Supplement 2

Meshing with snappyHexMesh

Continuous stirring tank reactor mesh with moving regions

snappyHexMesh guided tutorials

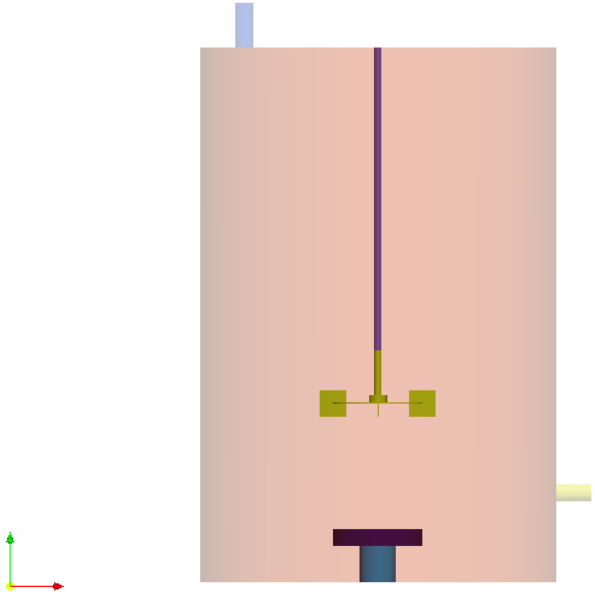
- Meshing with snappyHexMesh.
- Parallel meshing of a continuous stirring tank reactor mesh with moving regions (internal mesh)
- You will find this case in the directory:

```
$PTOFC/advanced_SHM/M2_CSTR
```

- In the case directory, you will find a few scripts with the extension .sh, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on.
- These scripts can be used to run the case automatically by typing in the terminal, for example,
  - `$> sh run_solver`
- These scripts are human-readable, and we highly recommend you open them, get familiar with the steps, and type the commands in the terminal. In this way, you will get used with the command line interface and OpenFOAM commands.
- If you are already comfortable with OpenFOAM, run the cases automatically using these scripts.
- In the case directory, you will also find the `README.FIRST` file. In this file, you will find some additional comments.

snappyHexMesh guided tutorials

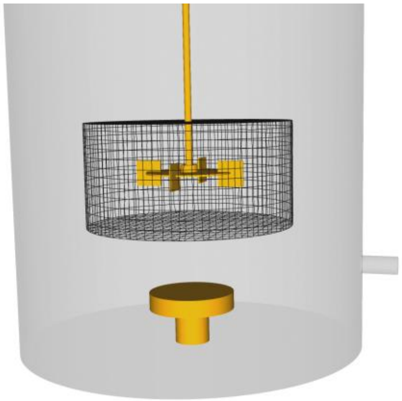
CSTR – Continuous stirring tank reactor mesh



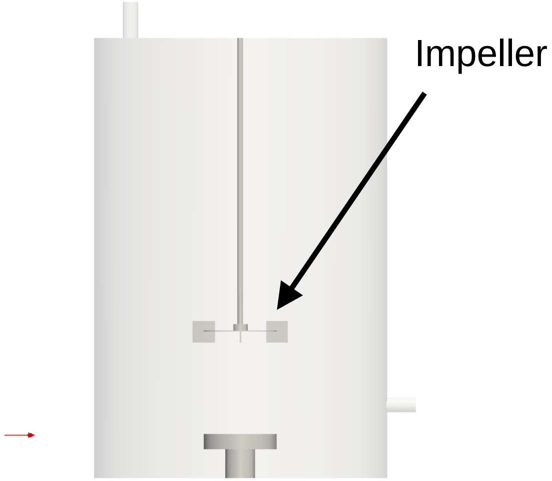
- In this case we are going to use multiple STL and eMesh files.
- Each color in the figure above represents a different STL.
- Working with multiple STL is no different from working with a single STL, we just need to read all the STLs.
- When working with multiple STL we have more control on the local refinement.

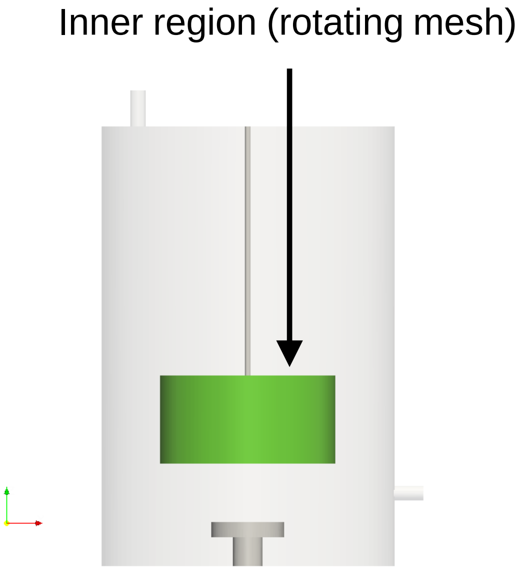
snappyHexMesh guided tutorials

CSTR – Continuous stirring tank reactor mesh



http://www.wolfdynamics.com/training/meshing/image5.gif





- We are going to work with sliding grids (the impeller will be rotating), therefore we need to divide the mesh in two regions, one fix region and one rotating region.
- To split the mesh in two regions we are going to use another STL file (the green surface), plus a few utilities to manipulate the mesh.
- We will show how to setup conforming patches between regions.

CSTR – Continuous stirring tank reactor mesh

- In this case we are going to generate a body fitted mesh with two regions and using multiple STL files.
- For simulation purposes, one of the regions will be in motion.
- This is an internal mesh.
- These are the dictionaries and files that will be used.
  - *system/snappyHexMeshDict*
  - *system/meshQualityDict*
  - *system/surfaceFeatureExtractDict*
  - *system/decomposeParDict*
  - *system/blockMeshDict*
  - *constant/triSurface/impeller.stl*
  - *constant/triSurface/impeller.eMesh*
  - *constant/triSurface/inner_volume.stl*
  - *constant/triSurface/inner_volume.eMesh*
  - *constant/triSurface/shaft.stl*
  - *constant/triSurface/shaft.eMesh*
  - *constant/triSurface/sparger.stl*
  - *constant/triSurface/sparger.eMesh*
  - *constant/triSurface/vesel.stl*
  - *constant/triSurface/vesel.eMesh*

snappyHexMesh guided tutorials

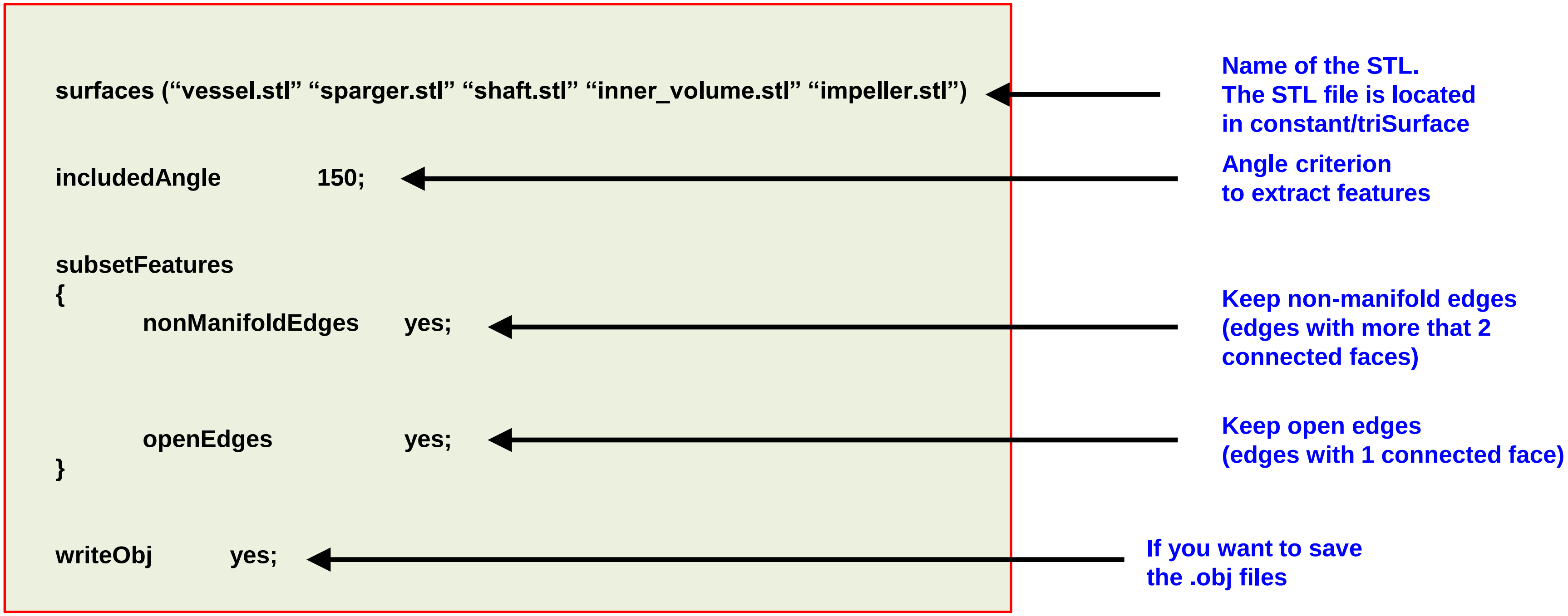
CSTR – Continuous stirring tank reactor mesh

- At this point, we are going to work in parallel.
- To generate the mesh, in the terminal window type:

```
1.  $> foamCleanTutorials
2.  $> surfaceFeatures
3.  $> blockMesh
4.  $> decomposePar
5.  $> mpirun -np 4 snappyHexMesh -parallel -overwrite
6.  $> mpirun -np 4 checkMesh -parallel -latestTime
7.  $> reconstructParMesh -constant
8.  $> paraFoam
```

CSTR – Continuous stirring tank reactor mesh

- Let us take a look at the dictionary *surfaceFeatures*.
- Notice that we are reading multiple STL files.



```
surfaces ("vessel.stl" "sparger.stl" "shaft.stl" "inner_volume.stl" "impeller.stl")

includedAngle          150;

subsetFeatures
{
          nonManifoldEdges     yes;


          openEdges            yes;
}

writeObj          yes;
```

Name of the STL.
The STL file is located
in constant/triSurface

Angle criterion
to extract features

Keep non-manifold edges
(edges with more that 2
connected faces)

Keep open edges
(edges with 1 connected face)

If you want to save
the .obj files

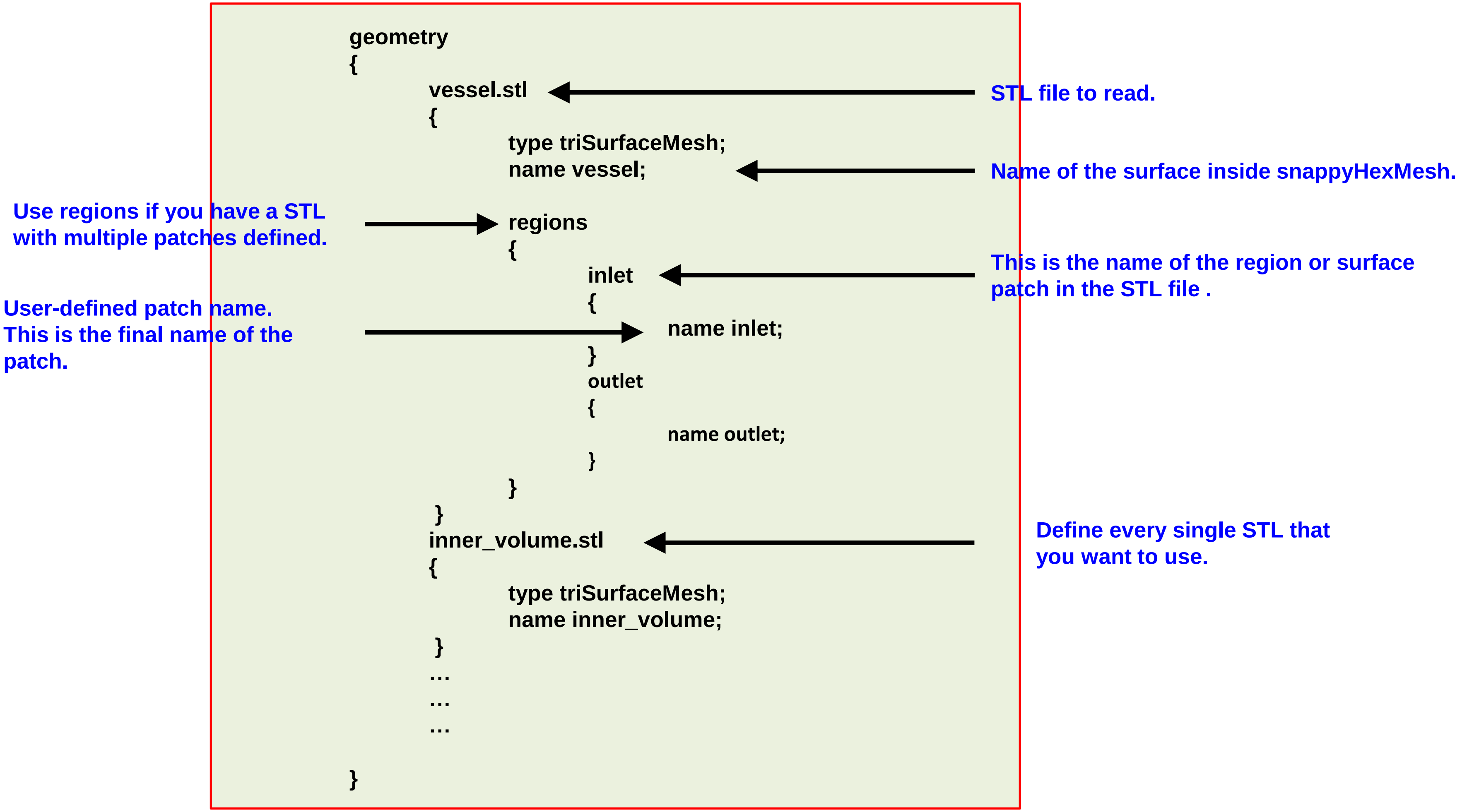
Note:
An individual eMesh file will be generated
for each individual STL. That is:

```
vessel.stl           →     vessel.eMesh
sparger.stl          →     sparger.eMesh
shaft.stl            →     shaft.eMesh
Inner_volume.stl     →     inner_volume.eMesh
impeller.stl         →     impeller.eMesh
```

snappyHexMesh guided tutorials

CSTR – Continuous stirring tank reactor mesh

- Let us take a look at the geometry section of the dictionary *snappyHexMeshDict*.
- Notice that we are reading multiple STL files.



```
geometry
{
          vessel.stl
          {
                    type triSurfaceMesh;
                    name vessel;

                    regions
                    {
                              inlet
                              {
                                        name inlet;
                              }
                              outlet
                              {
                                        name outlet;
                              }
                    }
           }
          inner_volume.stl
          {
                    type triSurfaceMesh;
                    name inner_volume;
           }
          …
          …
          …

}
```

STL file to read.

Name of the surface inside snappyHexMesh.

Use regions if you have a STL with multiple patches defined.

This is the name of the region or surface patch in the STL file .

User-defined patch name. This is the final name of the patch.

Define every single STL that you want to use.

CSTR – Continuous stirring tank reactor mesh

- Let us take a look at the **castellatedMeshControls** section of the dictionary *snappyHexMeshDict*.
- Notice that we are reading multiple eMesh files.

```
castellatedMeshControls
{
        ...
        ...
        ...

        //Explicit feature edge refinement
        features
        (
                {
                        file ”vessel.eMesh";
                        level 0;
                }
                {
                        file ”shaft.eMesh";
                        level 0;
                }
                ...
                ...
                ...

        );
}
```

Define every single eMesh file that you want to use.

Define every single eMesh file that you want to use.

CSTR – Continuous stirring tank reactor mesh

- Let us take a look at the **castellatedMeshControls** section of the dictionary *snappyHexMeshDict*.
- In this block we define the cellZone and faceZone, as follows,

```
castellatedMeshControls
{
          ...
          ...
          ...

          //Surface based refinement
          inner_volume
          (
                    level (1 1);

                    cellZone cell_inner_volume;

                    faceZone face_inner_volume;

                    cellZoneInside insidePoint;

                    insidePoint (50 0 100);
          );
          ...
          ...
          ...
}
```

Using the surface inner_volume we create a mesh zone that we will use at a later time to split the whole mesh in two regions.

Name of the cellZone.

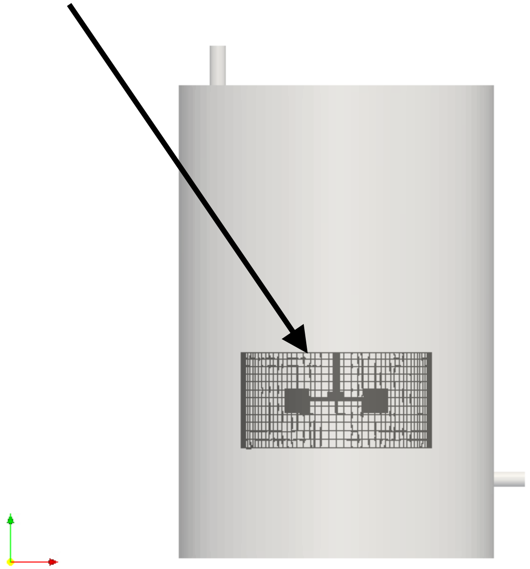
Name of the faceZone.

Use an inner point to define location of the zone

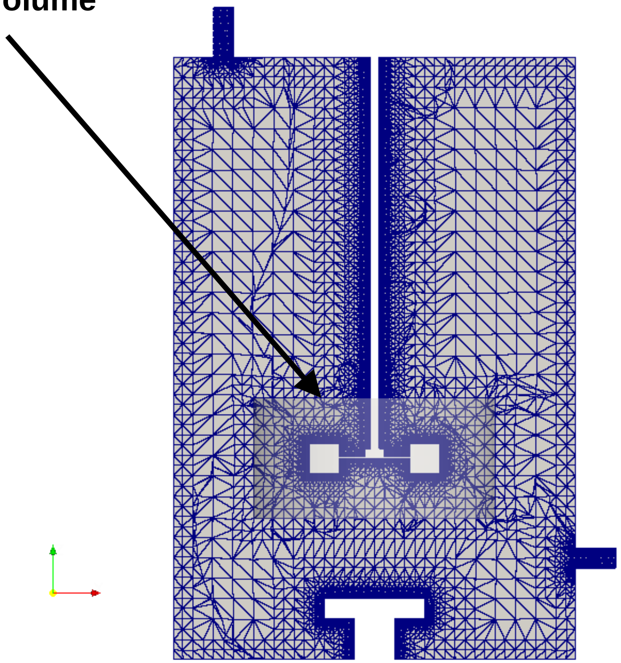
Location of the insidePoint.
The point is located inside the surface inner_volume, therefore the new zone is created inside the surface selected.

CSTR – Continuous stirring tank reactor mesh

- Using `paraFoam` let’s take a look at the newly created zone.



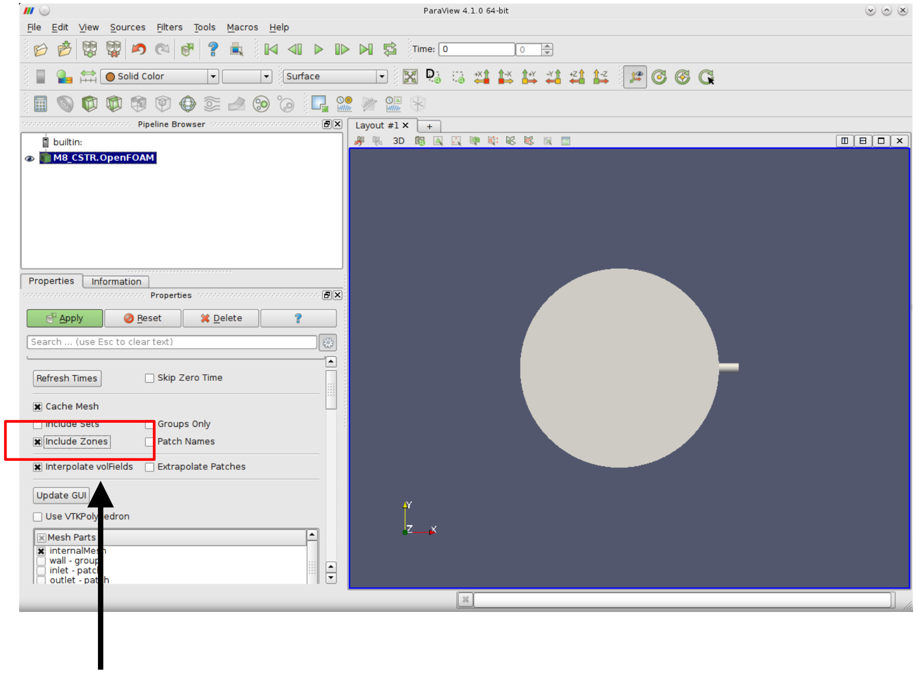
face_inner_volume



cell_inner_volume

CSTR – Continuous stirring tank reactor mesh

- To visualize the zones in `paraFoam` you will need to enable the option `Include Zones`
- Then select the mesh parts **cell_inner_volume** and **face_inner_volume**.



□ Skip Zero Time

☒ Cache Mesh

□ Include Sets □ Groups Only

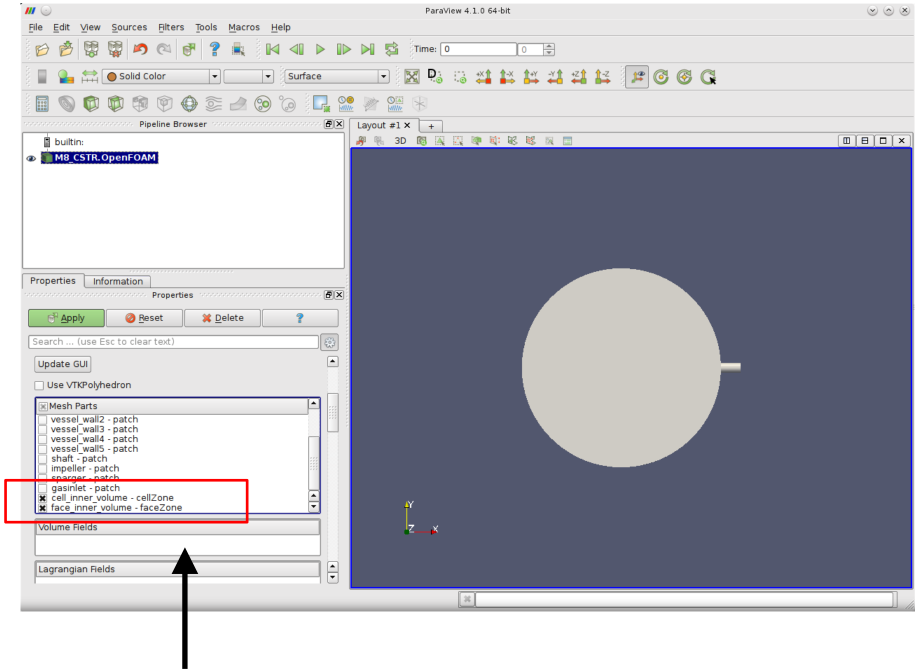
☒ Include Zones □ Patch Names

☒ Interpolate volFields □ Extrapolate Patches

□ Use VTKPolyhedron

Mesh Parts
☒ internalMesh
□ wall - group
□ inlet - patch
□ outlet - patch

1.



□ Use VTKPolyhedron

Mesh Parts
□ vessel_wall2 - patch
□ vessel_wall3 - patch
□ vessel_wall4 - patch
□ vessel_wall5 - patch
□ shaft - patch
□ impeller - patch
□ sparger - patch
□ gasinlet - patch
☒ cell_inner_volume - cellZone
☒ face_inner_volume - faceZone

2.

CSTR – Continuous stirring tank reactor mesh

- At this point and if you run `checkMesh`, you will get the following information:
  - `$> checkMesh`

```
…
…
…

Checking topology…
        Boundary definition OK.
        Cell to face addressing OK.
        Point usage OK.
        UPPER triangular ordering OK.
        Face vertices OK.
        Number of regions: 1 (OK).
…
…
…
```

- As you can see, we only have one region, but we are interested in having two regions.

CSTR – Continuous stirring tank reactor mesh

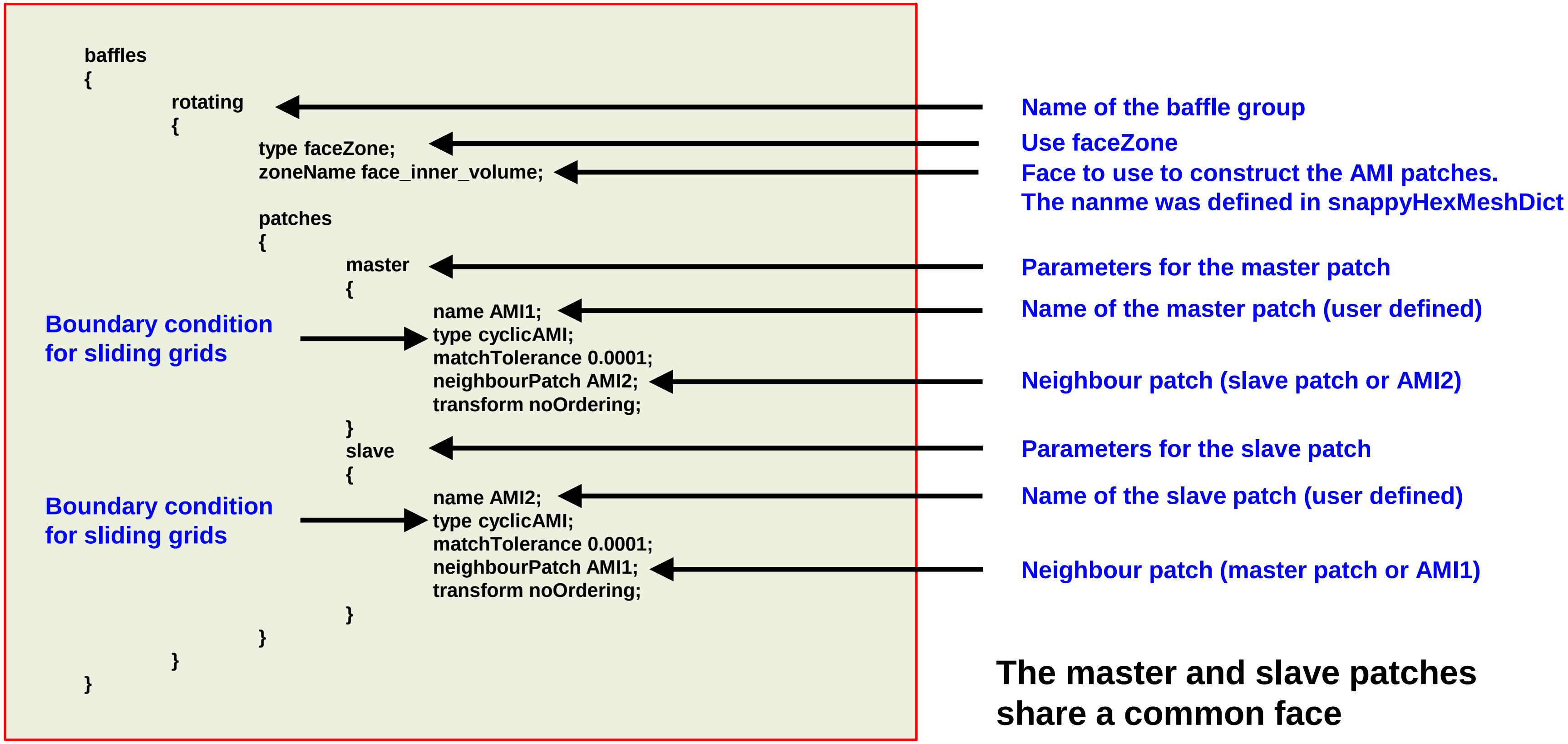
- So far, we only generated the mesh.
- The next step will consist in splitting the mesh in two regions.
- Let us now create the two regions.
- We will use the following dictionaries and files:

  - `system/createBafflesDict`
  - `system/createPatchDict`
  - `system/topoSetDict`

- The utility `createBaffles`, reads the dictionary *`createBafflesDict`*.
- The utility `createPatch`, reads the dictionary *`createPatchDict`*.
- The utility `topoSet`, reads the dictionary *`topoSetDict`*.

CSTR – Continuous stirring tank reactor mesh

- The utility `createBaffles`, reads the dictionary *`createBafflesDict`*.
- With this utility we create the interface patches between the fix zone and the rotating zone.



```
baffles
{
    rotating
    {
        type faceZone;
        zoneName face_inner_volume;

        patches
        {
            master
            {
                name AMI1;
                type cyclicAMI;
                matchTolerance 0.0001;
                neighbourPatch AMI2;
                transform noOrdering;
            }
            slave
            {
                name AMI2;
                type cyclicAMI;
                matchTolerance 0.0001;
                neighbourPatch AMI1;
                transform noOrdering;

            }
        }
    }
}
```

Boundary condition for sliding grids

Boundary condition for sliding grids

Name of the baffle group

Use faceZone

Face to use to construct the AMI patches. The nanme was defined in snappyHexMeshDict

Parameters for the master patch

Name of the master patch (user defined)

Neighbour patch (slave patch or AMI2)

Parameters for the slave patch

Name of the slave patch (user defined)

Neighbour patch (master patch or AMI1)

The master and slave patches share a common face

snappyHexMesh guided tutorials

CSTR – Continuous stirring tank reactor mesh

- To create the two regions, we proceed as follows (notice that we are going to work in serial from now on)

```
1.  $> createBaffles -overwrite
2.  $> splitBaffles -overwrite
3.  $> createPatch -overwrite
4.  $> splitMeshRegions -makeCellZones -overwrite
5.  $> splitMeshRegions -detectOnly
6.  $> transformPoints -scale ‘(0.01 0.01 0.01)’
```

- Steps 3-6 are optional.

snappyHexMesh guided tutorials

CSTR – Continuous stirring tank reactor mesh

- So, what did we do?
    - Step 1:
        - Splits the mesh in regions using the baffles (**faceZone**), created during the meshing stage.
        - We also create the **cyclicAMI** patches **AMI1** and **AMI2**.
        - At this point we have two regions and one zone. However, the two regions are stich together via the patches **AMI1** and **AMI2**.
    - Step 2: topologically split the patches **AMI1** and **AMI2**. As we removed the link between **AMI1** and **AMI2**, the regions are free to move.
    - Step 3 (optional): gets rid of zero faced patches if hey exist. These are the patches remaining from the base mesh, as they are empty, we do not need them.
    - Step 4 (optional):
        - Splits mesh into multiple zones. It will create automatically the sets and zones.
        - At this point we have two regions and two zones.
    - Step 5 (optional): just to show the regions and names.
    - Step 6 (optional): scales the mesh.

snappyHexMesh guided tutorials

CSTR – Continuous stirring tank reactor mesh

- At this point and if you run `checkMesh`, you will get the following information:
  - `$> checkMesh`

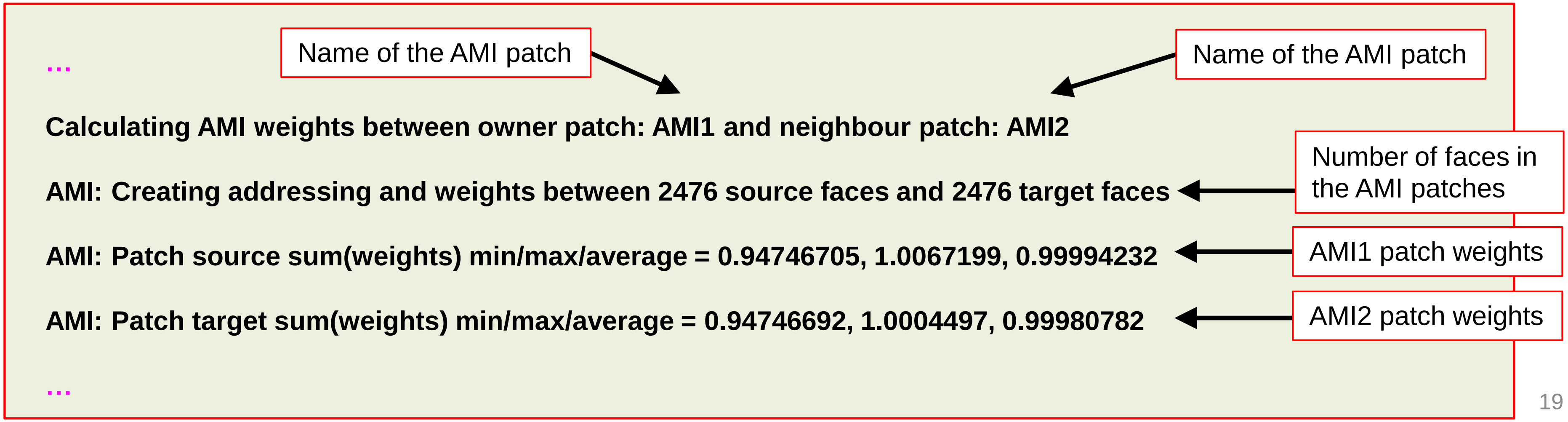
```
…
…
…
Checking topology…
          Boundary definition OK.
          Cell to face addressing OK.
          Point usage OK.
          UPPER triangular ordering OK.
          Face vertices OK.
          *Number of regions: 2
          The mesh has multiple regions which are not connected by any face.
          <<Writing region information to ”0/cellToRegion”
          <<Writing region 0 with 136187 cells to cellSet region0
          <<Writing region 1 with 67682 cells to cellSet region1
…
…
…
```

- As you can see, we now have two regions.
- At this point the mesh is ready to use.
- You can visualize the mesh (with all the sets and zones) using `paraFoam`.

snappyHexMesh guided tutorials

CSTR – Continuous stirring tank reactor mesh

- At this point the mesh is ready to use. You can visualize the mesh using `paraFoam`.
- If you use `checkMesh`, it will report that there are two regions.
- In the dictionary *`constant/dynamicsMeshDict`* we set which region will move and the rotation parameters.
- To preview the region motion, in the terminal type:
  - `$> moveDynamicMesh -checkAMI -noFunctionObjects`
- The command `moveDynamicMesh -checkAMI` will print on screen the quality of the AMI interfaces for every time step.
- Ideally, you should get the AMI patches weights as close as possible to one.
- Weight values close to one will guarantee a good interpolation between the AMI patches.



```
...

Calculating AMI weights between owner patch: AMI1 and neighbour patch: AMI2

AMI: Creating addressing and weights between 2476 source faces and 2476 target faces

AMI: Patch source sum(weights) min/max/average = 0.94746705, 1.0067199, 0.99994232

AMI: Patch target sum(weights) min/max/average = 0.94746692, 1.0004497, 0.99980782

...
```