Nozzle CFD Tutorial:

> 💡 This tutorial teaches the user, how to generate the mesh using the BE-FORGE code and how to run the simulation using OpenFOAM 11 or 12.

▼ 1) Creating the Mesh:

The first step is to generate a mesh, the mesh is the discretized volume in which we will simulate our fluid.

Overall the meshing process is the one which takes the longest time when performing CFD.

> 💡 The meshing takes on average 80% of the project time

The reason for this is that an incorrectly setup mesh can result in so called ‘floating point errors’, the only correct way of resolving these is by improving the mesh quality. Especially when it is the first time creating a custom mesh, this will go hand in hand with plenty of re-iterations, and thus a lengthy setup time.

In OpenFOAM there are a couple of ways to mesh:

- BlockMesh (Built-in)
- SnappyHexMesh (Built-in)
- Gmsh
- Netgen
- Other 3rd party meshers

In this case, as we are working with internal flow, BlockMesh and Gmsh or Netgen are the most promising. For automation reasons, the mesh generation process through BlockMesh was included in the BE-FORGE design code.

▼ blockMesh code:

```
if export == "OF":
x_combined1 = np.concatenate((np.append(x_condi, x_c),np.app
                np.append(x_condi, x_exit), np.append(x_condi
#,np.append(x_coord,x_inlet)

y_combined1 = np.concatenate((np.append(y_coord, y_c), np.app
                np.append(y_coord, y_exit), np.append(y_coor
  #,np.append(y_coord,y_inlet)
ContourCoordOF = pd.DataFrame({"x": x_combined1 ,"y": y_coml
ContourCoordOF = ContourCoordOF.sort_values("x",ignore_inde
ContourCoordOF = ContourCoordOF.drop_duplicates()
  if Grid == "Coarse":
  ScaleF = 5/2
elif Grid == "Medium":
  ScaleF = 5/3
elif Grid == "Fine":
  ScaleF == 5/4
elif Grid == "Ultra Fine":
  ScaleF == 1

with open("Nozzle", 'w') as file:
  #file.write("convertToMeters 0.001; //all dimensions are in mm
  file.write("vertices\n(\n")
  file.write(f"({x_inlet[0]}  0  0.0005) //0\n")
  file.write(f"({x_inlet[0]} {y_inlet[0]} 0.0005) //1\n")
  file.write(f"({x_c[0]} {Dc/2} 0.0005) //2\n")
  file.write(f"({x_t[-1]} {y_t[-1]} 0.0005) //3\n")
  file.write(f"({x_t[-1]} 0  0.0005) //4\n")
  file.write(f"({x_c[0]} 0  0.0005) //5\n \n")

  file.write(f"({x_inlet[0]}  0  -0.0005) //6\n")
  file.write(f"({x_inlet[0]} {y_inlet[0]} -0.0005) //7\n")
  file.write(f"({x_c[0]} {Dc/2} -0.0005) //8\n")
```

```
file.write(f"({x_t[-1]} {y_t[-1]} -0.0005) //9\n")
file.write(f"({x_t[-1]} 0  -0.0005) //10\n")
file.write(f"({x_c[0]} 0  -0.0005) //11\n \n")

file.write(f"(0.500  {y_t[-1]} 0.0005) //12 \n")
file.write(f"(0.500  0 0.0005) //13 \n")
file.write(f"({x_t[-1]}  0.100 0.0005) //14 \n")
file.write(f"(0.500  0.100 0.0005) //15 \n")
file.write(f"(0.500  {y_t[-1]} -0.0005) //16 \n")
file.write(f"(0.500  0 -0.0005) //17 \n")
file.write(f"({x_t[-1]}  0.100 -0.0005) //18 \n")
file.write(f"(0.500  0.100 -0.0005) //19 \n")
file.write(f"(0.120 {y_t[-1]} 0.0005) //20 \n")
file.write(f"(0.120 0.100 0.0005) //21 \n")
file.write(f"(0.120 {y_t[-1]} -0.0005) //22 \n")
file.write(f"(0.120 0.100 -0.0005) //23 \n")


file.write(");\n\nblocks\n(\n")
# Example: Write a single block definition using the vertices
# Adjust this according to your actual block structure
file.write(f"    hex (0 1 2 5 6 7 8 11) ({int(150/ScaleF)} {int(25/Sc
file.write(f"    hex (5 2 3 4 11 8 9 10) ({int(150/ScaleF)} {int(250
file.write(f"    hex (4 3 12 13 10 9 16 17) ({int(150/ScaleF)} {int(2
file.write(f"    hex (3 14 15 12 9 18 19 16) ({int(120/ScaleF)} {int(
#file.write(f"    hex (9 22 23 18 3 20 21 14) ({20/ScaleF} {120/S
file.write(");\n\nedges\n(\n")
# Example: Write edges using vertex indices for curves
# Adjust this according to your actual curves
file.write("polyLine 2 3 \n ( \n")
for i in range(len(ContourCoordOF) - 1):
    file.write(f"({ContourCoordOF.iloc[i]['x']} {ContourCoordOF.
file.write(") \n")
file.write("polyLine 8 9 \n ( \n")
for i in range(len(ContourCoordOF) - 1):
    file.write(f"({ContourCoordOF.iloc[i]['x']} {ContourCoordOF.
```

```
            file.write(") \n")

            file.write(");\n\nboundary\n(\n")
            # Define boundary patches (e.g., inlet, outlet, walls)
            file.write("    inlet\n    {\n        type patch;\n        faces\n        (\n
            file.write("    outlet-1\n    {\n        type patch;\n        faces\n
            file.write("    outlet-2\n    {\n        type patch;\n        faces\n
            file.write("    nozzle\n    {\n        type wall;\n        faces\n        (\n
            file.write("    bottom\n    {\n        type symmetryPlane;\n        fac
            file.write");\n \n // ****************************************
```

```
if export == "OF_axs":
        x_combined1 = np.concatenate((np.append(x_condi, x_c),np.appen
                                np.append(x_condi, x_exit), np.append(x_condi, x_
        #,np.append(x_coord,x_inlet)

        y_combined1 = np.concatenate((np.append(y_coord, y_c), np.apper
                                np.append(y_coord, y_exit), np.append(y_coord,
          #,np.append(y_coord,y_inlet)
        ContourCoordOF = pd.DataFrame({"x": x_combined1 ,"y": y_combir
        ContourCoordOF = ContourCoordOF.sort_values("x",ignore_index=
        ContourCoordOF = ContourCoordOF.drop_duplicates()
        if Grid == "Coarse":
          ScaleF = 5/2
        elif Grid == "Medium":
          ScaleF = 5/3
        elif Grid == "Fine":
          ScaleF == 5/4
        elif Grid == "Ultra Fine":
          ScaleF == 1

        wedge_angle = 2.5 * np.pi/180
        L_ch = 0.5
        H_ch = 5 * De
```

```
with open("Nozzle_axs", 'w') as file:
    #file.write("convertToMeters 0.001; //all dimensions are in mm th
    file.write("vertices\n(\n")
    file.write(f"({x_inlet[0]}  0  0) //0\n")
    file.write(f"({x_inlet[0]} {y_inlet[0]} {y_inlet[0]*np.tan(wedge_ang
    file.write(f"({x_c[0]} {Dc/2} {y_c[0]*np.tan(wedge_angle/2)}) //2\
    file.write(f"({x_t[-1]} {y_t[-1]} {y_t[-1]*np.tan(wedge_angle/2)}) //3
    file.write(f"({x_t[-1]} 0  0) //4\n")
    file.write(f"({x_c[0]} 0  0) //5\n \n")


    file.write(f"({x_inlet[0]} {y_inlet[0]} {-1 * y_inlet[0]*np.tan(wedge_
    file.write(f"({x_c[0]} {Dc/2} {-1 * y_c[0]*np.tan(wedge_angle/2)})
    file.write(f"({x_t[-1]} {y_t[-1]} {-1 * y_t[-1]*np.tan(wedge_angle/2)

    file.write(f"({L_ch}  {y_t[-1]} {y_t[-1]*np.tan(wedge_angle/2)}) //9
    file.write(f"({L_ch}  0 0) //10 \n")
    file.write(f"({x_t[-1]}  {H_ch} {H_ch*np.tan(wedge_angle/2)}) //11
    file.write(f"({L_ch}  {H_ch} {H_ch*np.tan(wedge_angle/2)}) //12 \n
    file.write(f"({L_ch}  {y_t[-1]} {-1 * y_t[-1]*np.tan(wedge_angle/2)})

    file.write(f"({x_t[-1]}  {H_ch} {-1 * H_ch*np.tan(wedge_angle/2)} )
    file.write(f"({L_ch}  {H_ch}  {-1 * H_ch*np.tan(wedge_angle/2)}) /



    file.write(");\n\nblocks\n(\n")
    # Example: Write a single block definition using the vertices
    # Adjust this according to your actual block structure
    file.write(f"    hex (0 6 1 0 5 7 2 5) ({int(150/ScaleF)} 1 {int(25/Sca
    file.write(f"    hex (5 7 2 5 4 8 3 4) ({int(150/ScaleF)} 1 {int(250/S
    file.write(f"    hex (4 8 3 4 10 13 9 10) ({int(150/ScaleF)} 1 {int(250
    file.write(f"    hex (3 11 12 9 8 14 15 13) ({int(120/ScaleF)} {int(250
    #file.write(f"    hex (9 22 23 18 3 20 21 14) ({20/ScaleF} {120/Sca
    file.write(");\n\nedges\n(\n")
```

```
        # Example: Write edges using vertex indices for curves
        # Adjust this according to your actual curves
        file.write("polyLine 2 3 \n ( \n")
        for i in range(len(ContourCoordOF) - 1):
            file.write(f"({ContourCoordOF.iloc[i]['x']} {ContourCoordOF.ilo
        file.write(") \n")
        file.write("polyLine 7 8 \n ( \n")
        for i in range(len(ContourCoordOF) - 1):
            file.write(f"({ContourCoordOF.iloc[i]['x']} {ContourCoordOF.ilo
        file.write(") \n")

        file.write(");\n\nboundary\n(\n")
        # Define boundary patches (e.g., inlet, outlet, walls)
        file.write("    inlet\n    {\n        type patch;\n        faces\n        (\n
        file.write("    outlet-1\n    {\n        type patch;\n        faces\n        (\r
        file.write("    outlet-2\n    {\n        type patch;\n        faces\n        (\
        file.write("    nozzle\n    {\n        type wall;\n        faces\n        (\n
        file.write("    wedge1\n    {\n        type wedge;\n        faces\n        (
        file.write("    wedge2\n    {\n        type wedge;\n        faces\n        (
        file.write(");\n \n // ****************************************
```

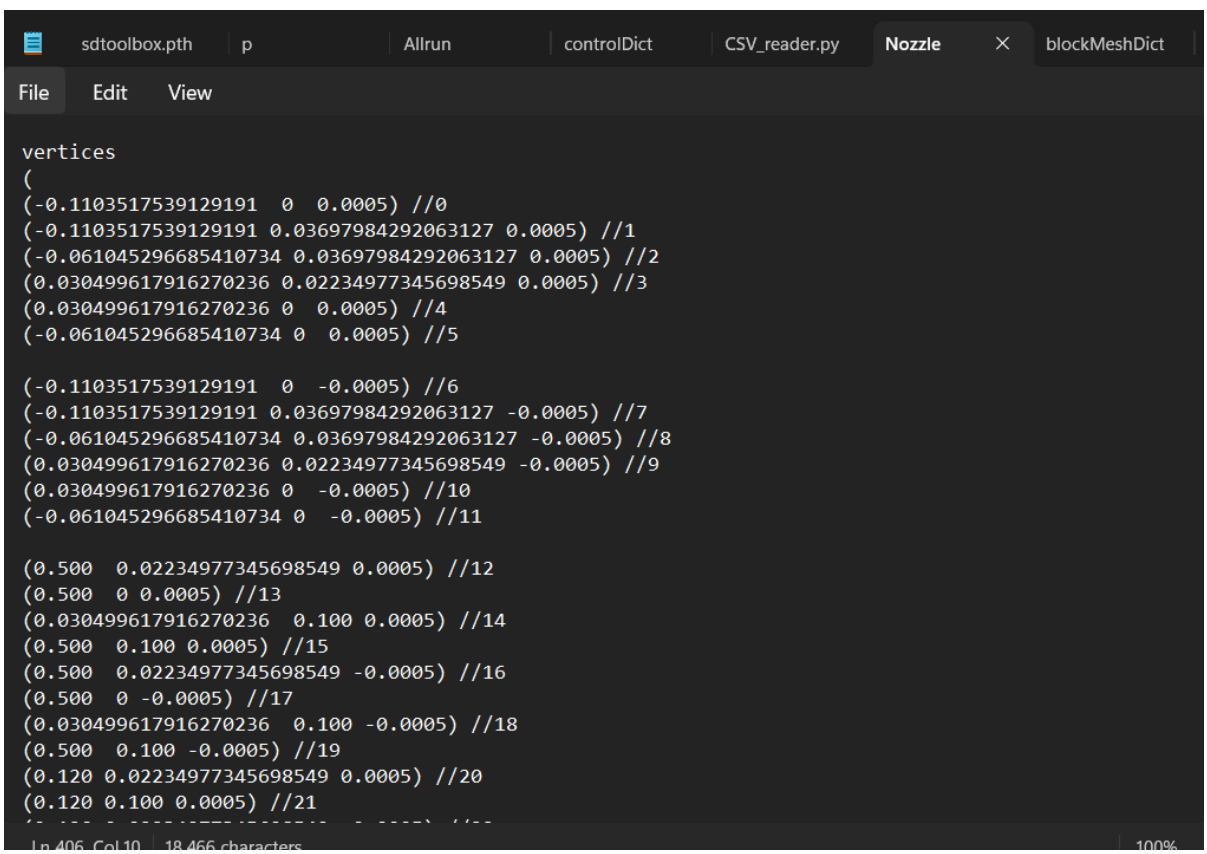
By running the following line of code in the rocket engine design script, it will compute the geometry of the nozzle, and provide the user with an OpenFOAM blockMeshDict output format

💡 There are two options available:
”OF” ⟶ this will create a 2D mesh
”OF_axs” ⟶ this will create a wedge for axisymmetric boundary conditions (recommended)

```
HRE_1_Geo = CH_Geo(HRE_1, 'Bell', 10, 0.8 ,45, 35, 5,"OF","True")
```

```
#note that there is the option to set the mesh resolution:
#By default if the user doesn't specify anything, the mesh is coarse
HRE_1_Geo = CH_Geo(HRE_1, 'Bell', 10, 0.8 ,45, 35, 5,"OF","True", Grid = 'Coar
HRE_1_Geo = CH_Geo(HRE_1, 'Bell', 10, 0.8 ,45, 35, 5,"OF","True", Grid = 'Medi
HRE_1_Geo = CH_Geo(HRE_1, 'Bell', 10, 0.8 ,45, 35, 5,"OF","True", Grid = 'Fine'
HRE_1_Geo = CH_Geo(HRE_1, 'Bell', 10, 0.8 ,45, 35, 5,"OF","True", Grid = 'Ultra

#For the axisymmetric cases:
HRE_1_Geo = CH_Geo(HRE_1, 'Bell', 10, 1.5 ,45, 35, 2,"OF_axs","True")
HRE_1_Geo = CH_Geo(HRE_1, 'Bell', 10, 1.5 ,45, 35, 5,"OF_axs","True", Grid = 'C
HRE_1_Geo = CH_Geo(HRE_1, 'Bell', 10, 1.5 ,45, 35, 5,"OF_axs","True", Grid = 'M
HRE_1_Geo = CH_Geo(HRE_1, 'Bell', 10, 1.5 ,45, 35, 5,"OF_axs","True", Grid = 'F
HRE_1_Geo = CH_Geo(HRE_1, 'Bell', 10, 1.5 ,45, 35, 5,"OF_axs","True", Grid = 'U
```

This will generate a file called ‘nozzle’ which can be used open a .txt editor such as notepad or notepad ++ .



Next, we will navigate to the ‘template case’ for the nozzle flow, which can be found at #file location# (to be added later)

Within the directory, navigate to the system folder:

Name	Date modified	Type	Size
0	15/02/2025 21:01	File folder	
constant	15/02/2025 23:39	File folder	
system	15/02/2025 20:55	File folder	
Allclean	06/03/2024 15:59	File	1 KB
Allrun	03/05/2024 11:47	File	1 KB
desktop.ini	15/02/2025 18:36	Configuration setti...	1 KB
nozzle.foam	15/02/2025 22:19	FOAM File	0 KB
probe_plotting.py	03/05/2024 17:26	Python Source File	3 KB
Spreadsheet_exit_Conditions.csv	11/05/2024 18:26	OpenOffice.org 1....	115 KB

Within the system folder, there is a file called ‘blockMeshDict’ , open this, and replace the existing code by the one generated in the nozzle file.

> ❗❗ **Do not remove the heading, as this will make the file unreadable for OpenFOAM!**

sdtoolbox.pth | p | Allrun | controlDict | CSV_reader.py | Nozzle | blockMesh

File Edit View

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  11
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

vertices
(
(-0.1103517539129191  0  0.0005) //0
(-0.1103517539129191 0.03697984292063127 0.0005) //1
(-0.061045296685410734 0.03697984292063127 0.0005) //2
(0.030499617916270236 0.02234977345698549 0.0005) //3
(0.030499617916270236 0  0.0005) //4
(-0.061045296685410734 0  0.0005) //5

(-0.1103517539129191  0  -0.0005) //6
(-0.1103517539129191 0.03697984292063127 -0.0005) //7
```

Ln 48, Col 34 | 19,035 characters | 100%

Once this is done, save the blockMeshDict.

Next open up a terminal window:

The first step is starting the WSL environment, and navigating to the run folder of your OpenFOAM installation.

```
WSL
```

```
cd $FOAM_RUN
```

```
PS C:\Users\brams> wsl
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 5.15.167.4-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Feb 16 09:21:44 CET 2025

  System load:  0.38                Processes:             59
  Usage of /:   0.4% of 1006.85GB   Users logged in:       0
  Memory usage: 6%                  IPv4 address for eth0: 172.24.162.148
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

This message is shown once a day. To disable it please create the
/home/brams/.hushlogin file.
brams@MSI:/mnt/c/Users/brams$ cd $FOAM_RUN
brams@MSI:~/OpenFOAM/brams-12/run$
```

next navigate to the case folder:

```
brams@MSI:/mnt/c/Users/brams$ cd $FOAM_RUN
brams@MSI:~/OpenFOAM/brams-12/run$ cd HRE_nozzle/
brams@MSI:~/OpenFOAM/brams-12/run/HRE_nozzle$
```

with the blockMesh code saved, now run the blockMesh command:

```
blockMesh
```

If you followed everything correctly, the following process has been executed.

```
        k : 0.001
    Block 3 cell size :
        i : 0.0001264 .. 0.006318
        j : 0.000625 .. 0.01562
        k : 0.001

No patch pairs to merge

Writing polyMesh
----------------
Mesh Information
----------------
  boundingBox: (-0.1104 0 -0.0005) (0.5 0.1 0.0005)
  nPoints: 35438
  nCells: 17400
  nFaces: 69918
  nInternalFaces: 34482
----------------
Patches
----------------
  patch 0 (start: 34482 size: 60) name: inlet
  patch 1 (start: 34542 size: 108) name: outlet-1
  patch 2 (start: 34650 size: 148) name: outlet-2
  patch 3 (start: 34798 size: 110) name: nozzle
  patch 4 (start: 34908 size: 210) name: bottom
  patch 5 (start: 35118 size: 34800) name: defaultFaces

End

brams@MSI:~/OpenFOAM/brams-12/run/HRE_nozzle$
```

Now let us check the mesh:

First make sure to create a .foam file, in order to view it within ParaView

```
#creating a .foam file can be done using the following methods
#method 1:

> nozzle.foam

#method 2:
```

```
touch nozzle.foam
```

Both the ‘>’ and ‘touch’ will create a new file with the name and extension you’ve given it.

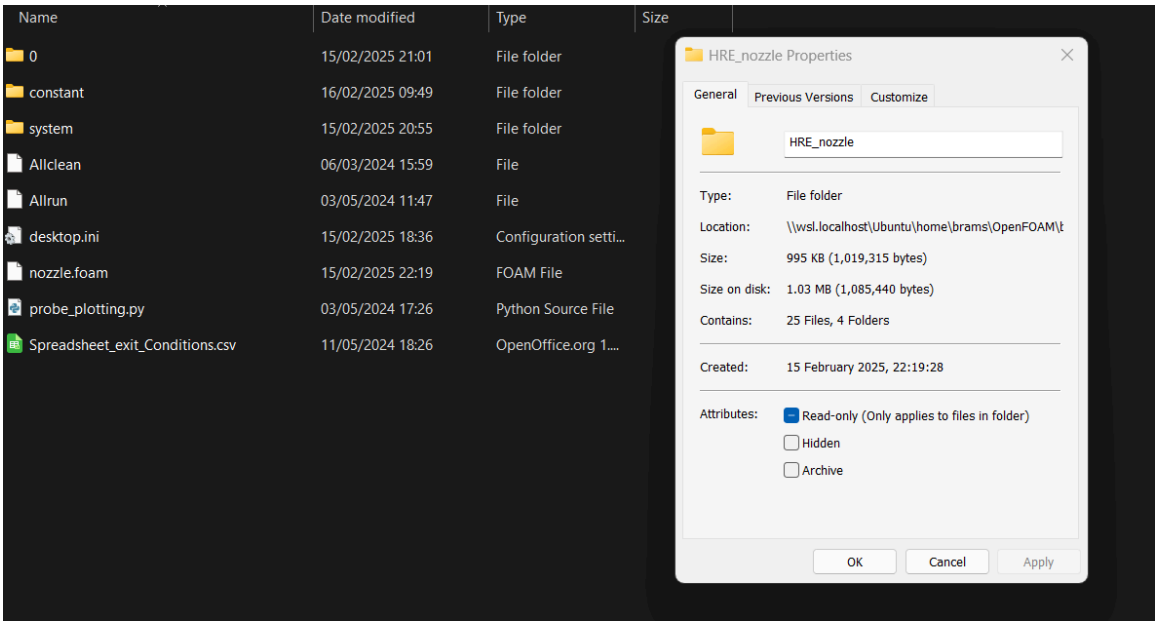
Viewing the mesh in Paraview:

▼ Quick access to the Run folder in paraview

When using paraview for the first time, it might be challenging to find your run folder, especially with the virtual linux subsystem.

A method to not having to worry about finding your run folder is presented here:

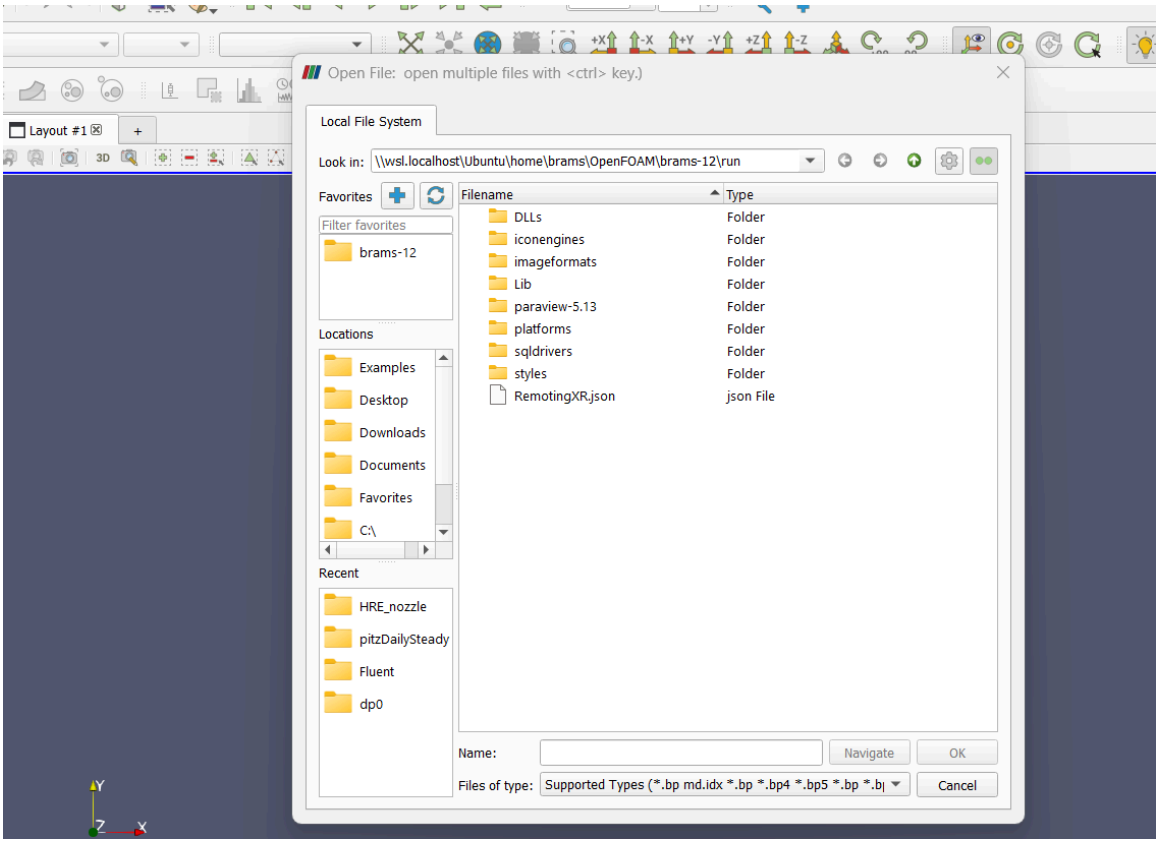
In your file manager, navigate to the run folder within your WSL installation. Once you’ve done that, open a case within the run folder, and open the properties tab. In the properties tab, you’ll now want to copy the file path location.



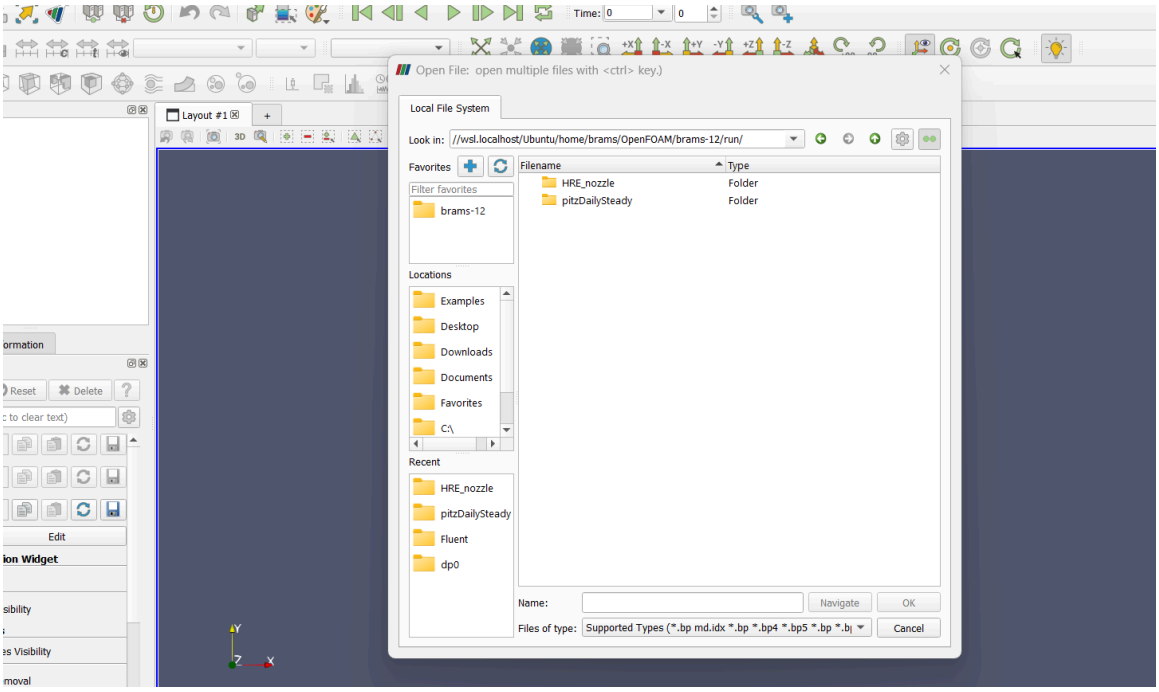
in this case: \\wsl.localhost\Ubuntu\home\brams\OpenFOAM\brams-12\run

```
\\wsl.localhost\Ubuntu\home\<USER>\OpenFOAM\<USER>-12\run
```

Next open ParaView, and open a file (Ctrl + o)

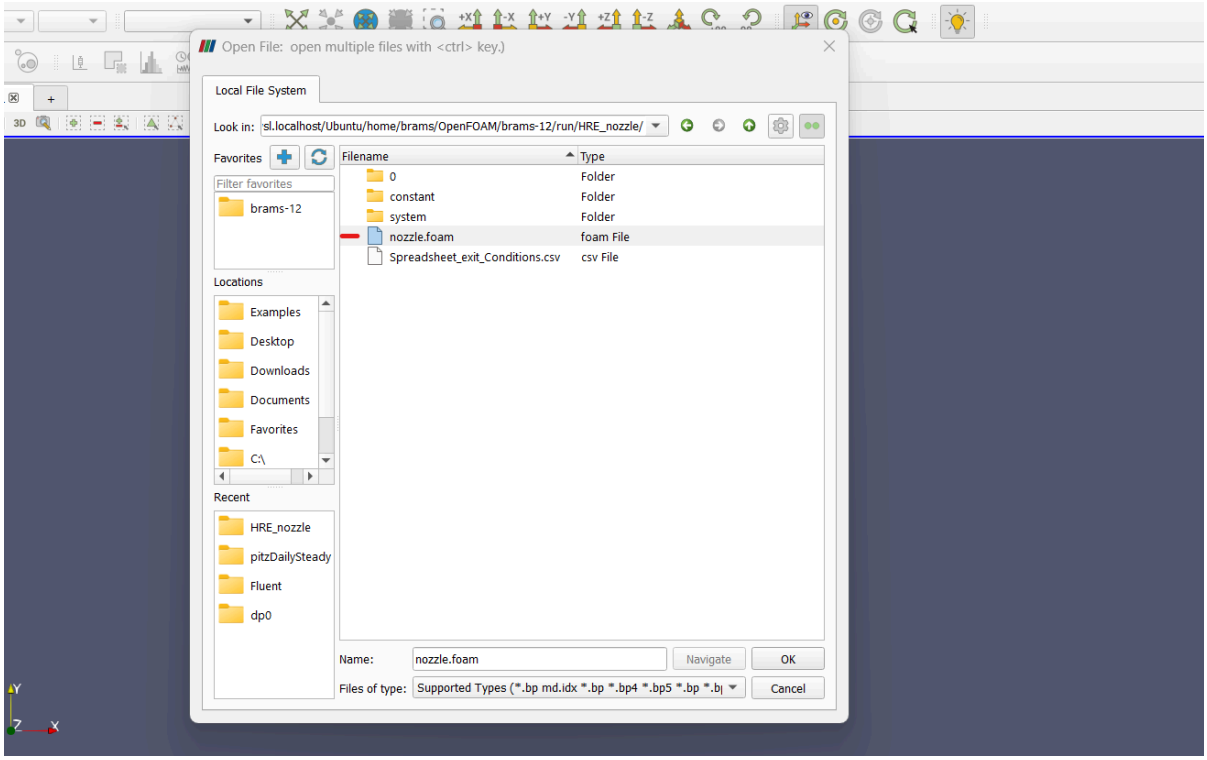


You'll want to paste the file path in the 'Look in: ' tab. Then hit enter or the search button, this will bring you to the run folder location.

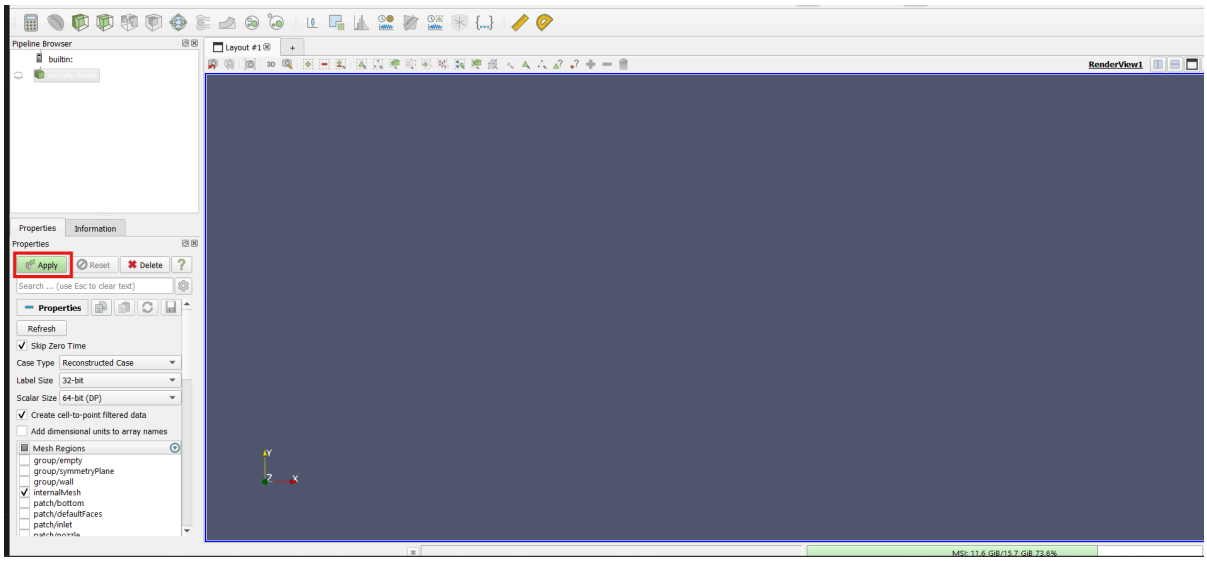


Now add this location to your favorites by clicking the ‘Plus’ button. This will save the run folder location in your favorites list, which serves as a quick access shortcut.

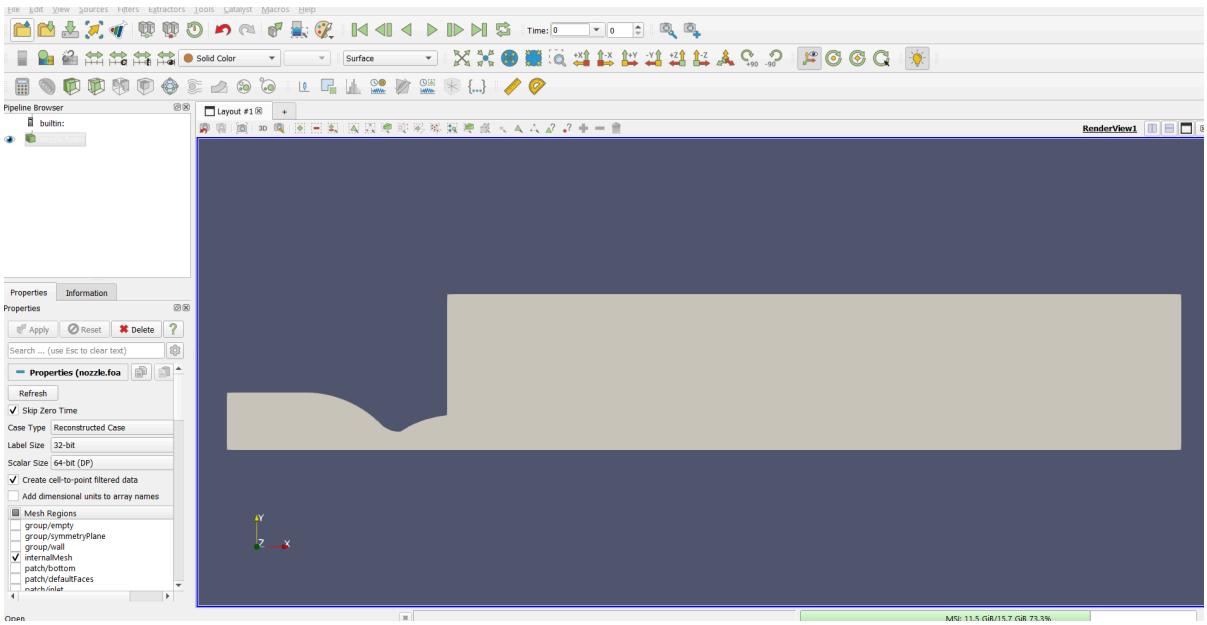
Now open the nozzle.foam file



And hit the ‘Apply’ button on your left:

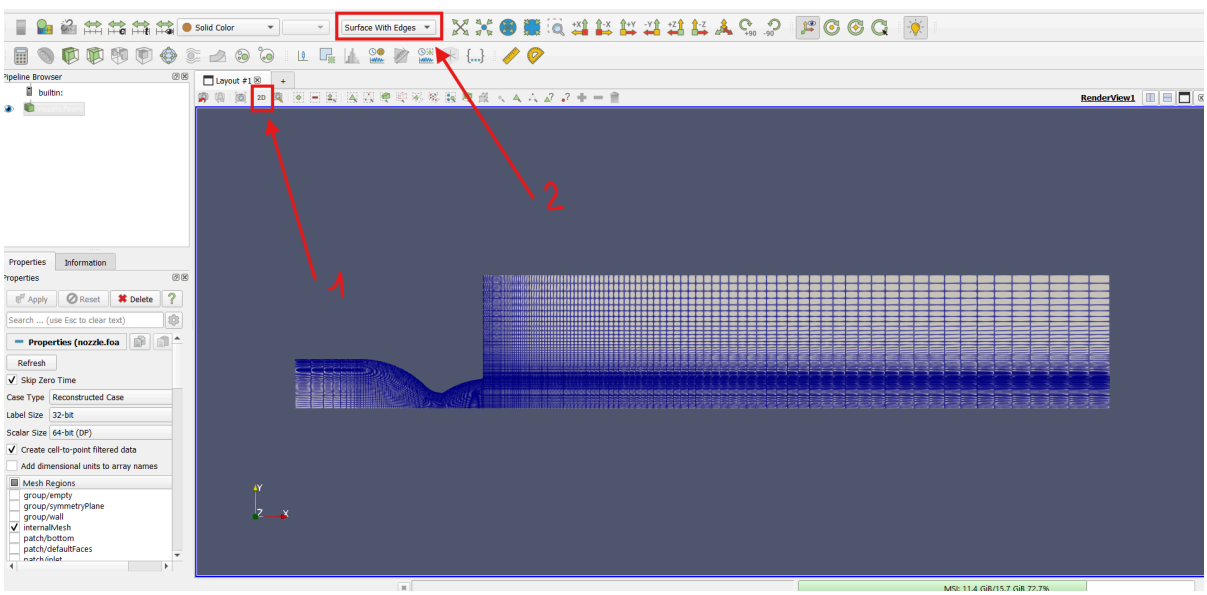


Once you did this, you’ll see the volume.



It is always recommended to perform a visual check on the mesh:

First set the viewing mode to 2D, and next select the Surface with Edges or Wireframe option.



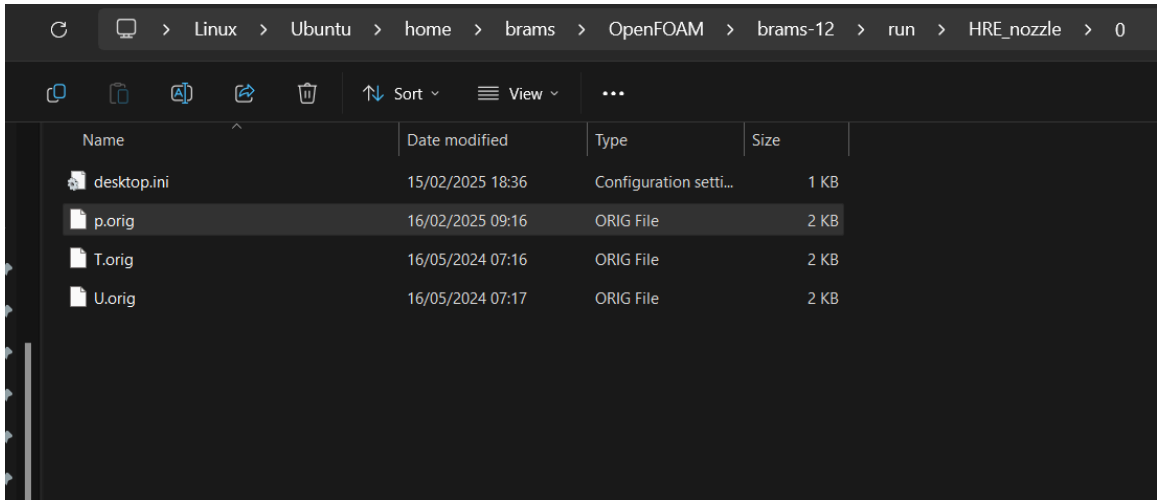
If all looks good, and there are no anomalies, you can close ParaView, the Meshing is complete!

▼ 2) Setting up the case:

Preparing the case requires us to define all the boundary conditions correctly, here we'll go over the most important files to define / check.

▼ Initial / boundary conditions:

The initial and/or boundary conditions can be found in the 0 folder. there the files p, U, T can be found. These define the boundary and initial conditions for the pressure, temperature and velocity.



▼ Pressure:

In the pressure file, set the desired inlet and atmospheric pressures. In this case, the inlet pressure is 20 00000 Pa (20 Bar) and the atmospheric pressure is 101325 Pa.

> 💡 The waveTransmissive boundary conditions prevent field reflections du to the shocks present in the flow.

```
/*---------------------------------*- C++ -*----------------------------
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  11
```

```
    \\/     M anipulation  |
\*---------------------------------------------------------------------------
FoamFile
{
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
//2D case
dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 101325;

boundaryField
{
    inlet
    {
        type    waveTransmissive;
        field   p;
        phi         phi;
        rho         rho;
        psi        psi;
        fieldInf    2000000;
        gamma       1.1724;
        lInf      0.0025;
        value       uniform 2000000;
    }

    outlet-1
    {
        type    waveTransmissive;
        field   p;
        phi         phi;
        rho         rho;
        psi        psi;
```

```
        fieldInf     101325;
        gamma        1.1724;
        lInf       1;
        value        uniform 101325;
    }

    outlet-2
    {
        type     waveTransmissive;
        field     p;
        phi          phi;
        rho          rho;
        psi         psi;
        fieldInf     101325;
        gamma        1.1724;
        lInf       1;
        value        uniform 101325;
    }
    bottom
    {
        type     symmetryPlane;
    }
    nozzle
    {
        type     zeroGradient;
    }
    defaultFaces
    {
        type     empty;
    }

}

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

```
/*--------------------------------*- C++ -*----------------------------
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  11
     \\/     M anipulation  |
\*---------------------------------------------------------------------
FoamFile
{
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
//Axisymmetric
dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 101325;

boundaryField
{
    inlet
    {
        type    waveTransmissive;
        field   p;
        phi         phi;
        rho         rho;
        psi         psi;
        fieldInf    2000000;
        gamma       1.1724;
        lInf        0.0025;
        value       uniform 2000000;
    }

    outlet-1
```

```
    {
        type    waveTransmissive;
        field   p;
        phi         phi;
        rho         rho;
        psi         psi;
        fieldInf    101325;
        gamma       1.1724;
        lInf    10;
        value       uniform 101325;
    }

    outlet-2
    {
        type    waveTransmissive;
        field   p;
        phi         phi;
        rho         rho;
        psi         psi;
        fieldInf    101325;
        gamma       1.1724;
        lInf    1;
        value       uniform 101325;
    }
    wedge1
    {
        type    wedge;
    }
    wedge2
    {
        type    wedge;
    }
    nozzle
    {
        type    zeroGradient;
    }
```

```
    defaultFaces
    {
        type    empty;
    }

}

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

▼ Velocity:

Similar to the pressure, check the inlet velocity.

```
/*---------------------------------*- C++ -*----------------------------
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  11
     \\/     M anipulation  |
\*---------------------------------------------------------------------
FoamFile
{
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
```

```
        type    waveTransmissive;
        field    U;
        phi          phi;
        rho          rho;
        psi        psi;
        fieldInf    (0 0 0);
        gamma       1.1724;
        lInf      1;
        value       uniform (0 0 0);
    }

    outlet-1
    {
        type    waveTransmissive;
        field    U;
        phi          phi;
        rho          rho;
        psi        psi;
        fieldInf    (0 0 0);
        gamma       1.1724;
        lInf      1;
        value       uniform (0 0 0);
    }

    outlet-2
    {
        type    waveTransmissive;
        field    U;
        phi          phi;
        rho          rho;
        psi        psi;
        fieldInf    (0 0 0);
        gamma       1.1724;
        lInf      1;
        value       uniform (0 0 0);
    }
```

```
    bottom
    {
        type    symmetryPlane;
    }
    nozzle
    {
        type    noSlip;
    }
    defaultFaces
    {
        type    empty;
    }

}

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

```
/*---------------------------------*- C++ -*----------------------------
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  11
     \\/     M anipulation  |
\*---------------------------------------------------------------------
FoamFile
{
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
//Axisymmetric
dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);
```

```
boundaryField
{
    inlet
    {
        type    waveTransmissive;
        field    U;
        phi          phi;
        rho          rho;
        psi         psi;
        fieldInf    (0 0 0);
        gamma       1.1724;
        lInf      1;
        value       uniform (0 0 0);
    }

    outlet-1
    {
        type    waveTransmissive;
        field    U;
        phi          phi;
        rho          rho;
        psi         psi;
        fieldInf    (0 0 0);
        gamma       1.1724;
        lInf      1;
        value       uniform (0 0 0);
    }

    outlet-2
    {
        type    waveTransmissive;
        field    U;
        phi          phi;
        rho          rho;
        psi         psi;
```

```
        fieldInf    (0 0 0);
        gamma       1.1724;
        lInf      1;
        value       uniform (0 0 0);
    }
    wedge1
    {
        type    wedge;
    }
    wedge2
    {
        type    wedge;
    }
    nozzle
    {
        type    noSlip;
    }
    defaultFaces
    {
        type    empty;
    }

}

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

▼ Temperature:

```
/*---------------------------------*- C++ -*----------------------------
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  11
     \\/     M anipulation  |
\*---------------------------------------------------------------------
FoamFile
```

```
{
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
//2D case
dimensions      [0 0 0 1 0 0 0];

internalField   uniform 298;

boundaryField
{
    inlet
    {
        type    waveTransmissive;
        field   T;
        phi         phi;
        rho         rho;
        psi        psi;
        fieldInf    3297;
        gamma       1.1724;
        lInf       0.0025;
        value       uniform 3297;
    }

    outlet-1
    {
        type    waveTransmissive;
        field   T;
        phi         phi;
        rho         rho;
        psi        psi;
        fieldInf    298;
        gamma       1.1724; //1.1724
        lInf       0.01;
```

```
        value       uniform 298;
    }

    outlet-2
    {
        type    waveTransmissive;
        field    T;
        phi          phi;
        rho          rho;
        psi         psi;
        fieldInf    298;
        gamma       1.1724; //1.1724
        lInf       0.01;
        value       uniform 298;
    }
    bottom
    {
        type    symmetryPlane;
    }
    nozzle
    {
        type    zeroGradient;
    }
    defaultFaces
    {
        type    empty;
    }
}

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

```
/*---------------------------------*- C++ -*----------------------------
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
```

```
    \\  /    A nd           | Version:  11
     \\/     M anipulation  |
\*---------------------------------------------------------------------------
FoamFile
{
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
//Axisymmetric
dimensions      [0 0 0 1 0 0 0];

internalField   uniform 298;

boundaryField
{
    inlet
    {
        type    waveTransmissive;
        field   T;
        phi         phi;
        rho         rho;
        psi        psi;
        fieldInf    3297;
        gamma       1.1724;
        lInf      0.0025;
        value       uniform 3297;
    }

    outlet-1
    {
        type    waveTransmissive;
        field   T;
        phi         phi;
        rho         rho;
```

```
        psi         psi;
        fieldInf    298;
        gamma       1.1724; //1.1724
        lInf        0.01;
        value       uniform 298;
    }

    outlet-2
    {
        type    waveTransmissive;
        field   T;
        phi         phi;
        rho         rho;
        psi         psi;
        fieldInf    298;
        gamma       1.1724; //1.1724
        lInf        0.01;
        value       uniform 298;
    }
    wedge1
    {
        type    wedge;
    }
    wedge2
    {
        type    wedge;
    }
    nozzle
    {
        type    zeroGradient;
    }
    defaultFaces
    {
        type    empty;
    }
}
```

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

▼ Parallel running:

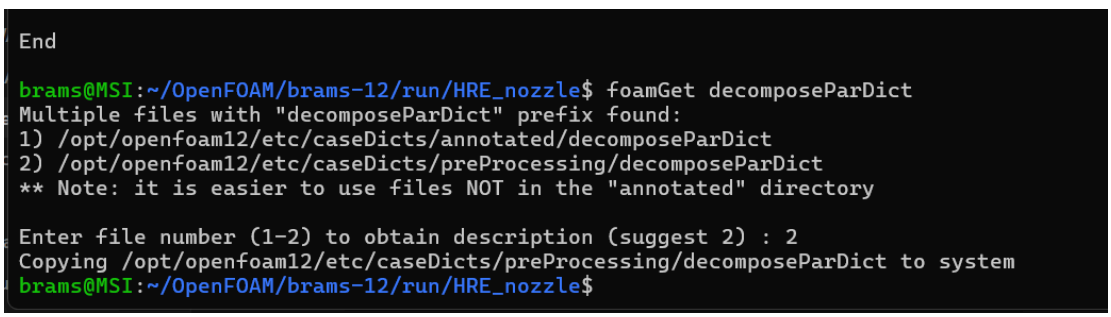
Parallel running is highly recommended for CFD, especially for cases with relatively long runtimes. Which is the case here due to the high flow velocity, causing the time step to be small in order to not exceed the max Courant number. (which would result in data loss due to skipped cells per time step)

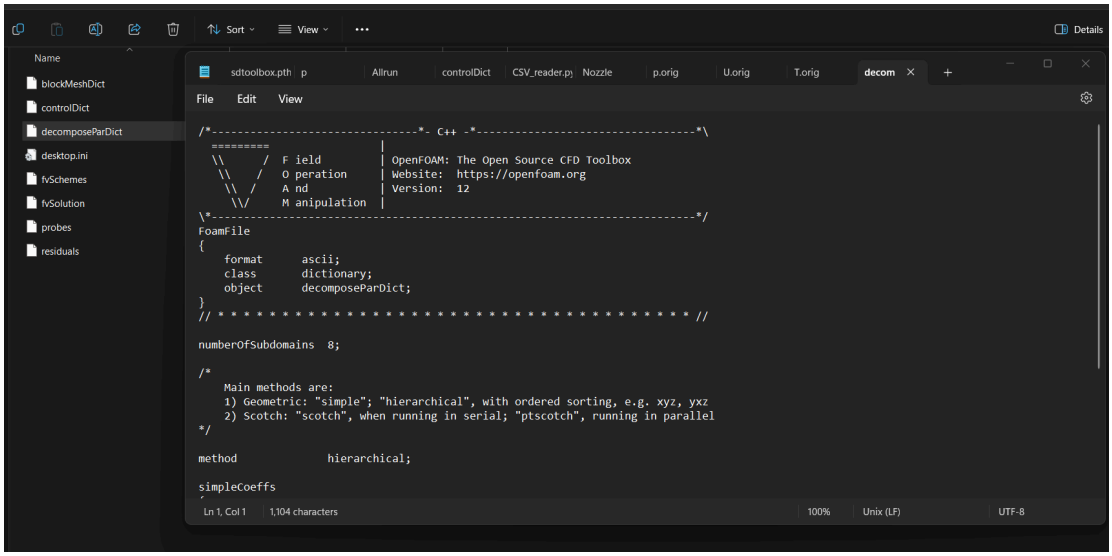
▼ Creating parallel run directory: (in case it is not present)

[OpenFOAM v12 User Guide - 3.4 Running applications in parallel]

```
foamGet decomposeParDict
```

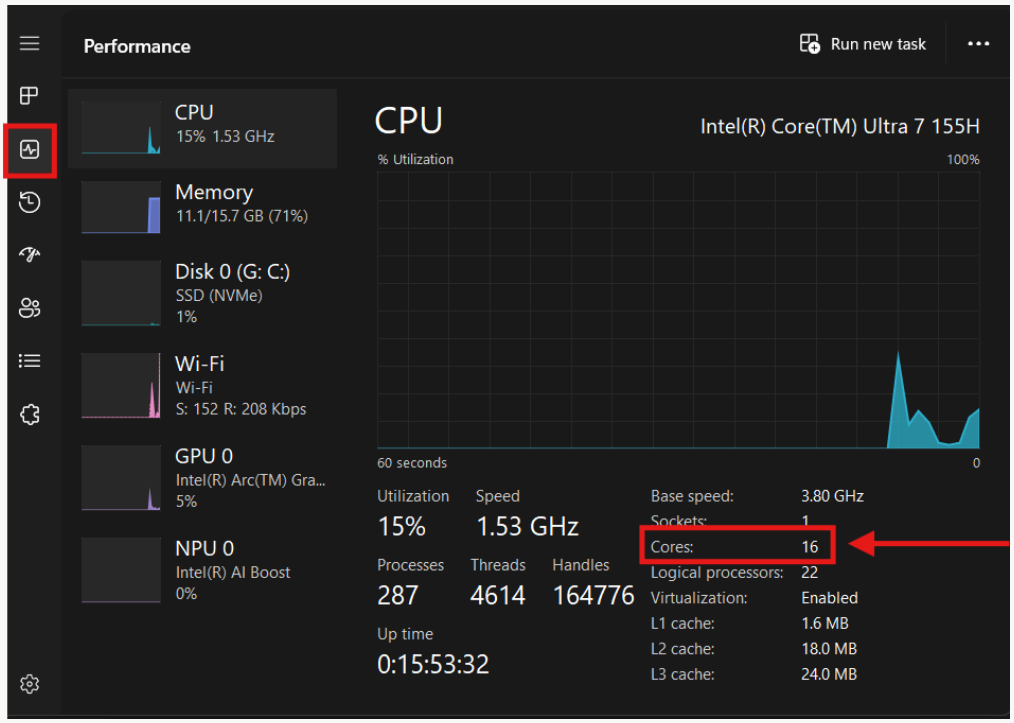
This will create a decomposeParDict file in the system folder.





Once you have found the file, open it and let us define the amount of parallel CPU cores which will be used to speed up the simulation.

💡 First make sure to check how many 'physical CPU cores' you have, you can do this on Windows by going to the task manager Ctrl + Shift + Esc, and then click on the performance tab.



The amount of cores you can use, are the physical ones, NOT THE LOGICAL PROCESSORS!

In this case we have 16 CPU cores at our disposal, however for this simulation let us use 12 of them.

```
/*---------------------------------*- C++ -*----------------------------------
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
```

```
   \\  /    A nd          | Version:  12
    \\/     M anipulation  |
\*---------------------------------------------------------------------------
FoamFile
{
   format      ascii;
   class       dictionary;
   object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains  12; //the number of physical processor cores your

/*
   Main methods are:
   1) Geometric: "simple"; "hierarchical", with ordered sorting, e.g. xyz, yxz
   2) Scotch: "scotch", when running in serial; "ptscotch", running in parall
*/

method              simple;

simpleCoeffs
{
   n               (4 3 1); // total must match numberOfSubdomains
}

hierarchicalCoeffs
{
   n               (4 3 1); // total must match numberOfSubdomains
   order           xyz;
}


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

▼ Control Directory:

The control directory or controlDict defines the overall solving settings of the simulation.
It defines the solver which is being used, the time step, runtime and save format.

```
/*---------------------------------*- C++ -*----------------------------------
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  11
     \\/     M anipulation  |
\*---------------------------------------------------------------------------
FoamFile
{
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     foamRun;

solver          shockFluid;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         0.01;

deltaT          1e-6;
```

```
writeControl    adjustableRunTime;

writeInterval   1e-05;

cycleWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression on;

timeFormat      general;

timePrecision   6;

adjustTimeStep  yes;

maxCo           0.5;

maxDeltaT       1;

//////////////////////////////////////
functions
{
    #includeFunc residuals
    #includeFunc probes
}
//////////////////////////////////////

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

The writeCompression and reduced writePrecision, help us in reducing the overall file size of the simulation. As these can get big fast, especially with

high cell counts and high write precision.

At the bottom, it can be seen that some user defined functions are included, these are mainly meant for post-processing purposes.

▼ Functions:

As mentioned in Control Directory, there are some functions, defined to help us with the post-processing of the results.

You can open these files, to check the content, the residuals file within the system folder, should normally be good to go, the only file you have to adjust for every different geometry is the probes file.

```
/*--------------------------------*- C++ -*----------------------------------
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  11
     \\/     M anipulation  |
-------------------------------------------------------------------------------
Description
    Writes out values of fields from cells nearest to specified locations.

\*---------------------------------------------------------------------------


points  (
          (-0.1 0  0) //Chamber
          (0     0  0) //Throat
          (0.0304   0  0) //Exit
          (0.06    0  0) //Point1 Exhaust Plume
          (0.2  0  0) //Point2 Exhaust Plume
          (0.4  0  0) //Point3 Exhaust Plume
       );

fields  (p  U  T);


#includeEtc "caseDicts/postProcessing/probes/probes.cfg"
```

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

Make sure to adjust the coordinates of the probe points, such that they match, or approximately match the chamber, throat, exit and some arbitrary points on the exhaust plume.

▼ Allrun:

The final step, before running the case, is to check the Allrun file.
You can do this by opening the file in a text editor, or by using the 'nano' function in the terminal.

```
#!/bin/sh
cd ${0%/*} || exit 1    # Run from this directory

# Source tutorial run functions
. $WM_PROJECT_DIR/bin/tools/RunFunctions

runApplication blockMesh
runApplication checkMesh
runApplication decomposePar
runApplication setFields
runParallel $(getApplication)
runApplication reconstructPar
>nozzle.foam



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

▼ 3) Running the case:

For running the code, go to the terminal and type:

```
./Allrun
```

This will execute the Allrun script.

Note that if you don't have permission, you'll have to chmod 700 the files

```
chmod 700 Allrun
```

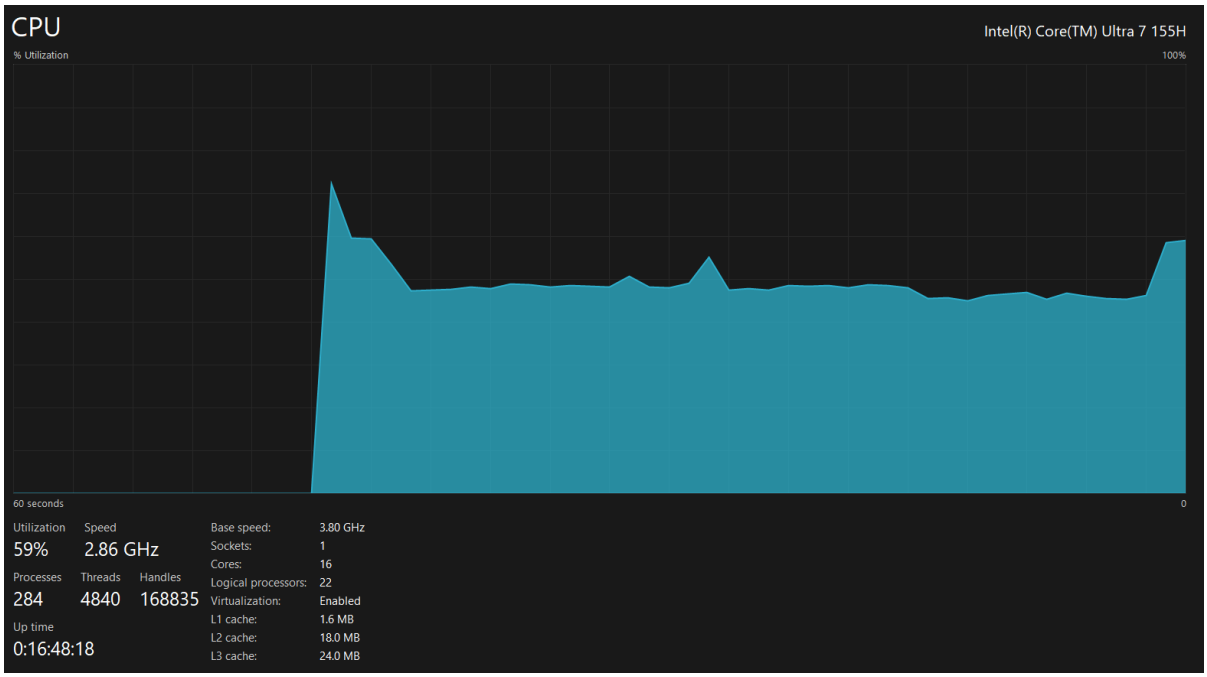
```
chmod 700 Allclean
```

notice that it might very well be that you get the following:

```
brams@MSI:~/OpenFOAM/brams-12/run/HRE_nozzle$ ./Allrun
Running blockMesh on /home/brams/OpenFOAM/brams-12/run/HRE_nozzle
Running checkMesh on /home/brams/OpenFOAM/brams-12/run/HRE_nozzle
Running decomposePar on /home/brams/OpenFOAM/brams-12/run/HRE_nozzle
Running setFields on /home/brams/OpenFOAM/brams-12/run/HRE_nozzle
Running foamRun in parallel on /home/brams/OpenFOAM/brams-12/run/HRE_nozzle using 12 processes
Running reconstructPar on /home/brams/OpenFOAM/brams-12/run/HRE_nozzle
brams@MSI:~/OpenFOAM/brams-12/run/HRE_nozzle$
```

This very fast running is mostly due to an error. In this case it was due to my PC not having 12 CPU cores available for the computation. This is most likely due to newer PC's having distinct efficiency and performance cores. Reducing the core count to 8 resolved the problem.

```
brams@MSI:~/OpenFOAM/brams-12/run/HRE_nozzle$ ./Allrun
Running blockMesh on /home/brams/OpenFOAM/brams-12/run/HRE_nozzle
Running checkMesh on /home/brams/OpenFOAM/brams-12/run/HRE_nozzle
Running decomposePar on /home/brams/OpenFOAM/brams-12/run/HRE_nozzle
Running foamRun in parallel on /home/brams/OpenFOAM/brams-12/run/HRE_nozzle using 8 processes
```



If everything goes well, you'll see your PC creates folders for each timestep of the simulation per processor core.

Name	Date modified	Type	Size
0	16/02/2025 11:20	File folder	
0.0001	16/02/2025 11:20	File folder	
0.001	16/02/2025 11:23	File folder	
0.0002	16/02/2025 11:20	File folder	
0.0003	16/02/2025 11:21	File folder	
0.0004	16/02/2025 11:21	File folder	
0.0005	16/02/2025 11:21	File folder	
0.0006	16/02/2025 11:22	File folder	
0.0007	16/02/2025 11:22	File folder	
0.0008	16/02/2025 11:23	File folder	
0.0009	16/02/2025 11:23	File folder	
0.00011	16/02/2025 11:20	File folder	
0.00012	16/02/2025 11:20	File folder	
0.00013	16/02/2025 11:20	File folder	
0.00014	16/02/2025 11:20	File folder	
0.00015	16/02/2025 11:20	File folder	
0.00016	16/02/2025 11:20	File folder	
0.00017	16/02/2025 11:20	File folder	
0.00018	16/02/2025 11:20	File folder	
0.00019	16/02/2025 11:20	File folder	

Once it is finished solving the case, it will reconstruct all the time steps, which is essentially stitching the results of each processor together.

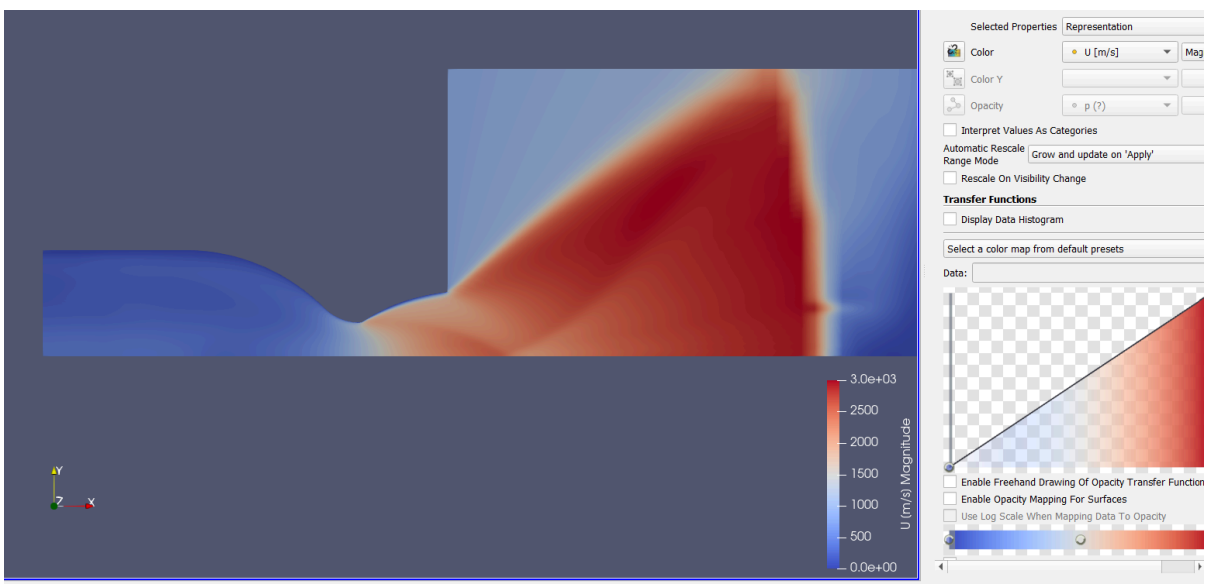
```
brams@MSI:~/OpenFOAM/brams-12/run/HRE_nozzle$ ./Allrun
Running blockMesh on /home/brams/OpenFOAM/brams-12/run/HRE_nozzle
Running checkMesh on /home/brams/OpenFOAM/brams-12/run/HRE_nozzle
Running decomposePar on /home/brams/OpenFOAM/brams-12/run/HRE_nozzle
Running foamRun in parallel on /home/brams/OpenFOAM/brams-12/run/HRE_nozzle using 8 processes
Running reconstructPar on /home/brams/OpenFOAM/brams-12/run/HRE_nozzle
```

On 8 processors, it took about 2201 seconds to solve the case (+- 40 min)

```
diagonal:  Solving for rho, Initial residual = 0, Final residual = 0, No Iterations 0
smoothSolver:  Solving for Ux, Initial residual = 7.192e-06, Final residual = 1.552e-11, No Iterations 4
smoothSolver:  Solving for Uy, Initial residual = 3.747e-05, Final residual = 7.974e-11, No Iterations 4
smoothSolver:  Solving for e, Initial residual = 2.04e-06, Final residual = 1.05e-11, No Iterations 4
ExecutionTime = 2201 s  ClockTime = 2204 s

Courant Number mean: 0.02668 max: 0.4508
deltaT = 3.222e-08
Time = 0.01s

diagonal:  Solving for rho, Initial residual = 0, Final residual = 0, No Iterations 0
smoothSolver:  Solving for Ux, Initial residual = 7.218e-06, Final residual = 1.66e-11, No Iterations 4
smoothSolver:  Solving for Uy, Initial residual = 3.79e-05, Final residual = 8.328e-11, No Iterations 4
smoothSolver:  Solving for e, Initial residual = 2.028e-06, Final residual = 9.876e-12, No Iterations 4
ExecutionTime = 2201 s  ClockTime = 2204 s

End

Finalising parallel run
```

When that is complete, you can open ParaView and look at the results:

▼ 4) Post processing the results:



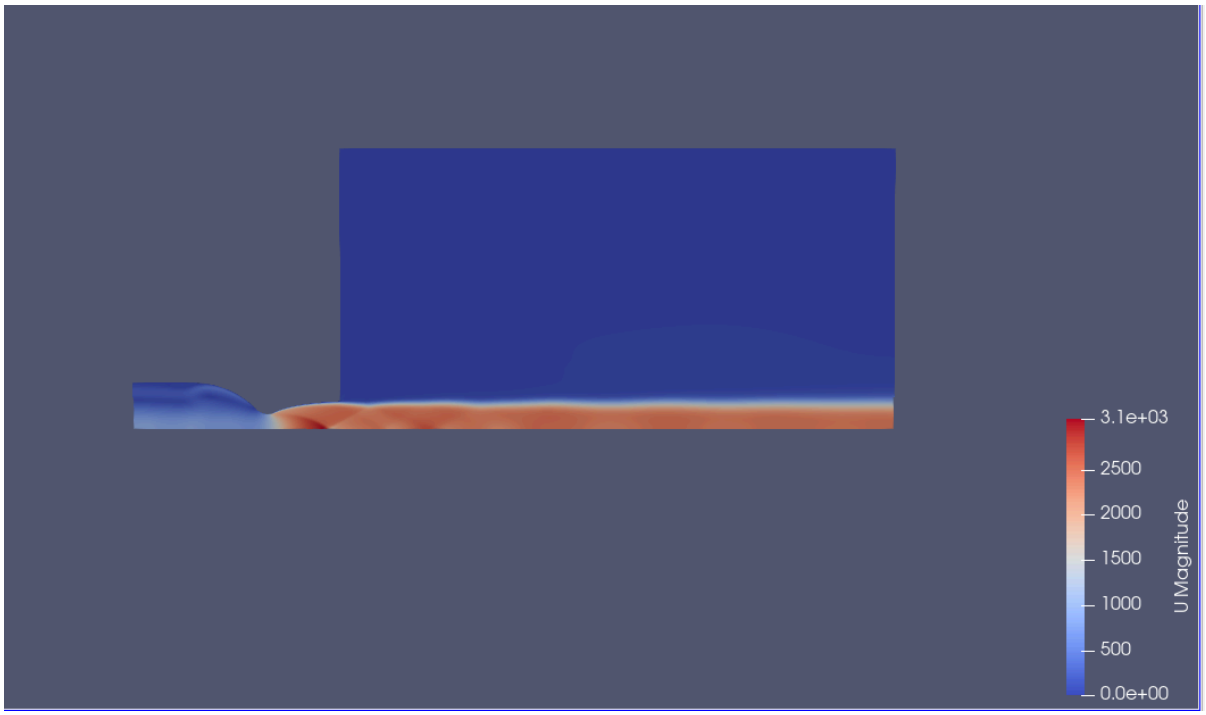
note, this nozzle had the following contour:

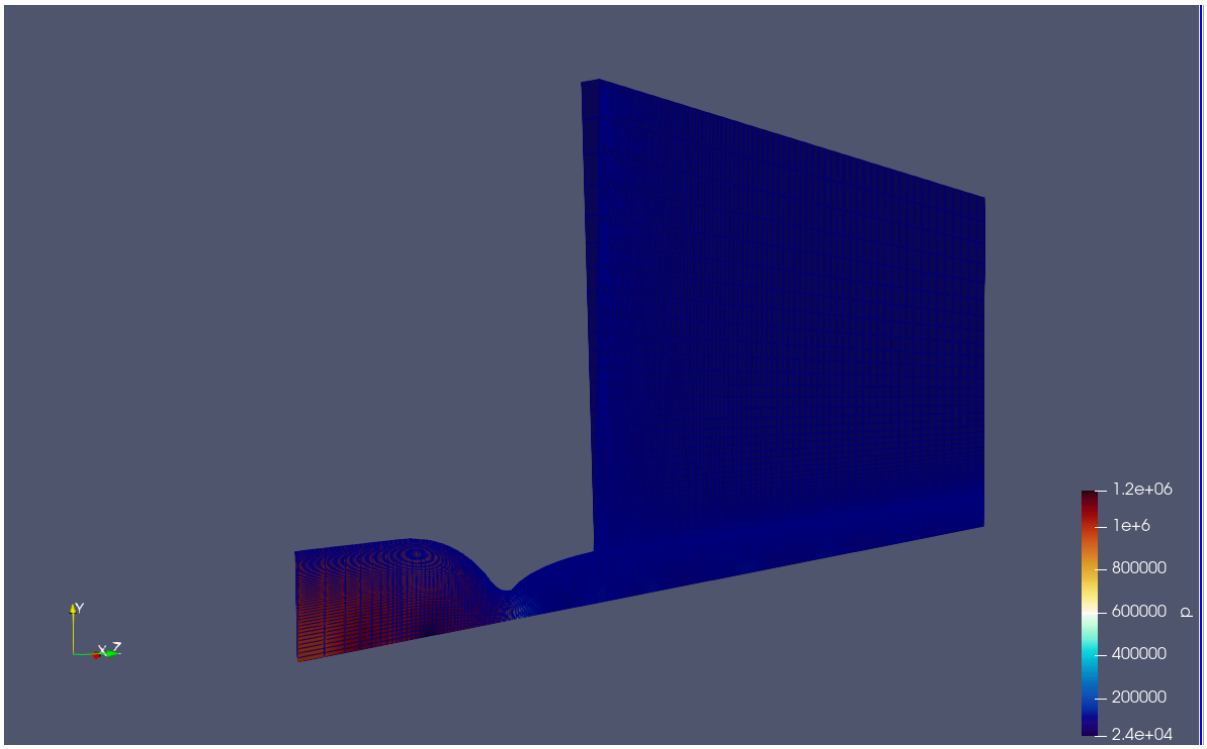
```
HRE_1_Geo = CH_Geo(HRE_1, 'Bell', 10, 0.8 ,45, 35, 5,"OF","True")
```

The reason for the above profile is double sided, on the one side nozzle length is of importance, to get the correct expansion, but on the other side, the purely 2D case doesn't model the atmospheric flow that accurately, leading to an artifical pressure buildup in the chamber after the nozzle.

The issue was resolved by utilizing a Wedged mesh, and setting axisymmetric boundary conditions.

```
HRE_1_Geo = CH_Geo(HRE_1, 'Bell', 10, 1.5 ,45, 35, 2,"OF_axs","True")
```


3.1e+03
2500
2000
1500
1000
500
0.0e+00
U Magnitude


1.2e+06
1e+6
800000
600000
400000
200000
2.4e+04
Y
X
Z