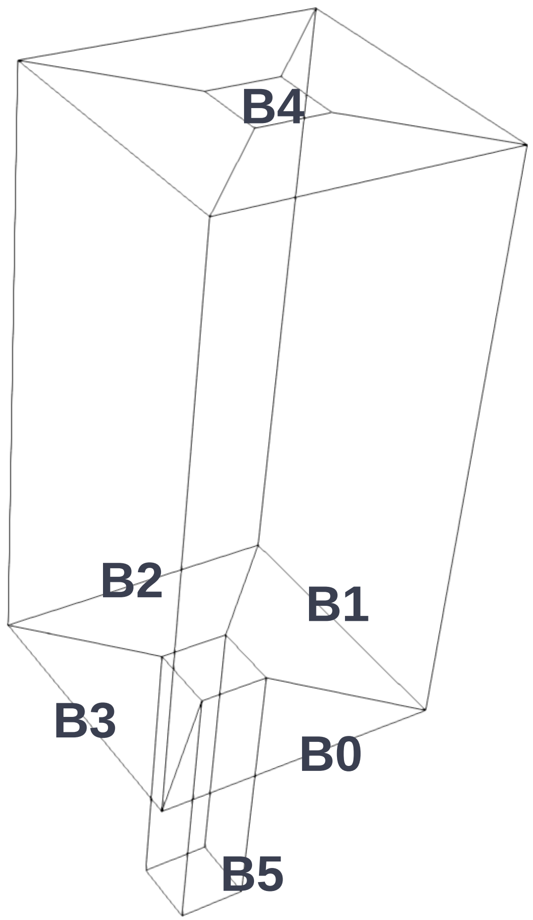
# blockMesh – Case 22 (General)

## Geometry

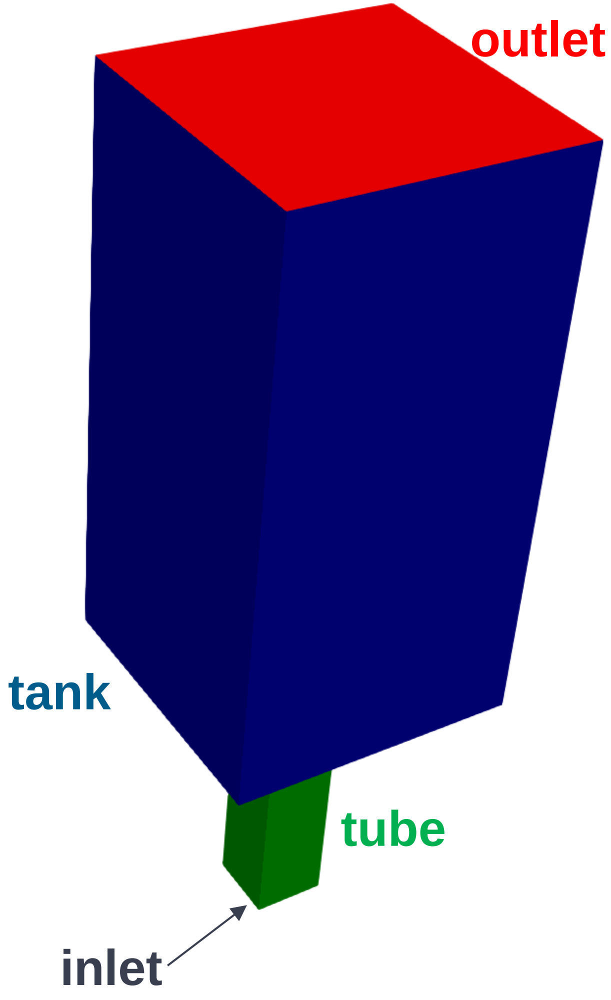
Vertices




Blocks



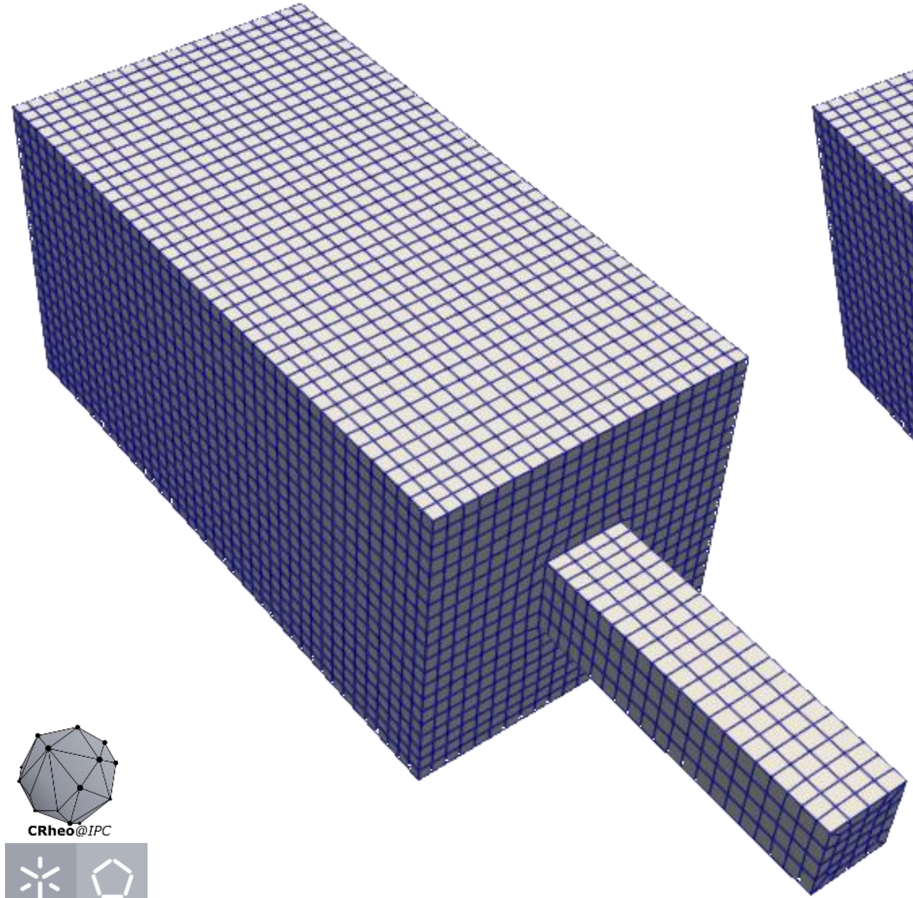

Patches



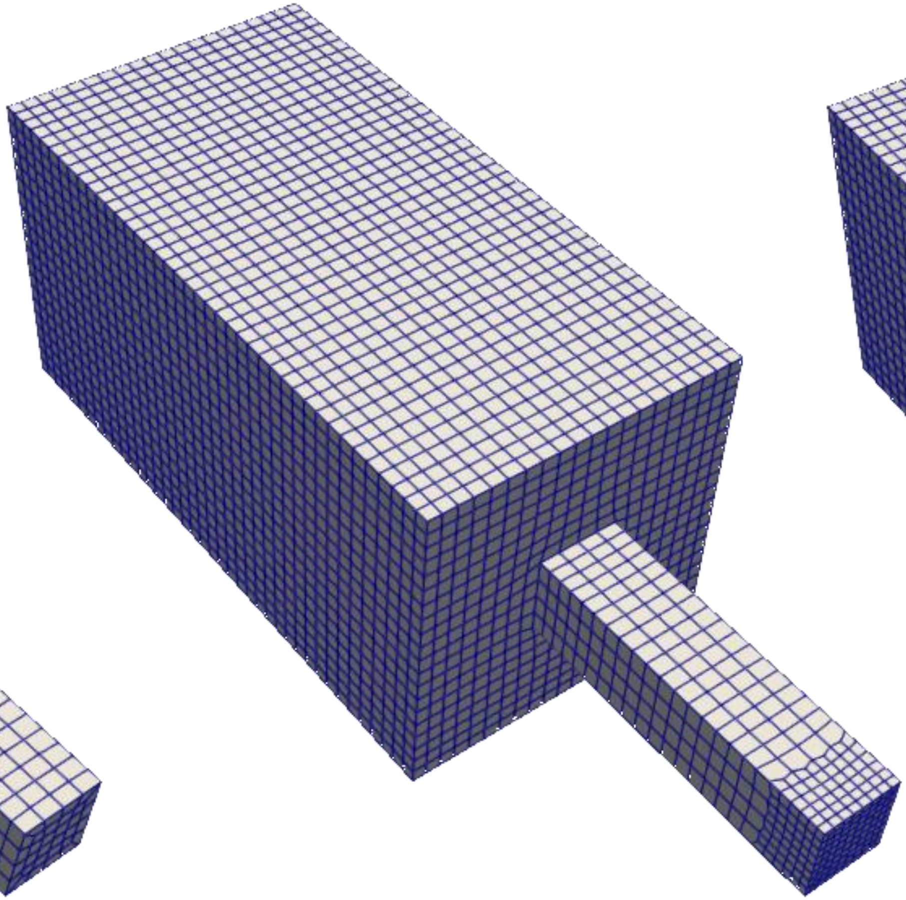
# blockMesh – Case 22 (General)

1. >> run
2. >> cd  case22/general
3. Visualize blockMeshDict file in VSCode
4. >> blockMesh
5. Visualize the Mesh in paraview

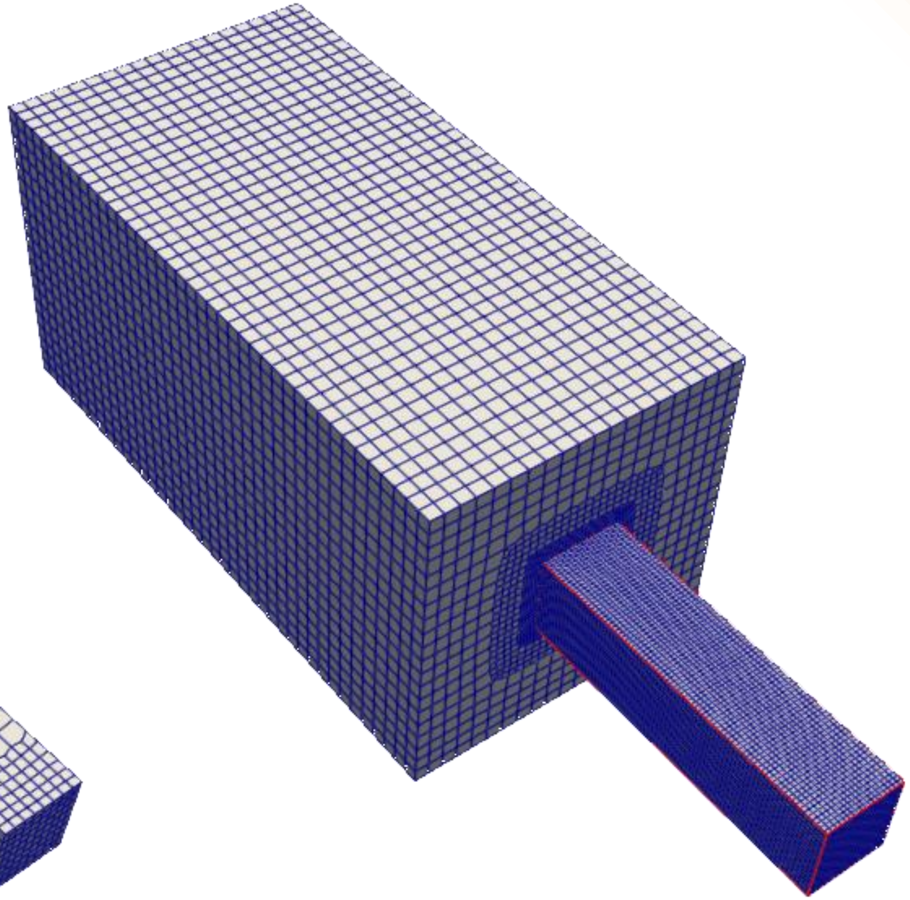
# cfMesh – Case22 (Parameterized Meshing)



Coarse Mesh without Refinement



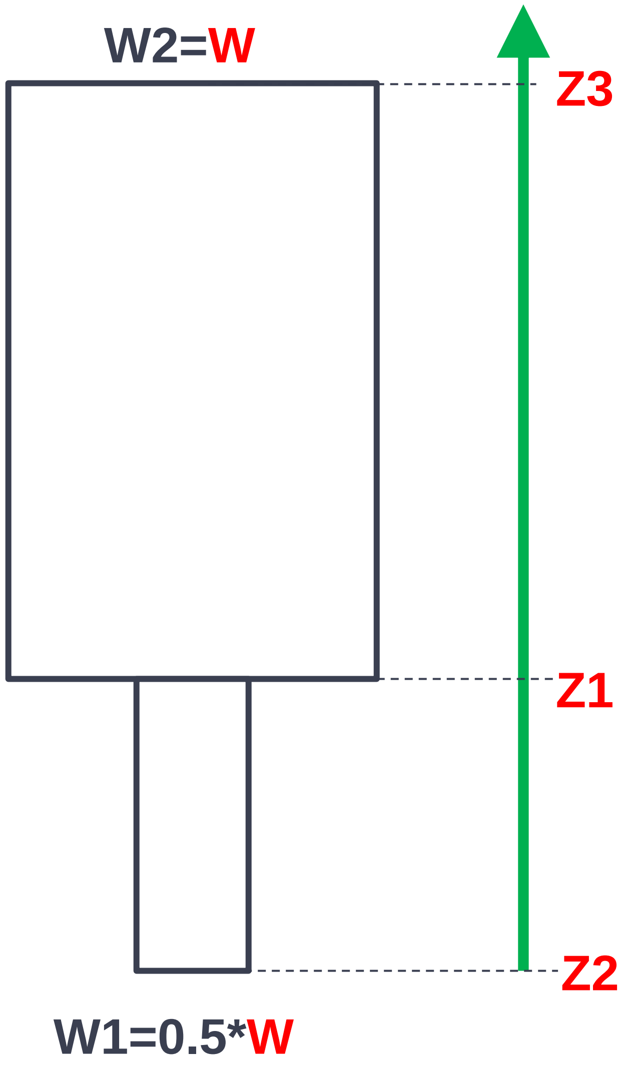
Local Refinement at the Inlet



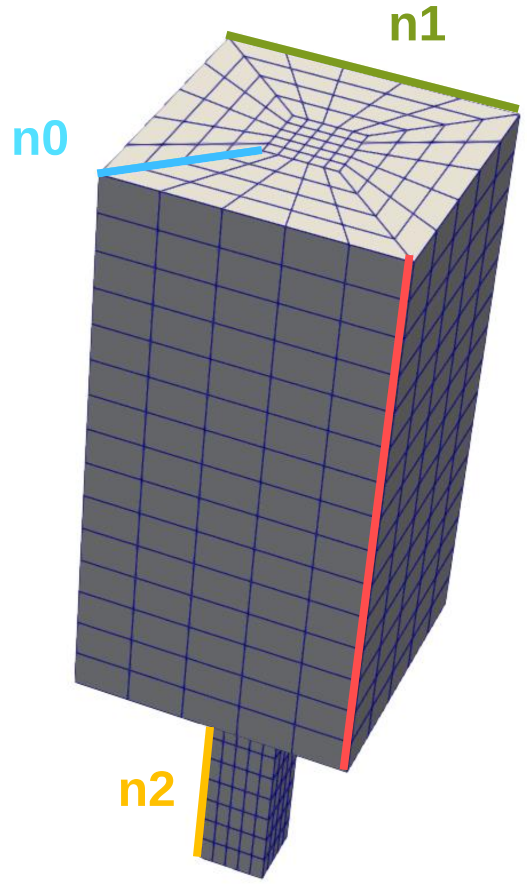
Object Refinement as a box

# blockMesh – Case 22 (Parameterized)
## Geometry & Mesh

**Dimensions**




**Number of cells**



# blockMesh – Case 22 (Parameterized)

1. >> run
2. >> cd  case22/parametrized/
3. `Visualize blockMeshDict file in VSCode`

CRheo@IPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

IPPD Institute of Polymer Processing and Digital Transformation

# blockMesh – Case 22 (Parameterized)

- Parameterized Geometry and Mesh Generation

```
//Parameters
        W   2;

        Z1  0;
        Z2  -2.0;
        Z3  4.0;

        n0  4;
        n1  5;
        n2  10;
        n3  15;
```

```
// Calculations
        X0      #calc "-0.5*$W";
        Y0      #calc "1*$X0";
        X1      #calc "-1*$X0";
        Y1      #calc "1*$X0";
        X2      #calc "-1*$X0";
        Y2      #calc "-1*$X0";
        X3      #calc "1*$X0";
        Y3      #calc "-1*$Y0";

        X4      #calc "0.25*$X0";
        Y4      #calc "0.25*$X0";
        X5      #calc "-0.25*$X0";
        Y5      #calc "0.25*$X0";
        X6      #calc "-0.25*$X0";
        Y6      #calc "-0.25*$X0";
        X7      #calc "0.25*$X0";
        Y7      #calc "-0.25*$X0";
```

CRheo@IPC

iPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

JKU IPPD Institute of Polymer Processing and Digital Transformation

# blockMesh – Case 22 (Parameterized)

- Parameterized Geometry and Mesh Generation

```
vertices
(
    (-1 -1 0) //0
    (1 -1 0) //1
    (1 1 0) //2
    (-1 1 0) //3
    (-0.25 -0.25 0) //4
    (0.25 -0.25 0) //5
    (0.25 0.25  0) //6
    (-0.25 0.25 0) //7
    (-1 -1 4) //8
    (1 -1 4) //9
    (1 1 4) //10
    (-1 1 4) //11
    (-0.25 -0.25 4) //12
    (0.25 -0.25 4) //13
    (0.25 0.25  4) //14
    (-0.25 0.25 4) //15
    (-0.25 -0.25 -2) //16
    (0.25 -0.25 -2) //17
    (0.25 0.25  -2) //18
    (-0.25 0.25 -2) //19
```

Normal Configuration of blockMeshDict

```
    ($X0 $Y0 $Z1) //0
    ($X1 $Y1 $Z1)//1
    ($X2 $Y2 $Z1)//2
    ($X3 $Y3 $Z1) //3
    ($X4 $Y4 $Z1)//4
    ($X5 $Y5 $Z1)//5
    ($X6 $Y6 $Z1) //6
    ($X7 $Y7 $Z1) //7

    ($X0 $Y0 $Z3) //8
    ($X1 $Y1 $Z3)//9
    ($X2 $Y2 $Z3)//10
    ($X3 $Y3 $Z3) //11
    ($X4 $Y4 $Z3)//12
    ($X5 $Y5 $Z3)//13
    ($X6 $Y6 $Z3) //14
    ($X7 $Y7 $Z3) //15

    ($X4 $Y4 $Z2)//16
    ($X5 $Y5 $Z2)//17
    ($X6 $Y6 $Z2) //18
    ($X7 $Y7 $Z2) //19
```

Parameterized Configuration of blockMeshDict

# blockMesh – Case 22 (Parameterized)

- Parameterized Geometry and Mesh Generation

```
blocks
(

    hex (0 1 5 4 8 9 13 12) (5 4 15) simpleGrading (1 1 1) //B0
    hex (5 1 2 6 13 9 10 14) (4 5 15) simpleGrading (1 1 1) //B1
    hex (7 6 2 3 15 14 10 11) (5 4 15) simpleGrading (1 1 1) //B2
    hex (0 4 7 3 8 12 15 11) (4 5 15) simpleGrading (1 1 1) //B3
    hex (4 5 6 7 12 13 14 15) (5 5 15) simpleGrading (1 1 1) //B4
    hex (16 17 18 19 4 5 6 7) (5 5 10) simpleGrading (1 1 1) //B5
)
```

```
blocks
(
    hex (0 1 5 4 8 9 13 12) ($n1 $n0 $n3) simpleGrading (1 1 1) //B0
    hex (5 1 2 6 13 9 10 14) ($n0 $n1 $n3) simpleGrading (1 1 1) //B1
    hex (7 6 2 3 15 14 10 11) ($n1 $n0 $n3) simpleGrading (1 1 1) //B2
    hex (0 4 7 3 8 12 15 11) ($n0 $n1 $n3) simpleGrading (1 1 1) //B3
    hex (4 5 6 7 12 13 14 15) ($n1 $n1 $n3) simpleGrading (1 1 1) //B4
    hex (16 17 18 19 4 5 6 7) ($n1 $n1 $n2) simpleGrading (1 1 1) //B5
);
```

```
n0   4;
n1   5;
n2   10;
n3   15;
```

CRheo@IPC

IPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

JKU IPPD Institute of Polymer Processing and Digital Transformation

# blockMesh – Case 22 (Parameterized)

- Parameterized Geometry and Mesh Generation

No changes for the boundaries

```
boundary
(
    inlet
    {
        type patch;
        faces
        (
            (16 17 18 19)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (8 9 13 12)
            (13 9 10 14)
            (15 14 10 11)
            (8 12 15 11)
            (12 13 14 15)
        );
    }
    tube
    {
        type wall;
        faces
        (
            (16 17 5 4)
            (17 18 6 5)
            (19 7 6 18)
            (19 16 4 7)
        );
    }
    tank
    {
        type wall;
        faces
        (
            (0 1 5 4)
            (5 1 2 6)
            (7 6 2 3)
            (0 4 7 3)
            (0 1 9 8)
            (1 2 10 9)
            (2 3 11 10)
            (3 0 8 11)
        );
    }

);
```

# blockMesh – Case 22 (Parameterized)

1. >> blockMesh
2. Visualize the Mesh in paraview

**Hands on (change and visualize)**

1. Change the parameters and check the result in paraview

CRheo@IPC

iPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

JKU IPPD Institute of Polymer Processing and Digital Transformation

# cfMesh – Case 22 (STL based Meshing)

1. >> run
2. >> cd  case22/cfMesh/
3. >> code .
4. >> cd STLBuilder
5. In VSCode check file myList contents
6. Open all the individual stl files in paraview
7. >> ./joinSTL.x
8. Open the total.stl file in paraview
9. >> cp total.fms ../  ## copy total.fms to the problem root folder
10. >> cd .. ##return to the problem root folder
11. In VSCode check file system/meshDict contents

```
outlet
inlet
tank
tube
```

CRheo@IPC

iPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

IPPD Institute of Polymer Processing and Digital Transformation

# cfMesh – Case 22 (STL based Meshing)

```
surfaceFile "total.fms";

maxCellSize 0.1;

//boundaryCellSize 0.8;

//minCellSize 0.01;

/*localRefinement
{
    inlet
    {
        cellSize 0.05;
        //additionalRefinementLevels 1;
        refinementThickness 0.01;
    }

}*/
```

The file created from STLs

Maximum Cell size

Refine part of the domain near a patch

```
/*objectRefinements
{

    Box
    {
        type        box;
        cellSize    0.05;
        centre      (0 0 -1);
        lengthX     0.5;
        lengthY     0.5;
        lengthZ     2;
    }

}*/
```

For refining part of the domain inside a box

CRheo@IPC

iPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

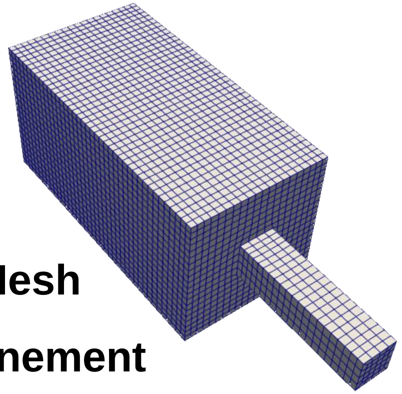
JKU IPPD Institute of Polymer Processing and Digital Transformation

# cfMesh – Case 22 (STL based Meshing)

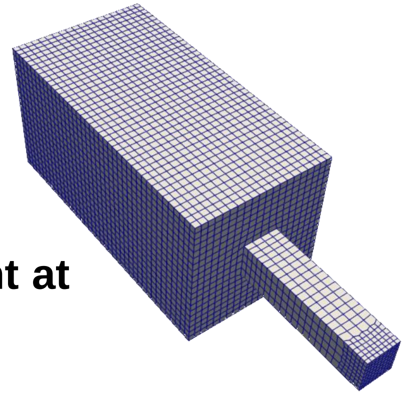
1. >> run #Return to the case root
2. >> cartesianMesh
3. Visualize the results in Paraview

**Hands on (change and visualize)**

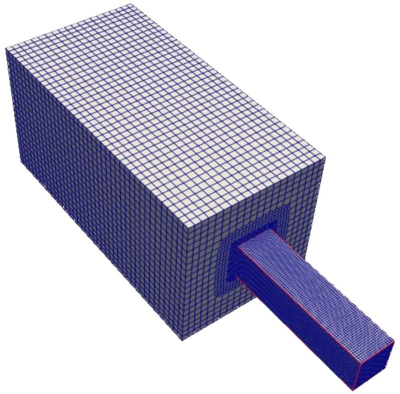
1. Uncomment the 2 mesh refinement level methods, in meshDict file, and check the result in paraview
2. Change the parameters in meshDict file and check the results in paraview



**Coarse Mesh without Refinement**



**Local Refinement at the Inlet**



**Object Refinement as a box**

# Post-processing

OpenFOAM®

ParaView

tecplot®

python™

blender®

# Post-processing

**Introduction**
ParaView's overview

**User Interface**
Tools and features

**Basic usage**
Sources and filters

**Post processing**
Data visualization

**Saving progress**
Images, animations, states

# Post-processing - Introduction

1. What is ParaView?
2. Basics of visualization
3. Why is it important for OpenFoam users?

ParaView
Parallel Visualization Application

# Post-processing - Introduction

1. What is ParaView?
2. Basics of visualization
3. Why is it important for OpenFoam users?

ParaView
Parallel Visualization Application

- An open-source application for visualizing two and three dimensional data sets.
- It is designed to handle complex scientific and engineering data.
- Offers a flexible and intuitive graphical user interface.
- Widely used by many academic, government and commercial institutions.
- Downloaded roughly 100.000 times every year.
- Developed and maintained by Kitware Inc.

# Post-processing - Introduction

1. What is ParaView?
2. Basics of visualization
3. Why is it important for OpenFoam users?

3 basic steps to visualize your data

Reading > Filtering > Rendering

Firstly, data must be read in Paraview.

Next, you may apply any number of filters that process the data to generate extract or derive features from the data.

Finally, a viewable image is rendered from the data.

CRheo@IPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

IPPD Institute of Polymer Processing and Digital Transformation

# Post-processing - Introduction

1. What is ParaView?
2. Basics of visualization
3. Why is it important for OpenFoam users?

**Data Visualization**: Allows the visualization of complex simulations in OpenFOAM, making it easier to understand and explore data results.

**Advanced Analysis**: It offers a wide range of filters and tools to perform in-depth analysis, extract insights, and validate simulations.

**Customization**: Scripting capabilities allow users to create custom workflows, automate tasks, and integrate with OpenFOAM simulations.

**Efficiency**: It accelerates the post-processing workflow, saving time and resources.

CRheo@IPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

IPPD Institute of Polymer Processing and Digital Transformation

# Post-processing – User Interface | Over view

Menu bar

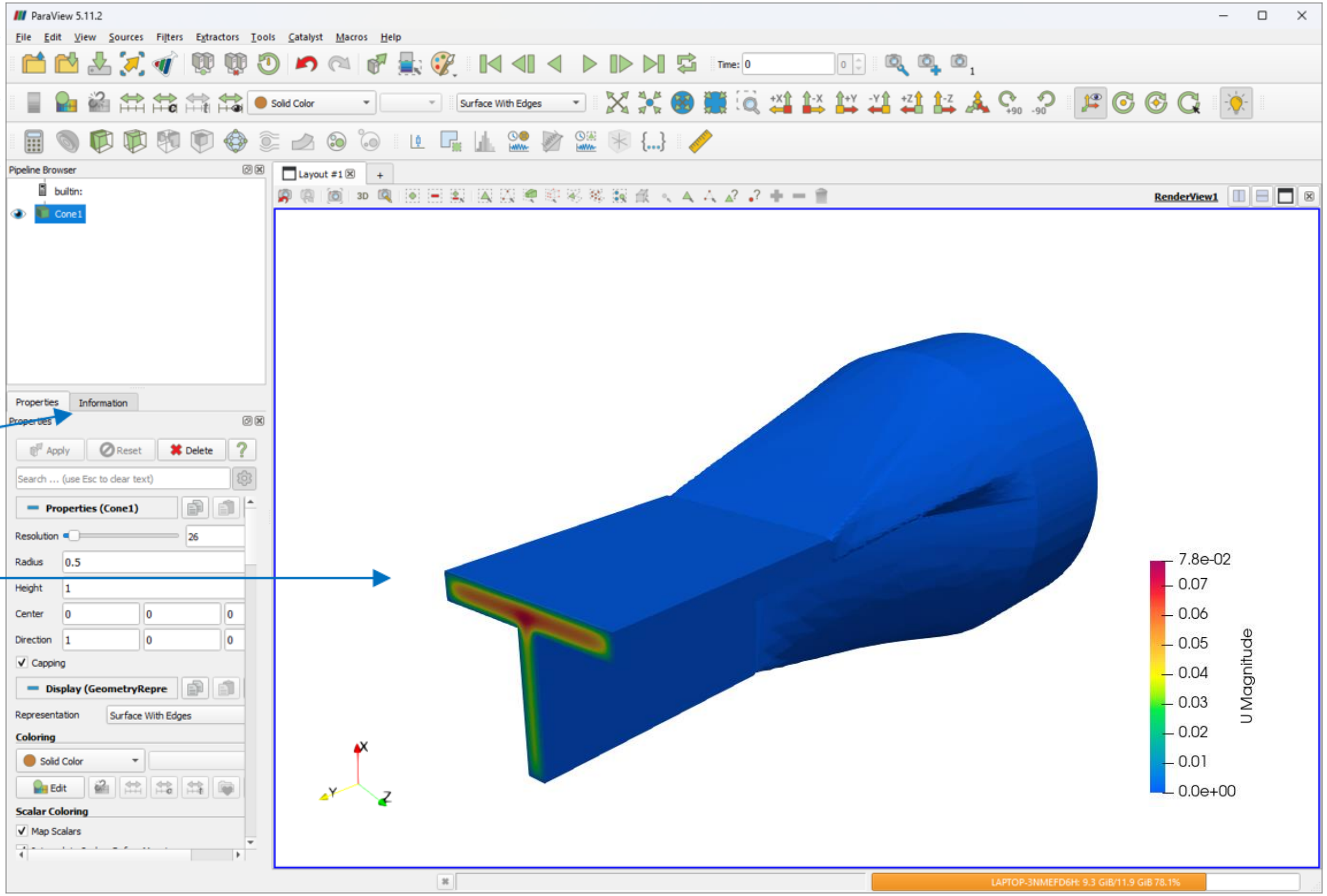
Toolbars

Pipeline Browser

Properties panel

Information panel

Canvas

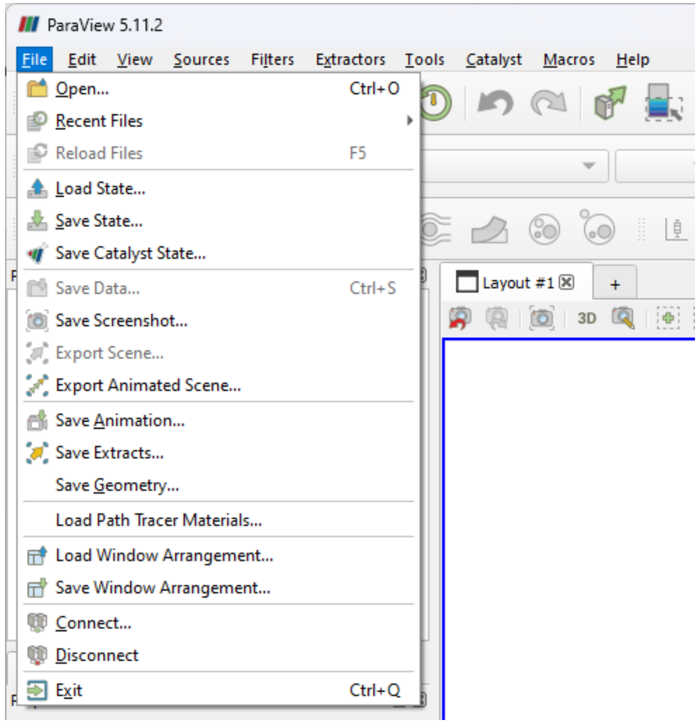


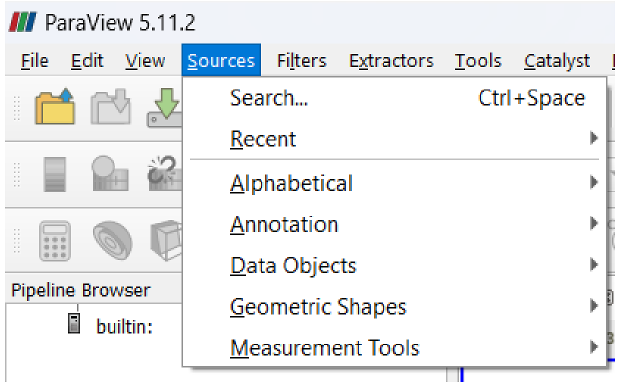


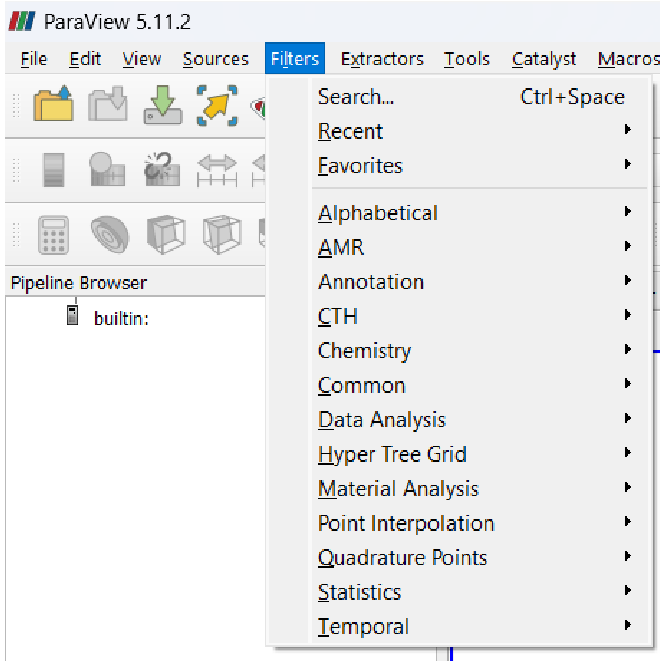

CRheo@IPC
PPS 2024 30 Sept.-03 Oct. | Ferrol · Spain
IPPD Institute of Polymer Processing and Digital Transformation

# Post-processing – User Interface | Menu Bar

The menu bar allows you to access the majority of tools and features, dividing it into sections.

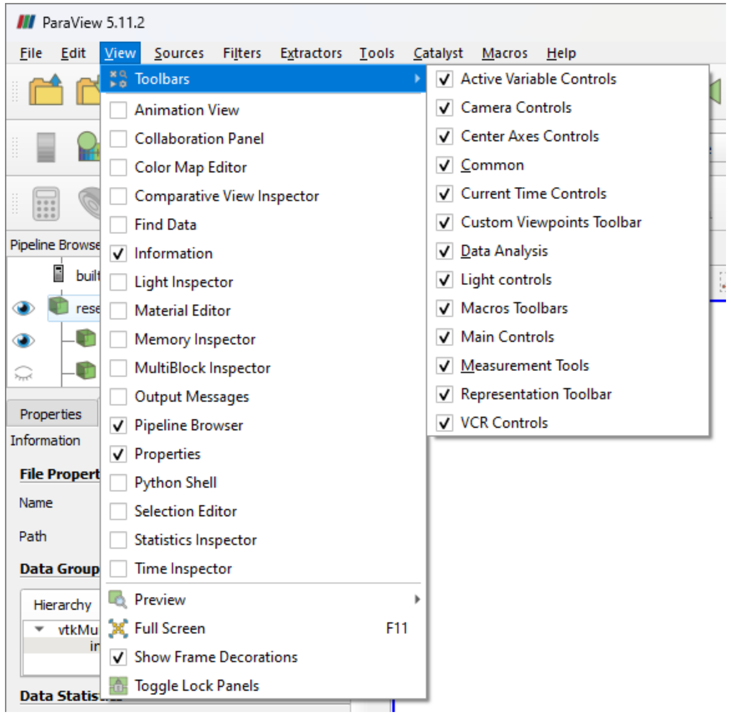



# Post-processing – User Interface | Toolbars

The toolbars provide shortcuts to some features and tools.


Toolbars are customized by hiding/showing what you desire.

Menu bar View - Toolbars



# Post-processing – User Interface | Toolbars

Main Controls

VCR Controls


Current Time Controls

Custom Viewpoints Toolbar


Representation Toolbar


Active Variable Controls


Camera controls

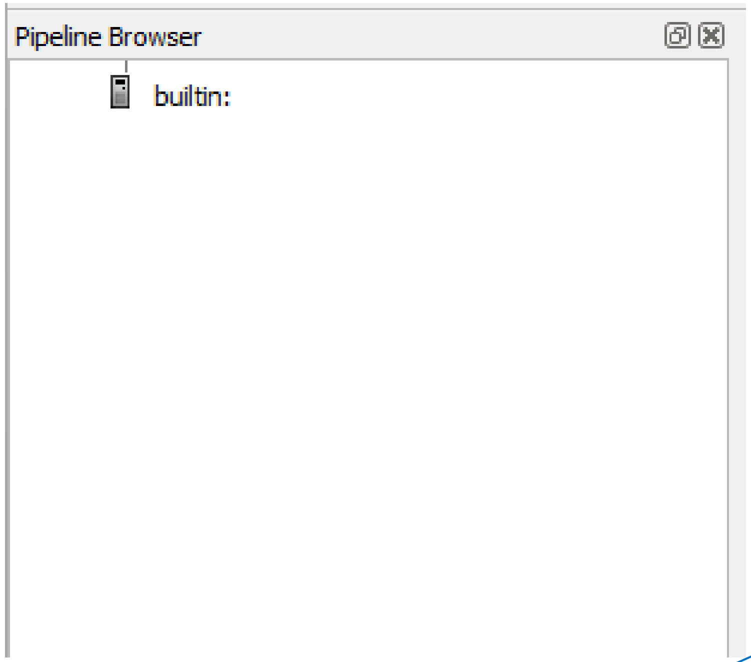
Center Axes Controls

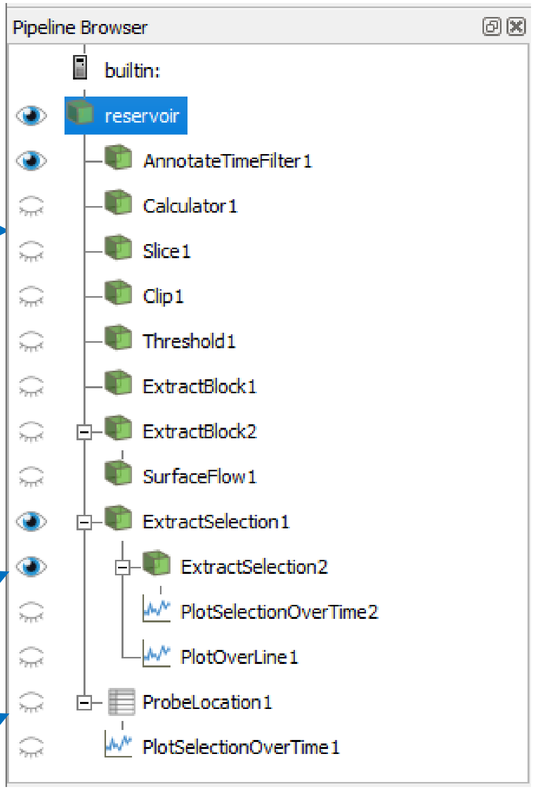
Common Filters


Data Analysis Toolbar

# Post-processing – User Interface | Pipeline browser

The pipeline browser shows the pipeline structure, providing the objects being post processed and its correspondent applied filters.




The objects/filters being shown in the canvas

The objects/filters being hidden in the canvas

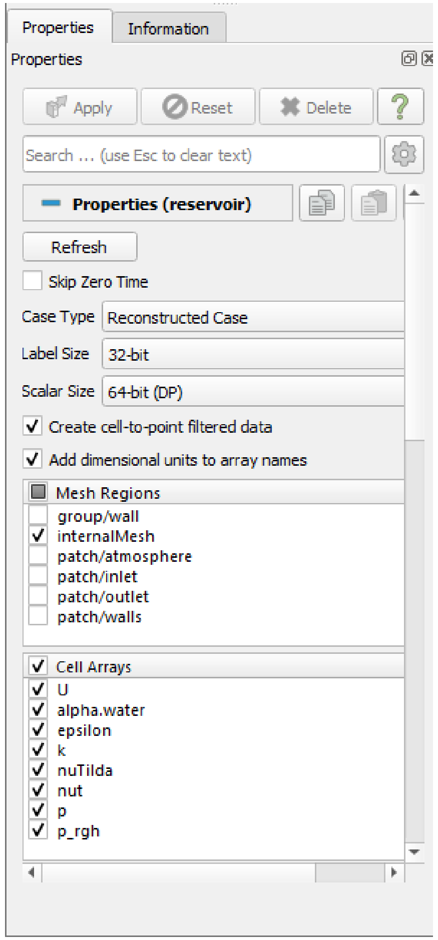
CRheo@IPC

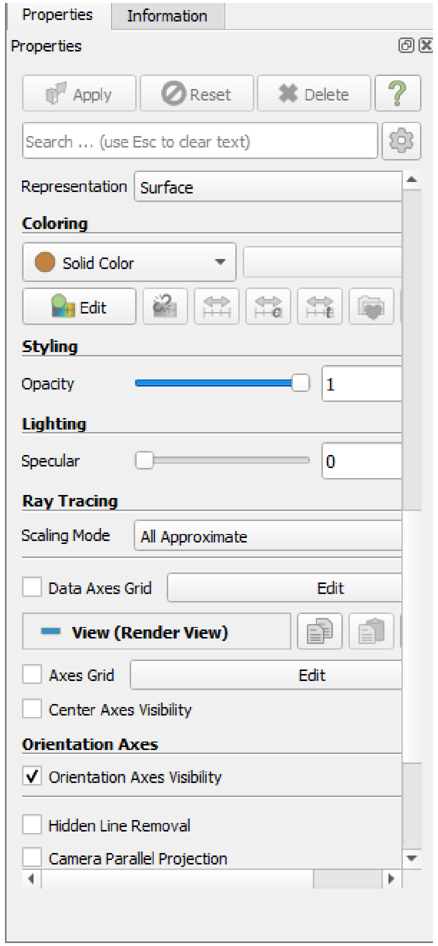
PPS 2024 30 Sept.-03 Oct. | Ferrol · Spain

IPPD Institute of Polymer Processing and Digital Transformation

# Post-processing – User Interface | Properties panel

The properties panel allows you to view and modify some parameters of the selected pipeline object/filter.



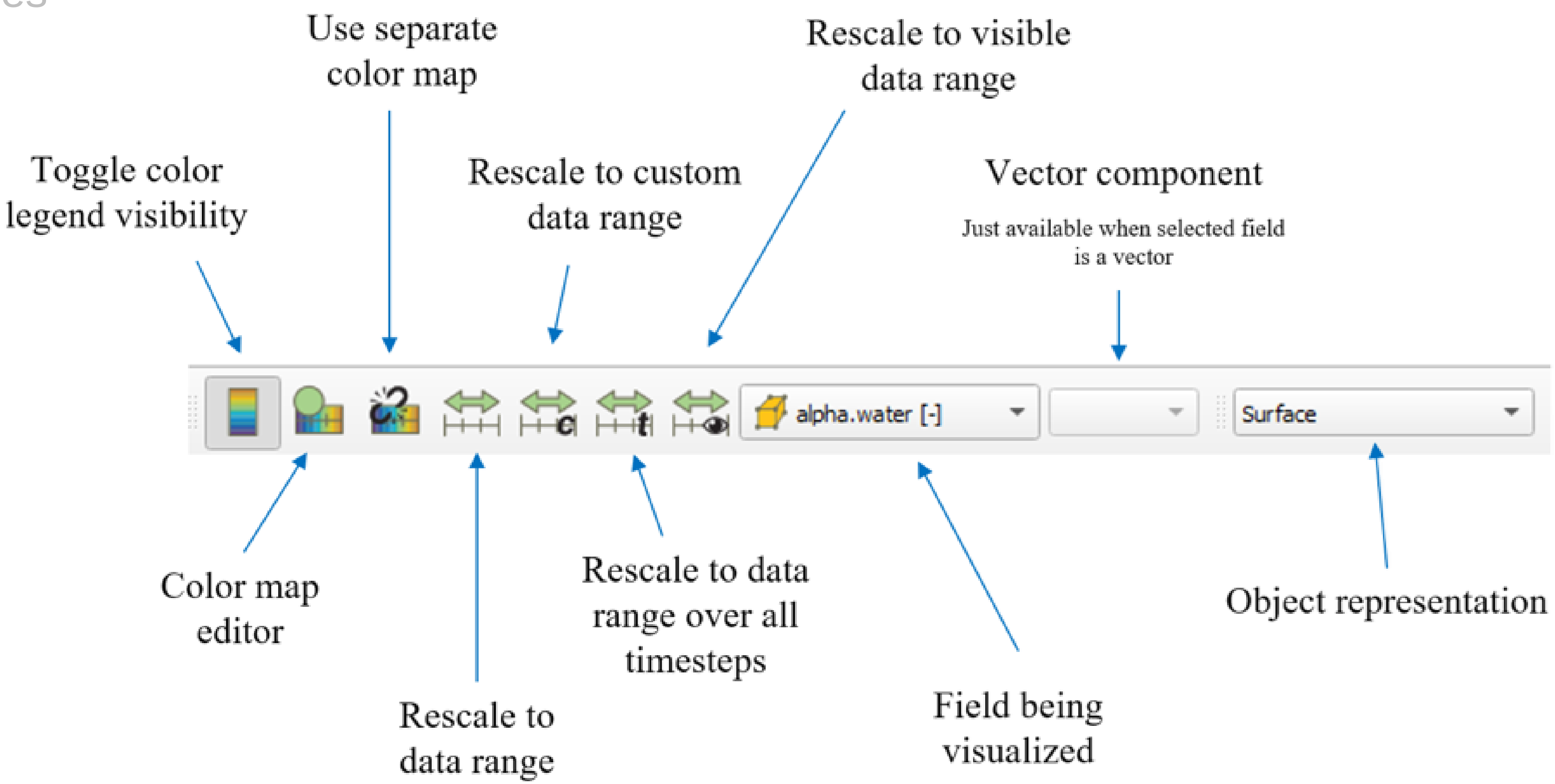



# Post-processing – Data Visualizing

1. Displaying scalar and vector fields
2. Adjusting colors, data range and labels
3. Applying Common Filters
4. Select cells and show values
5. Plot over line
6. Probe location
7. Plot selection over time

# Post-processing – Data Visualizing

1. Displaying scalar and vector fields
2. Adjusting colors, data range and labels
3. Applying Common Filters
4. Select cells and show values
5. Plot over line
6. Probe location
7. Plot selection over time

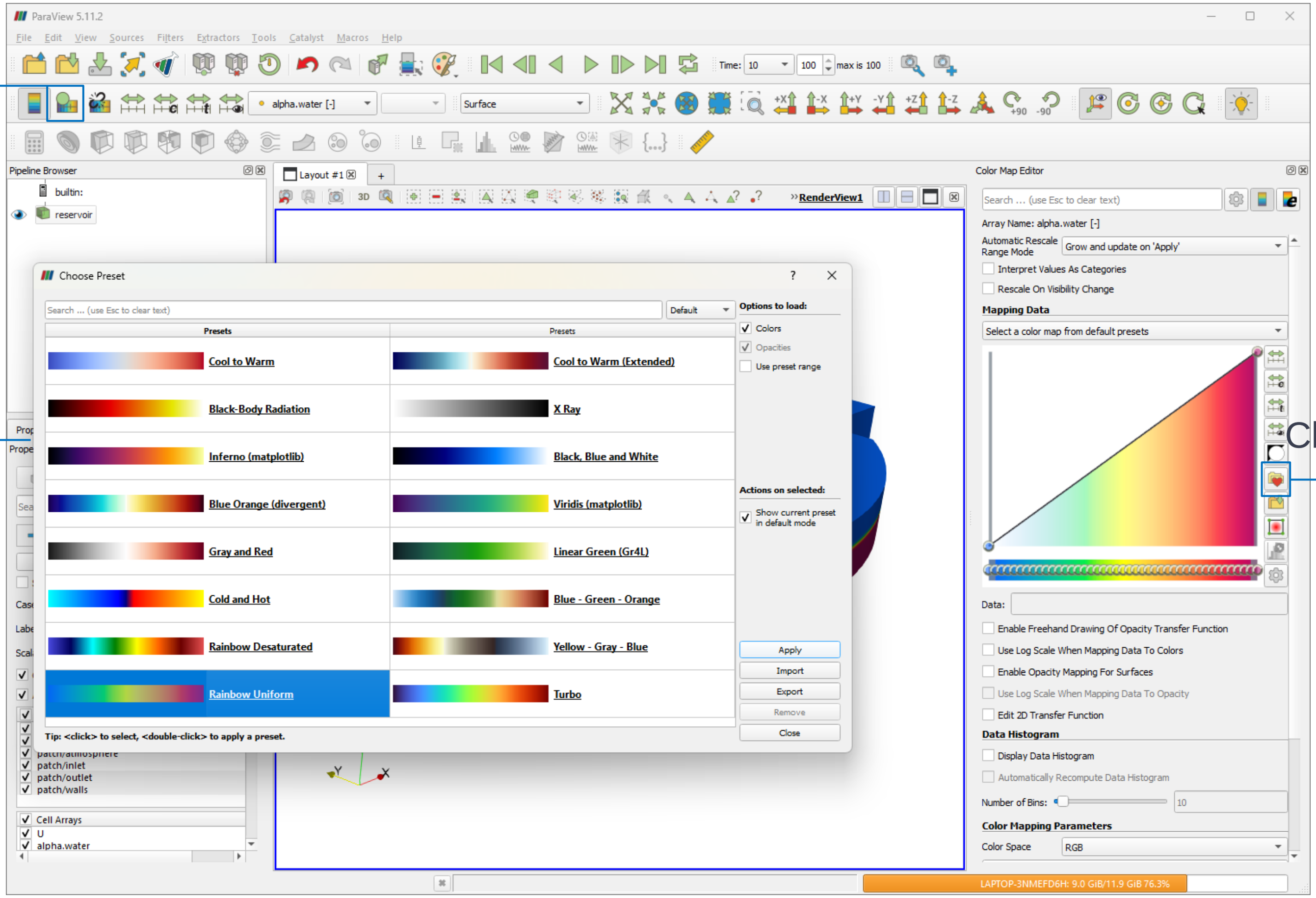


# Post-processing – Data Visualizing

1. Displaying scalar and vector fields
2. Adjusting colors, data range and labels
3. Applying Common Filters
4. Select cells and show values
5. Plot over line
6. Probe location
7. Plot selection over time

1 – click on
Edit color map

3 – select the
color map and
click Apply




2 – click on
Choose Preset

CRheo@IPC

iPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

IPPD Institute of Polymer Processing and Digital Transformation

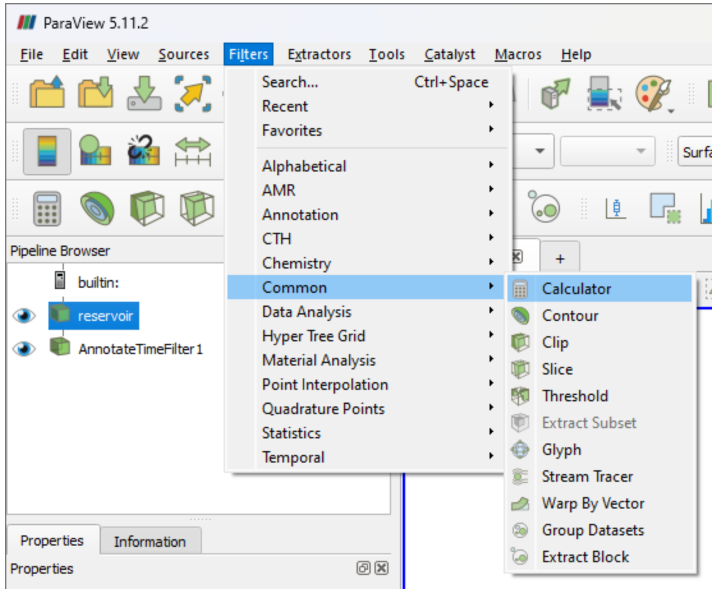
# Post-processing – Data Visualizing

1. Displaying scalar and vector fields
2. Adjusting colors, data range and labels
3. **Applying Common Filters**
4. Select cells and show values
5. Plot over line
6. Probe location
7. Plot selection over time

- Select the reservoir object in the pipeline browser

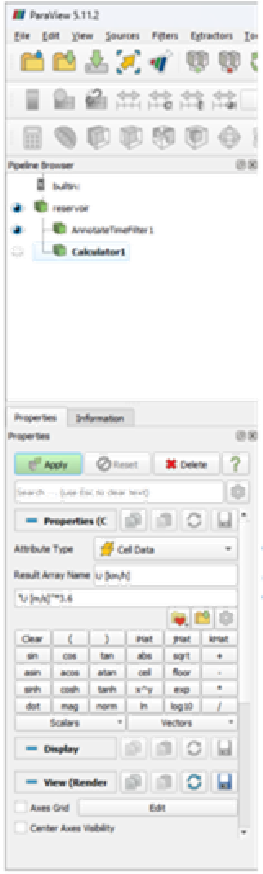
- Select the Calculator filter in the menu bar

Filters > Common > Calculator




## Calculator

Creates an variable field according to an expression



Define if it is point data or cell data

Name – U [km/h]

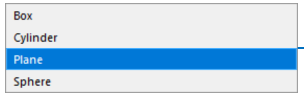
Equation

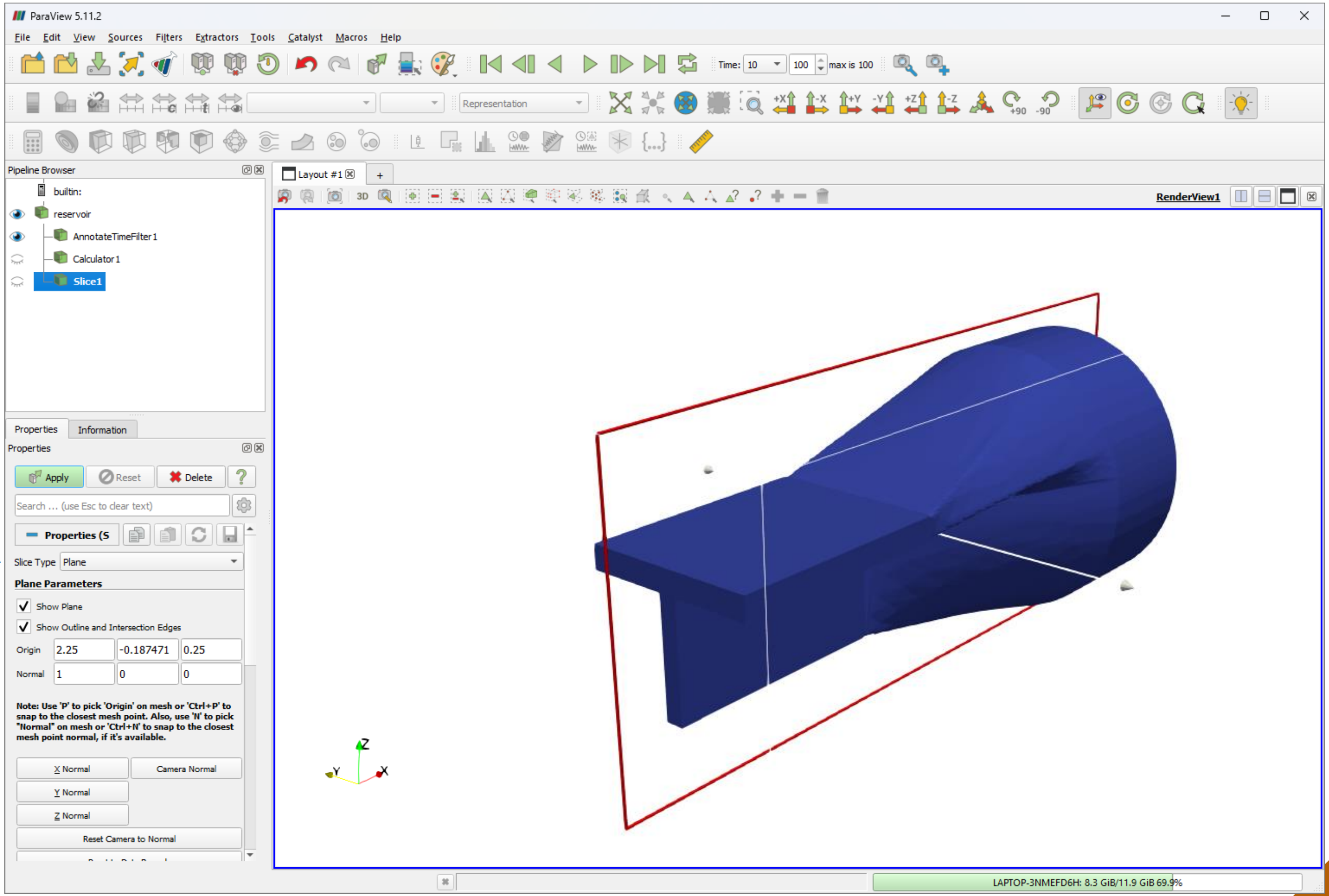
# Post-processing – Data Visualizing

1. Displaying scalar and vector fields
2. Adjusting colors, data range and labels
3. **Applying Common Filters**
4. Select cells and show values
5. Plot over line
6. Probe location
7. Plot selection over time

## Slice

Slices the object in a Box, Cylinder, Plane or Sphere

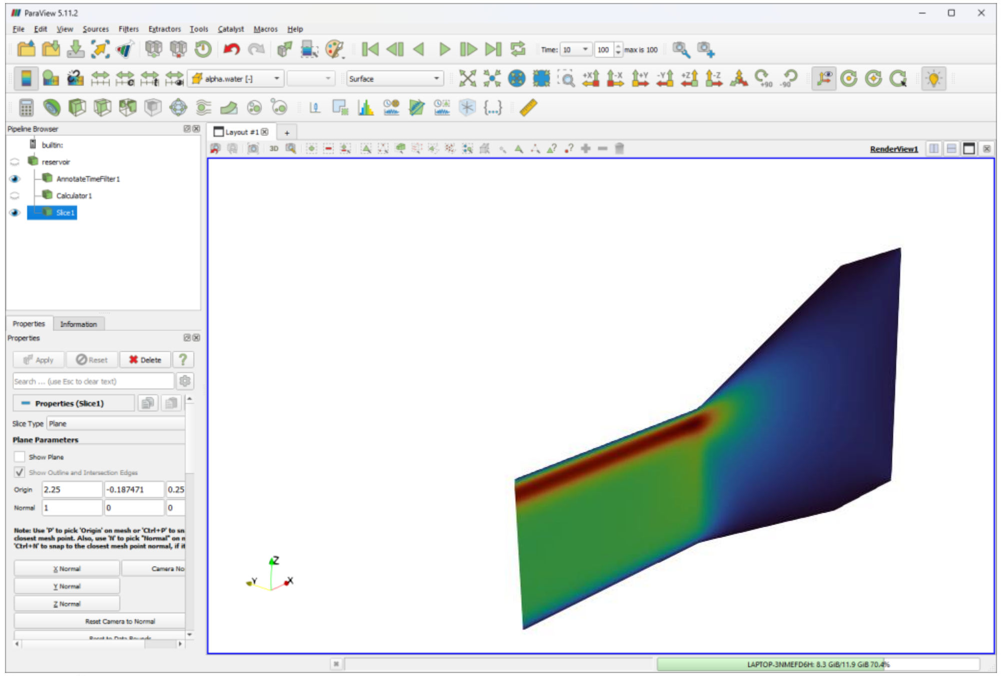



# Post-processing – Data Visualizing

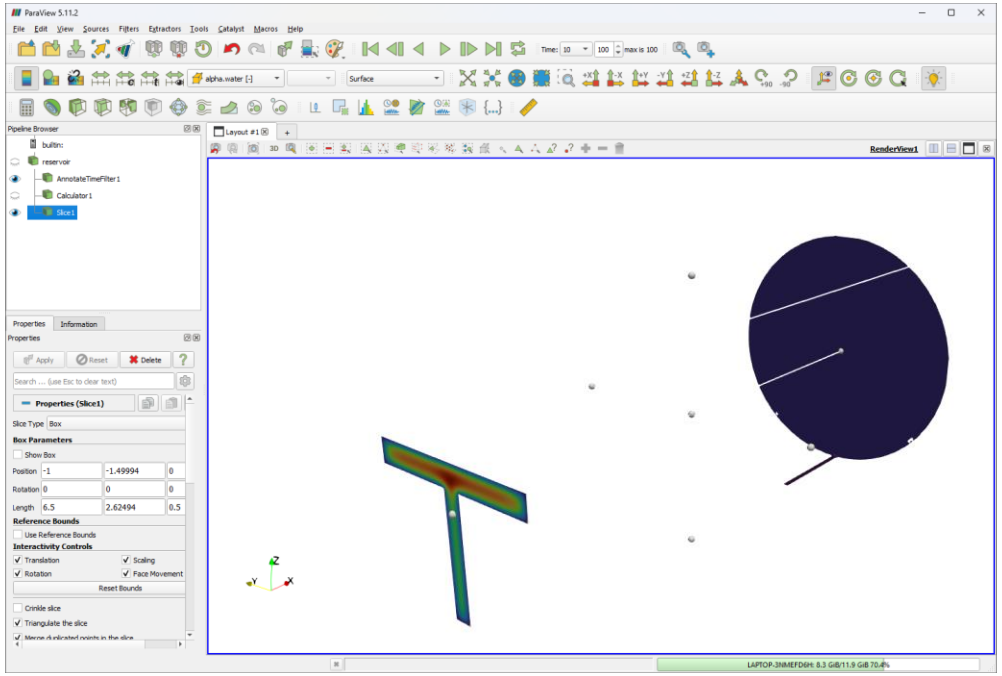
1. Displaying scalar and vector fields
2. Adjusting colors, data range and labels
3. **Applying Common Filters**
4. Select cells and show values
5. Plot over line
6. Probe location
7. Plot selection over time

## Slice

Slice - Plane



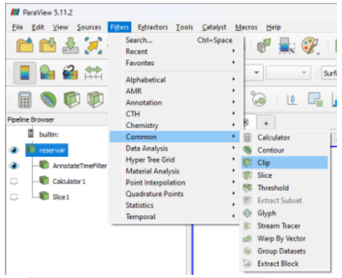
Slice - Box

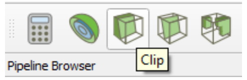


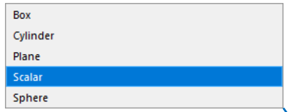
# Post-processing – Data Visualizing

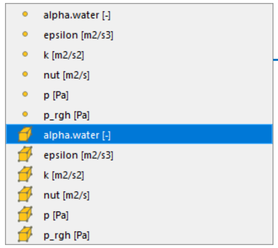
1. Displaying scalar and vector fields
2. Adjusting colors, data range and labels
3. **Applying Common Filters**
4. Select cells and show values
5. Plot over line
6. Probe location
7. Plot selection over time

## Clip

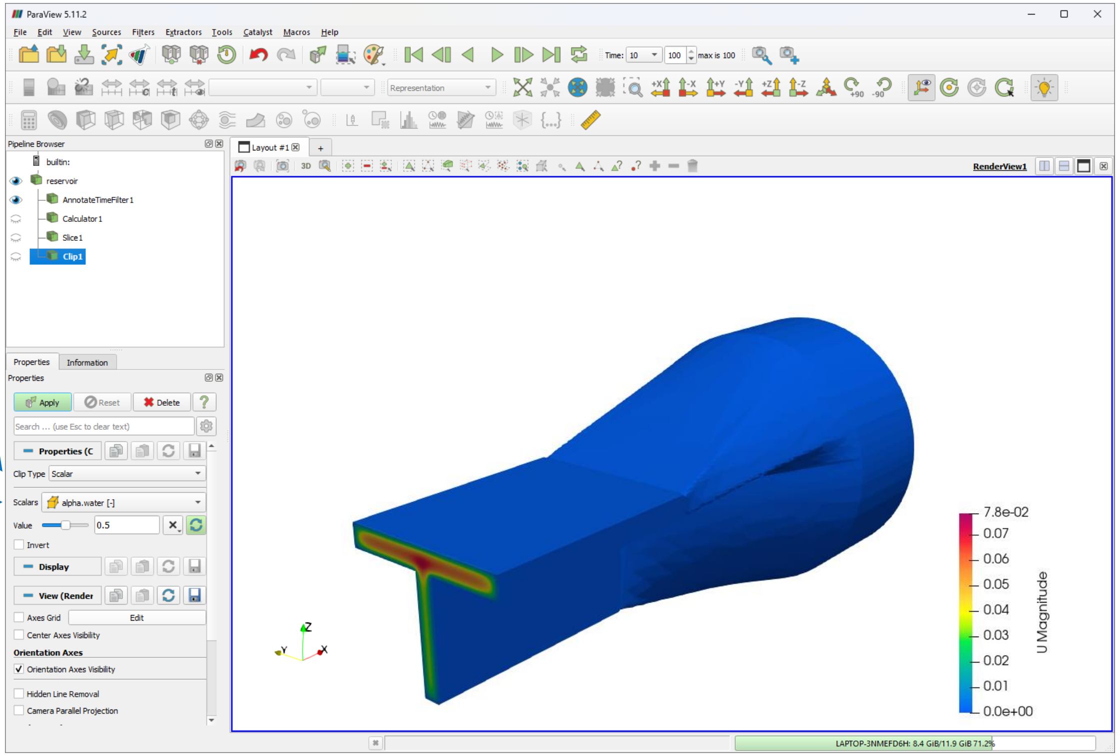





Clips the object's field greater or lower a defined value




CRheo@IPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

IPPD Institute of Polymer Processing and Digital Transformation

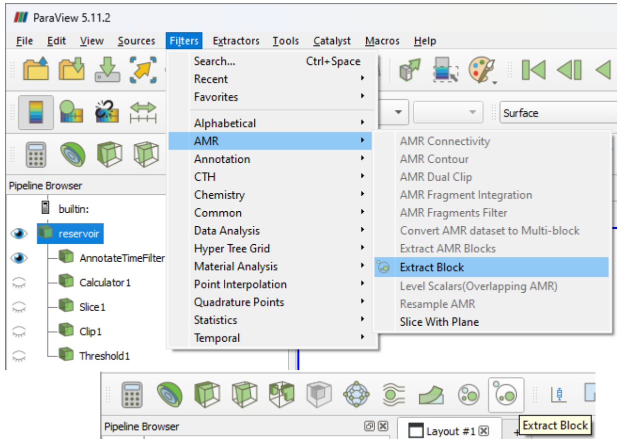
# Post-processing – Data Visualizing

1. Displaying scalar and vector fields
2. Adjusting colors, data range and labels
3. **Applying Common Filters**
4. Select cells and show values
5. Plot over line
6. Probe location
7. Plot selection over time

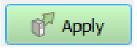
## Extract block

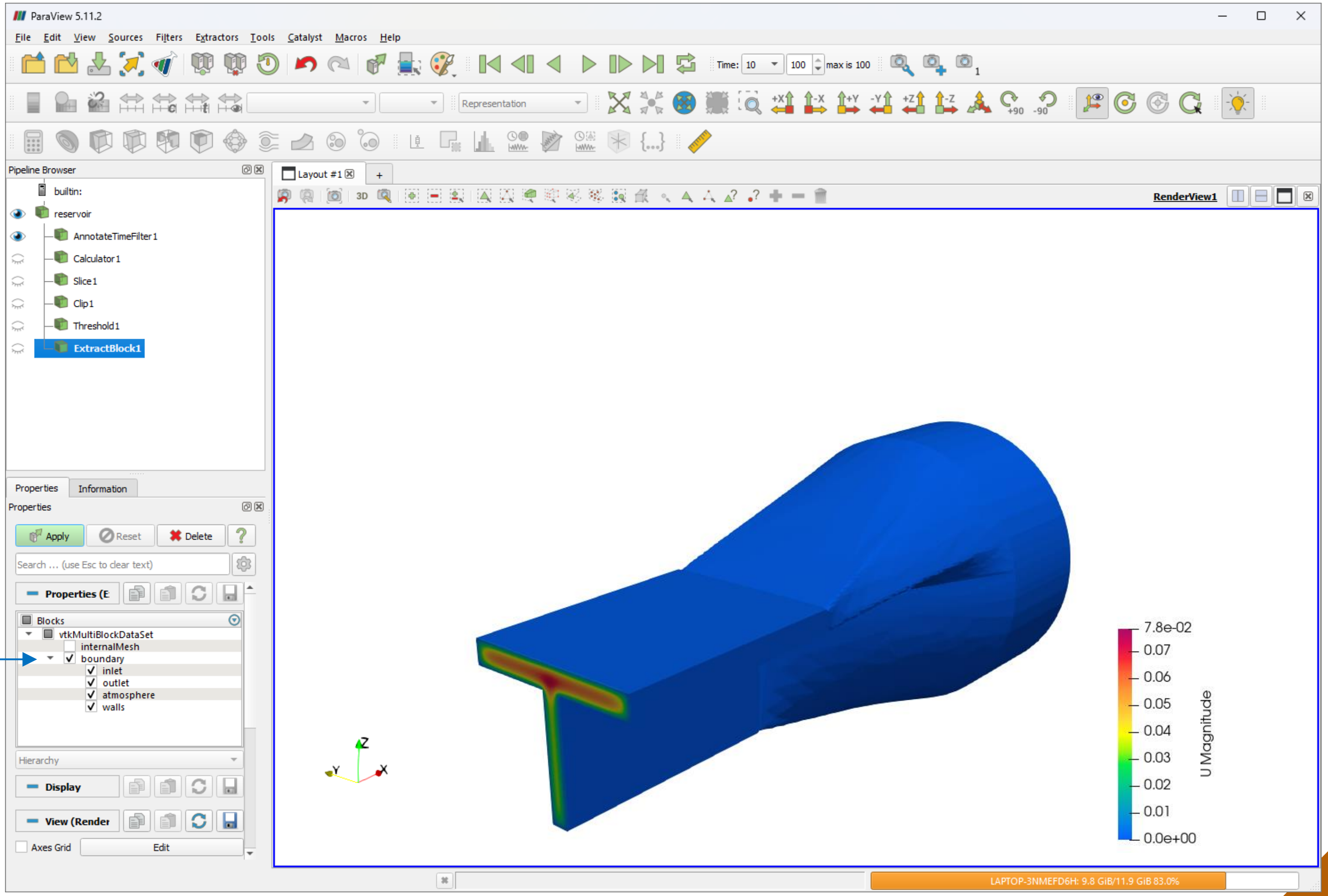
Extracting blocks allows you to obtain different objects from the same object.

It is particularly helpfull to analyse surface, and if you want to apply filter to the internal mesh and visualize surfaces.




Select boundary and click




CRheo@IPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

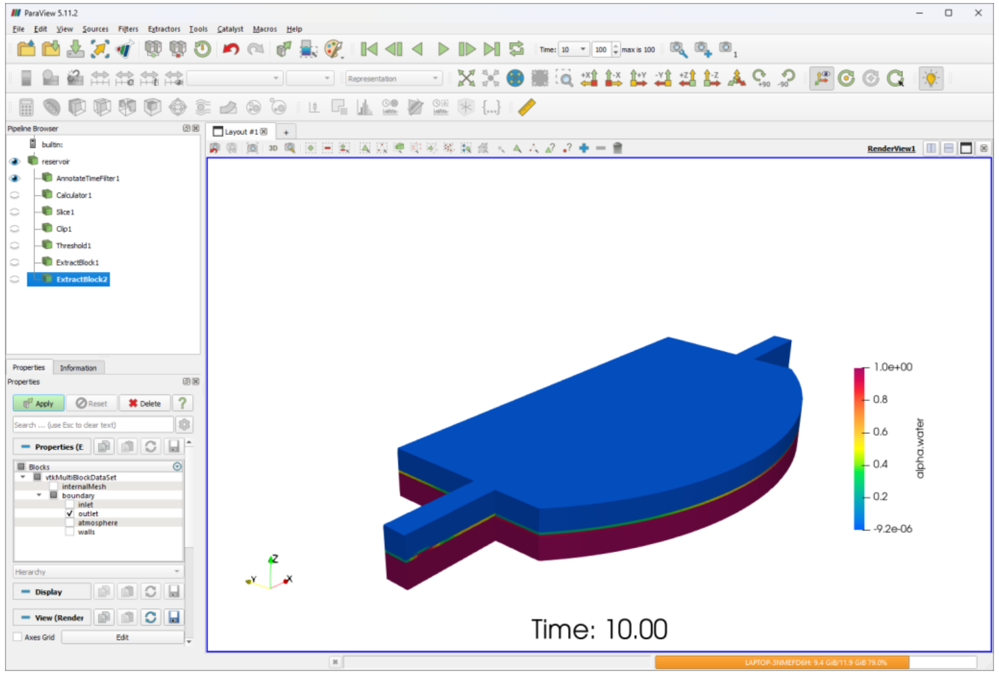
IPPD Institute of Polymer Processing and Digital Transformation

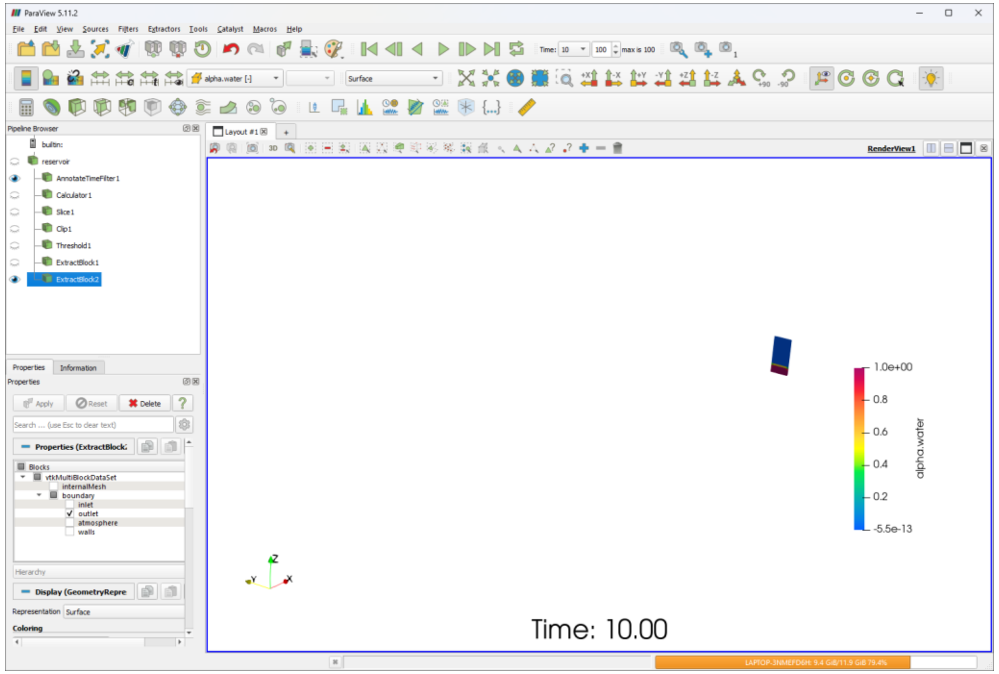
# Post-processing – Data Visualizing

1. Displaying scalar and vector fields
2. Adjusting colors, data range and labels
3. **Applying Common Filters**
4. Select cells and show values
5. Plot over line
6. Probe location
7. Plot selection over time

## Extract block

Create new extract block just with the Outlet patch to analyze it



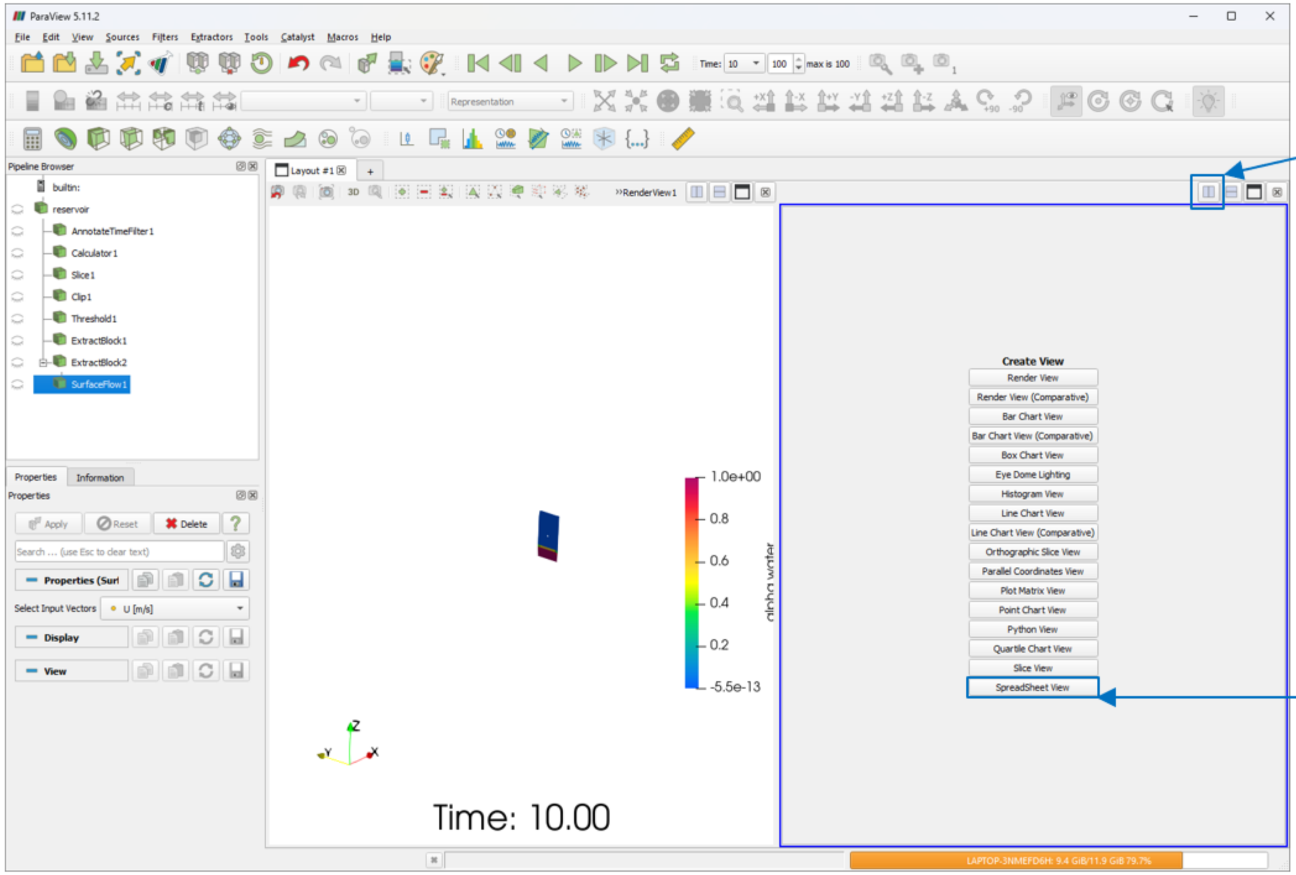



# Post-processing – Data Visualizing

1. Displaying scalar and vector fields
2. Adjusting colors, data range and labels
3. Applying Common Filters
4. Select cells and show values
5. Plot over line
6. Probe location
7. Plot selection over time

## Extract block

Use the Surface Flow filter in the ExtractBlock2 and open it in a SpreadSheet view




1 – split view in horizontal

2 – select SpreadSheet View

CRheo@IPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

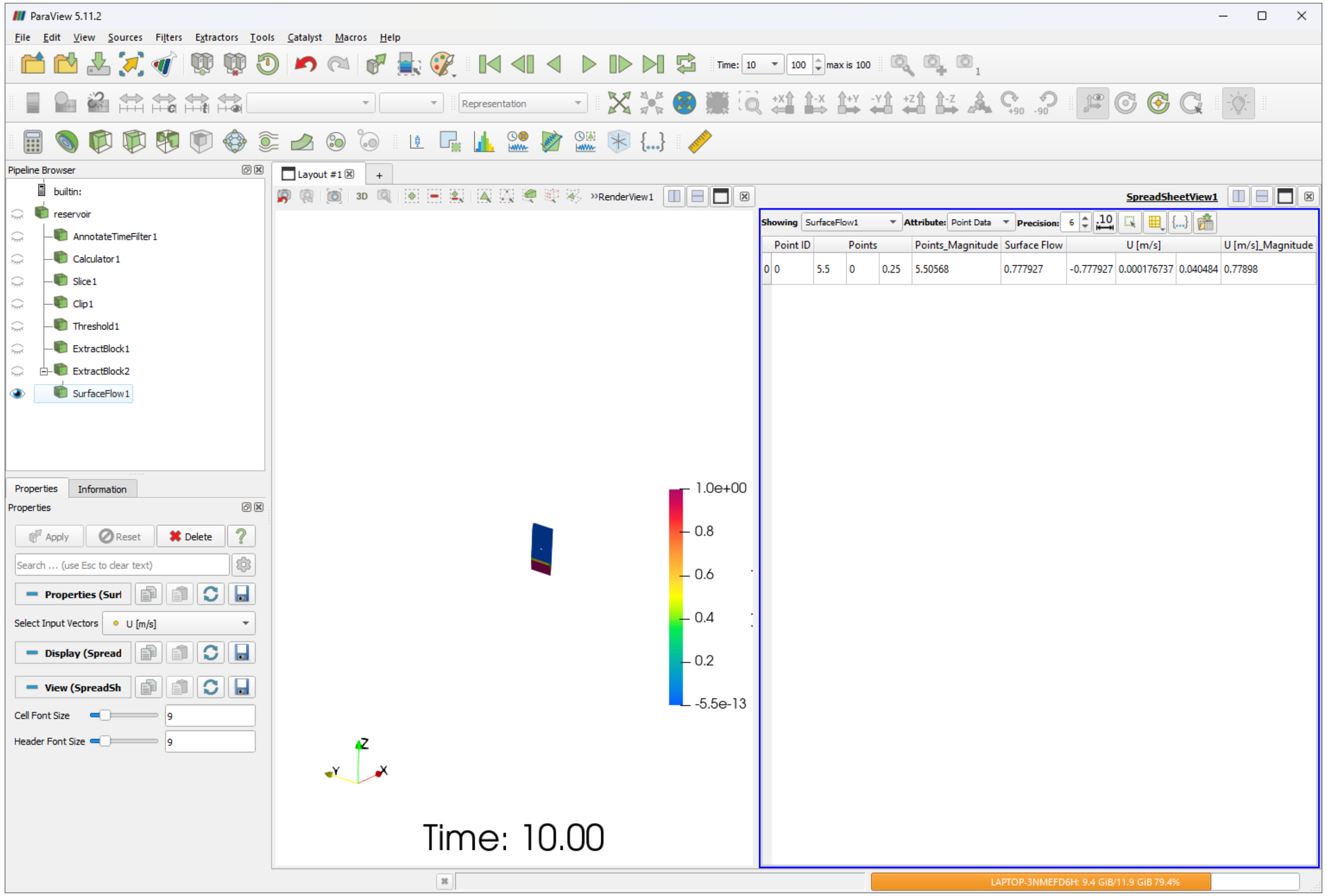
IPPD Institute of Polymer Processing and Digital Transformation

# Post-processing – Data Visualizing

1. Displaying scalar and vector fields
2. Adjusting colors, data range and labels
3. **Applying Common Filters**
4. Select cells and show values
5. Plot over line
6. Probe location
7. Plot selection over time

## Extract block

Move forward or backward in time to visualize data for all instants of time

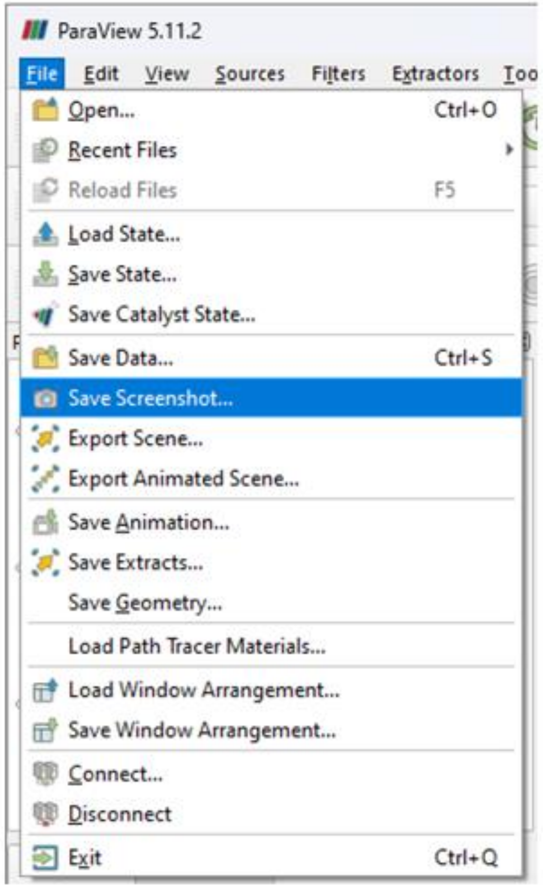





CRheo@IPC
PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain
IPPD Institute of Polymer Processing and Digital Transformation

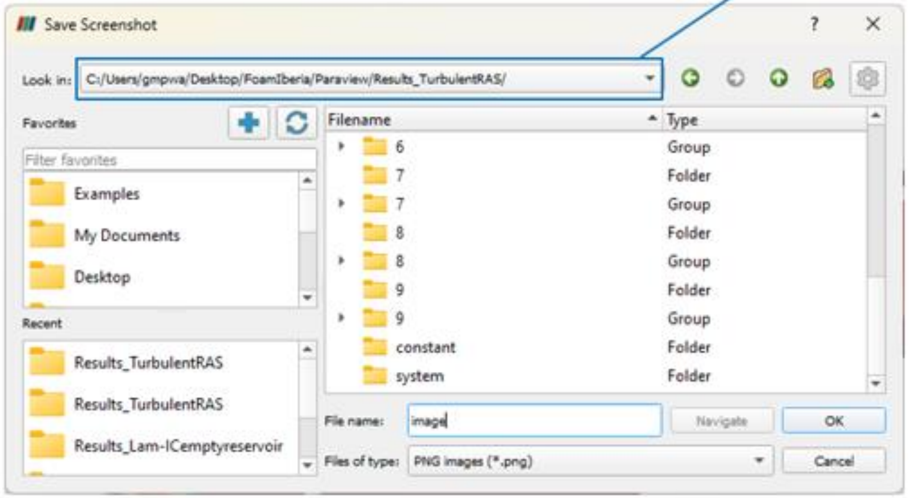
# Post-processing – Saving Progress

(1)



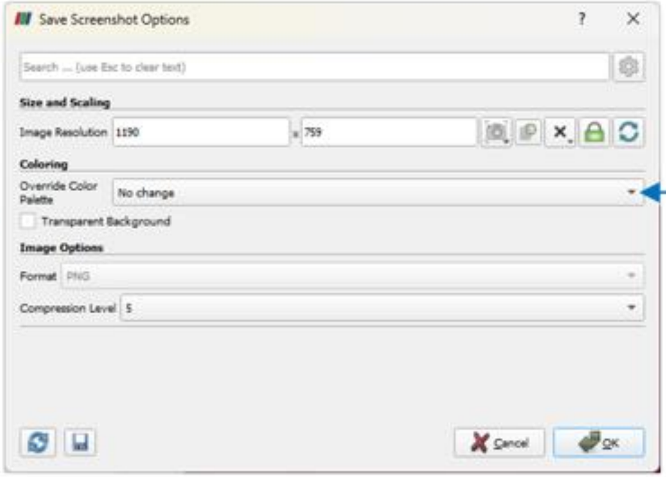

(2)

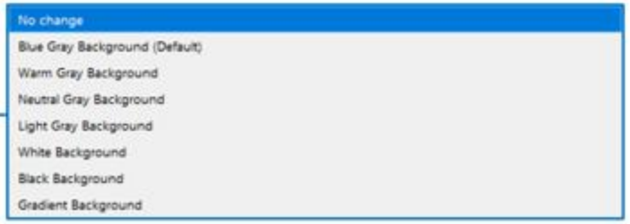
Saving location




(3)

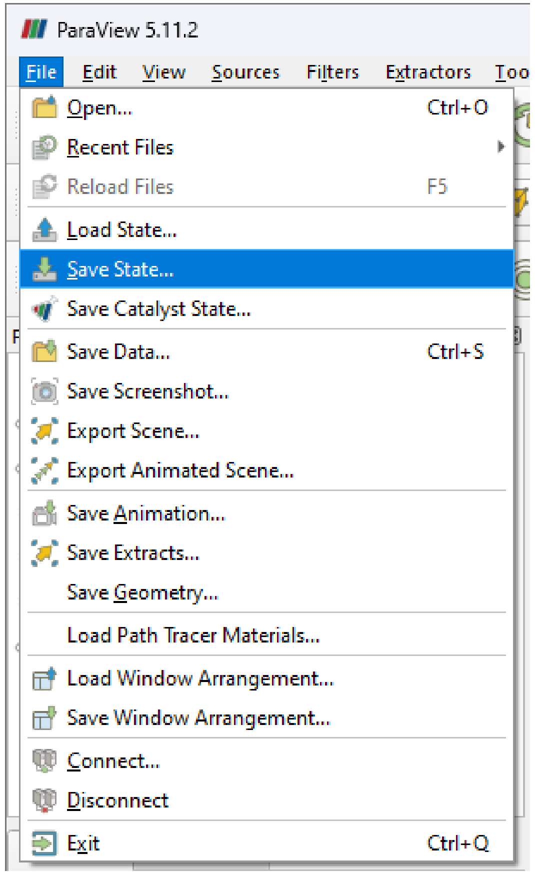
background



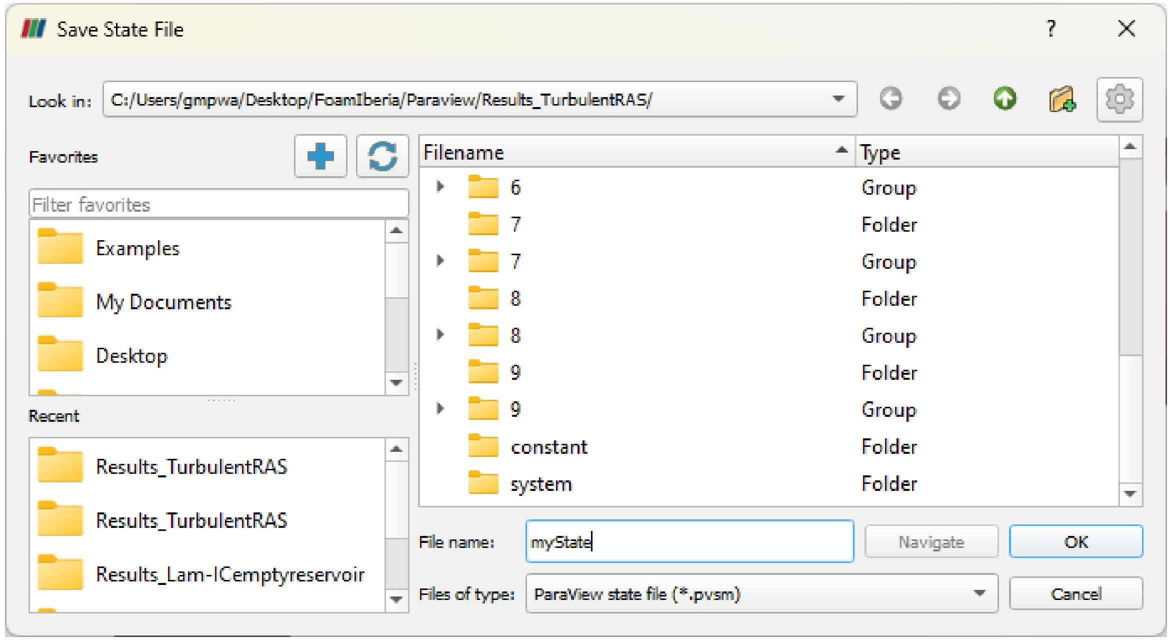



# Post-processing – Saving Progress

(1)




(2)




Saving a state allow you to reproduce your visualization settings and view without applying post-processing steps again.

It will save all your progress and it will work as a macro: once you load it, you will open your Paraview visualization the same way you saved it.

CRheo@IPC

iPC

PPS 2024
30 Sept.-03 Oct. | Ferrol · Spain

IPPD Institute of Polymer Processing and Digital Transformation