Supersonic flow past a wedge

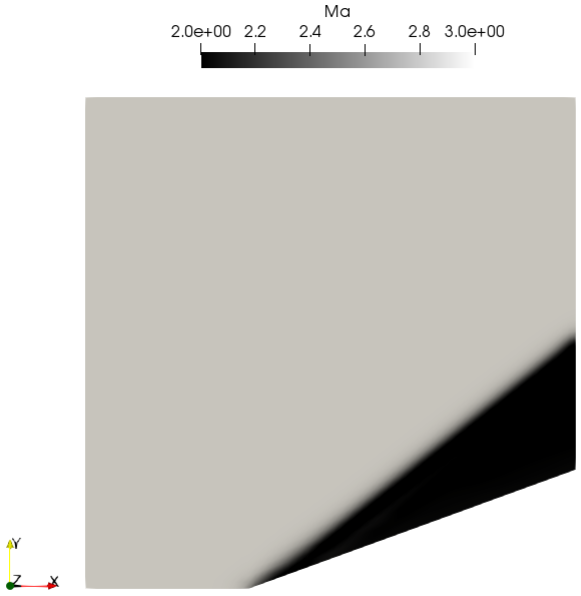
- Let us run this case. Go to the directory:

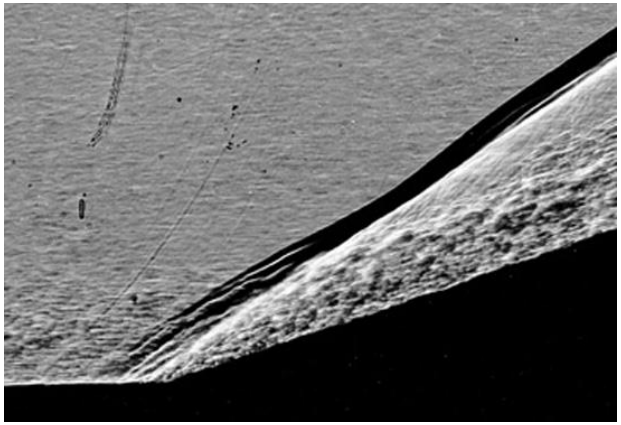
```
$PTOFC/supersonic_wedge
```

- $PTOFC is pointing to the directory where you extracted the training material.
- In the case directory, you will find a few scripts with the extension .sh, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on.
- These scripts can be used to run the case automatically by typing in the terminal, for example,
  - `$> 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.

Supersonic flow past a wedge

Supersonic flow past a wedge – Supersonic flow with shock waves





The picture is a shadowgraph of Mach 3 airflow over an 18 degree compression corner which shows the shock wave/turbulent boundary-layer interaction (from G. Settles, PhD thesis, Princenton, 1975)

Physical and numerical side of the problem:

- In this case, we are going to solve the flow past a 20 degrees compression corner.
- The inlet Mach number is equal to 3.
- We are going to use a compressible solver, in pseudo-transient mode and zero viscosity.
- Therefore, the governing equations of the problem are the compressible Euler equations.
- We are going to work in a 2D domain.
- This problem has an analytical solution and plenty of validation data.

Supersonic flow past a wedge

What are we going to do?

- We will use this case to learn how to setup supersonic flow cases.
- We will compare the numerical solution with the analytical solution.
- We will run the case with a robust numerics, but you are invited to try different setups and compare the different outcomes.
- To find the numerical solution we will use the solver `rhoPimpleFoam` in pseudo-transient mode and with zero viscosity (Euler equations).
- `rhoPimpleFoam` is a transient solver for turbulent flow of compressible fluids, with optional mesh motion and mesh topology changes.
- After finding the numerical solution we will do some sampling.
- Then we will do some plotting (using gnuplot or Python) and scientific visualization.

Supersonic flow past a wedge

Selecting thermophysical properties

```
1     thermoType
2     {
3         type            hePsiThermo;
4         mixture         pureMixture;
5         transport       const;
6         thermo          hConst;
7         equationOfState perfectGas;
8         specie          specie;
9         energy          sensibleEnthalpy;
10    }
11
12    mixture
13    {
14        specie
15        {
16            nMoles      1;
17            molWeight   28.9;
18        }
19        thermodynamics
20        {
21            Cp          1005;
22            Hf          0;
23        }
24        transport
25        {
26            mu          0.0;
27            Pr          0.713;
28        }
29    }
```

- The thermophysical properties are set in the dictionary *thermophysicalProperties*.
- This dictionary file is located in the directory **constant**.
- In the sub-dictionary **thermoType** (lines 1-10), we define the thermophysical models. Many of these options are hardwired with the solver used.
- The **transport** keyword (line 5) concerns evaluating dynamic viscosity. In this case the viscosity is constant.
- The thermodynamic models (**thermo** keyword) are concerned with evaluating the specific heat Cp (line 6). In this case Cp is constant.
- The **equationOfState** keyword (line 7) concerns to the equation of state of the working fluid. In this case, we are using the ideal gas equation model.

$$\rho = \frac{p}{RT}$$

Supersonic flow past a wedge

Selecting thermophysical properties

```
1     thermoType
2     {
3         type            hePsiThermo;
4         mixture         pureMixture;
5         transport       const;
6         thermo          hConst;
7         equationOfState perfectGas;
8         specie          specie;
9         energy          sensibleEnthalpy;
10    }
11
12    mixture
13    {
14        specie
15        {
16            nMoles      1;
17            molWeight   28.9;
18        }
19        thermodynamics
20        {
21            Cp          1005;
22            Hf          0;
23        }
24        transport
25        {
26            mu          0.0;
27            Pr          0.713;
28        }
29    }
```

- The form of the energy equation to be used is specified in line 9 (**energy**).
- In this case we are using enthalpy formulation (**sensibleEnthalpy**). In this formulation, the following equation is solved,

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot (\rho \mathbf{u} h) + \frac{\partial \rho K}{\partial t} + \nabla \cdot (\rho \mathbf{u} K) - \frac{\partial p}{\partial t} = \nabla \cdot (\alpha_{eff} \nabla e) + \rho \mathbf{g} \cdot \mathbf{u} + S$$

- In the sub-dictionary **mixture** (lines 12-29), we define the thermophysical properties of the working fluid (air in this case).
- In line 17, we define the molecular weight.
- In line 21, we define the specific heat **Cp**. The heat of formation **Hf** is defined in line 22 (not used in this case).
- As we are using the transport model **const** (line 5), we need to define the dynamic viscosity **mu** and Prandtl number **Pr** (lines 26 and 27).
- As we want to solve the Euler equations, we set the viscosity to zero (line 26). We also define the Prandtl number in line 27 but is not used in this case as we are solving the Euler equations.

Supersonic flow past a wedge

Selecting turbulence model

- As we are solving the Euler equations (no viscosity), there is no turbulence involved.
- Nevertheless, we need to set the turbulence model to laminar in the dictionary *momentumTransport*.
- This dictionary is located in the directory **constant**.

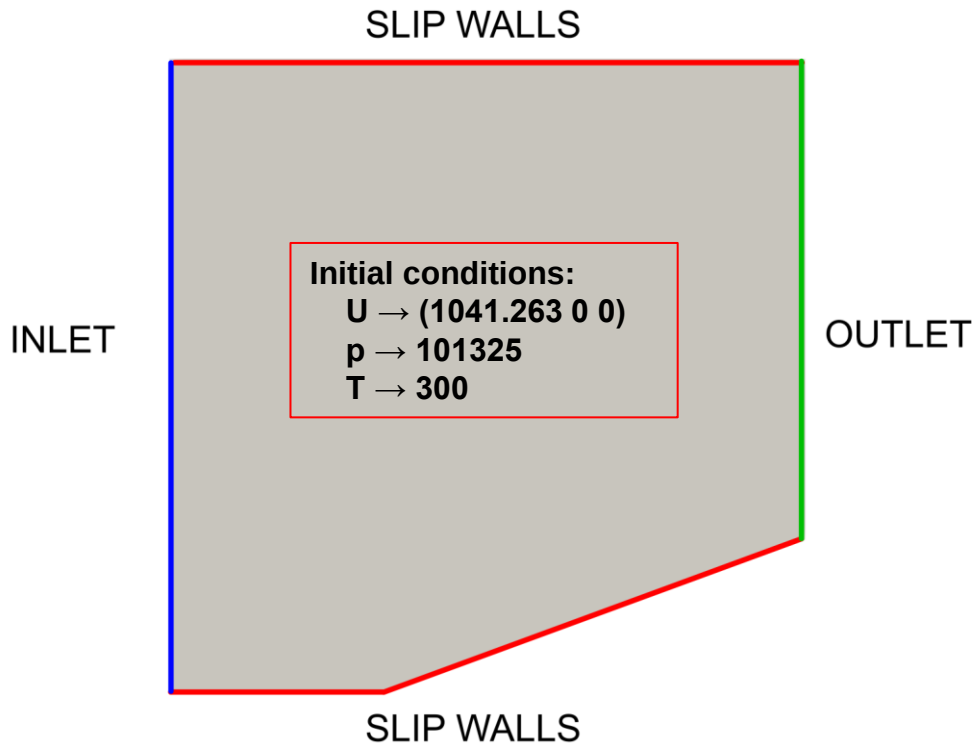
```
simulationType    laminar;
```

- At this point we are done with the physical properties and turbulence modeling.
- Let us define the boundary conditions.

Supersonic flow past a wedge

Boundary and initial conditions

- The boundary and initial conditions are defined as follows.



Boundary conditions

INLET:
- U → (1041.263 0 0)
- p → 101325
- T → 300

} **fixedValue**

OUTLET:
- U → inletOutlet
- p → zeroGradient
- T → inletOutlet

SLIP WALLS:
- U, p, T → slip

Note: the inlet velocity is equivalent to a Mach number of 3. The speed of sound used was for air at sea level and 300 Kelvin.

Supersonic flow past a wedge

Selecting the discretization schemes

```
1     ddtSchemes
2     {
3         default         localEuler;
4     }
5
6     gradSchemes
7     {
8         default         cellLimited Gauss linear 1;
9         grad(p)         cellLimited Gauss linear 0.333;
10        grad(h)         cellLimited Gauss linear 0.333;
11    }
12
13    divSchemes
14    {
15        default         none;
16        div(phi,U)      Gauss vanLeerV;
17        div(phi,K)      Gauss linear;
18        div(phi,h)      Gauss limitedLinear 1;
19        div(phid,p)     Gauss limitedLinear 1;
20       div((nuEff*dev2(T(grad(U))))) Gauss linear;
21    }
22
23    laplacianSchemes
24    {
25        default         Gauss linear limited 1;
26    }
27
28    interpolationSchemes
29    {
30        default         linear;
31    }
32
33    snGradSchemes
34    {
35        default         limited 1;
36    }
```

- The discretization schemes are set in the dictionary *fvSchemes* located in the directory **system**.
- When dealing with compressible flows and strong discontinuities (such as shock waves), it is of paramount importance to set a robust and accurate numerics, as the one used in this case.
- In line 3, we define a pseudo-transient temporal discretization scheme. This method is more robust than a steady formulation but at the cost of a higher computational cost (however, less than the computational cost of a full unsteady simulation).
- In lines 6-11 we define the gradient discretization scheme. When dealing with shock waves, it is recommended to use an aggressive limiter for **grad(U)** (line 8).
- Be careful not to add very aggressive limiters to **grad(p)** and **grad(h)** (line 8), as they may add a lot of numerical diffusion.
- In this case, we are using a stable and accurate limiter for **grad(p)** and **grad(h)** (lines 9 and 10).

Supersonic flow past a wedge

Selecting the discretization schemes

```
1     ddtSchemes
2     {
3         default         localEuler;
4     }
5
6     gradSchemes
7     {
8         default         cellLimited Gauss linear 1;
9         grad(p)         cellLimited Gauss linear 0.333;
10        grad(h)         cellLimited Gauss linear 0.333;
11    }
12
13    divSchemes
14    {
15        default         none;
16        div(phi,U)      Gauss vanLeerV;
17        div(phi,K)      Gauss linear;
18        div(phi,h)      Gauss limitedLinear 1;
19        div(phid,p)     Gauss limitedLinear 1;
20       div((nuEff*dev2(T(grad(U))))) Gauss linear;
21    }
22
23    laplacianSchemes
24    {
25        default         Gauss linear limited 1;
26    }
27
28    interpolationSchemes
29    {
30        default         linear;
31    }
32
33    snGradSchemes
34    {
35        default         limited 1;
36    }
```

- In lines 13-21 we define the discretization scheme of the convective terms.
- Notice that for velocity (line 16) we are using a TVD scheme.
- TVD schemes are highly recommended when you are dealing with strong discontinuities (such as shock waves).
- In lines 17-19 we define the discretization schemes for the variables related to the energy equation. In general, the setup used is accurate and stable.

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot (\rho \mathbf{u} h) + \frac{\partial \rho K}{\partial t} + \nabla \cdot (\rho \mathbf{u} K) - \frac{\partial p}{\partial t} = \nabla \cdot (\alpha_{eff} \nabla e) + \rho \mathbf{g} \cdot \mathbf{u} + S$$

`div(phi,h)` `div(phi,K)`

- Line 19 is related to the transonic correction used. this correction is set in the dictionary *`fvSolution`*.
- Line 20 is related to the Reynolds stresses (not relevant as we are solving the Euler equations).

Supersonic flow past a wedge

Selecting the solution method and linear solvers

```
1     solver
2     {
3         p
4         {
5             solver          PBiCGStab;
6             preconditioner  DILU;
7             tolerance       1e-06;
8             relTol          0.01;
9             minIter         2;
10        }
11
12        pFinal
13        {
14            $p;
15            relTol          0;
16            minIter         2;
17        }
18
19        "(U.*|h.*)"
20        {
21            solver          PBiCGStab;
22            preconditioner  DILU;
23            tolerance       1e-08;
24            relTol          0;
25            minIter         2;
26        }
27
28        "rho.*"
29        {
30            solver          diagonal;
31        }
32
33    }
34
```

- The solution method, corrections and linear solvers are set in the dictionary *fvSolution* located in the directory **system**.
- In this case, we are using the linear solver **PBiCGStab** for all variables except **rho**.
- In compressible solvers, **rho** is computed from the thermodynamical variables, therefore, we use a diagonal solver, in other words, back substitution (line 30).
- Notice that the notation **U.***, **h.***, and **rho.***, is a wildcard used to specify the same method in all variables (**U**, **UFinal**, **h**, **hFinal** and so on).
- It is recommended to set the minimum number of iterations to 2 or 3, this increases the stability and helps at linearizing the linear system.
- The minimum number of iterations is set by using the keyword **minIter**.

Supersonic flow past a wedge

Selecting the solution method and linear solvers

```
1     PIMPLE
2     {
3         momentumPredictor           yes;
4
5         nOuterCorrectors            1;
6         nCorrectors                 3;
7         nNonOrthogonalCorrectors    2;
8
9         transonic                   yes;
10
11        maxCo                       0.5;
12        rDeltaTSmoothingCoeff       0.1;
13        rDeltaTDampingCoeff         0.9;
14        maxDeltaT                   1;
15
16        pMin          10000;
17        pMax          1000000;
18    }
19
20    relaxationFactors
21    {
22        equations
23        {
24            ".*" 1;
25        }
26    }
27
```

- In this case we are using the **PIMPLE** pressure-velocity coupling (line 1).
- In line 3 we enable the momentum predictor option.
- It is strongly recommended to use this option when dealing with high-speed flows (it is enabled by default).
- Remember, when using this option, the user needs to specify the linear solvers for the variables **UFinal**, **hFinal**, **rhoFinal** and so on.
- In lines 5-7, we define the number of corrector steps. The setup used here is the recommended one when dealing with compressible flows and with good quality meshes.
- In line 9, we enable the transonic correction.
- This option is recommended when the critical Mach number is exceeded (usually for Mach number values above 0.5).

Supersonic flow past a wedge

Selecting the solution method and linear solvers

```
1     PIMPLE
2     {
3         momentumPredictor           yes;
4
5         nOuterCorrectors            1;
6         nCorrectors                 3;
7         nNonOrthogonalCorrectors    2;
8
9         transonic                   yes;
10
11        maxCo                       0.5;
12        rDeltaTSmoothingCoeff       0.1;
13        rDeltaTDampingCoeff         0.9;
14        maxDeltaT                   1;
15
16        pMin          10000;
17        pMax          1000000;
18    }
19
20    relaxationFactors
21    {
22        equations
23        {
24            ".*" 1;
25        }
26    }
27
```

- In lines 11-14, we define the control parameters related to the pseudo-transient formulation. These values should be adjusted by the user. The values used in this case are very conservative.
- In lines 16-17 we define the minimum and maximum values of pressure. This is not compulsory, but it is recommended to define them to avoid reaching unrealistic values.
- In lines 20-26, we define the under-relaxation factors (URF).
- In this case, we define all the URF to one (this will help in increasing the diagonal dominance of the matrix of coefficients).
- You can use smaller URF values if you are having stability problems. It is not recommended to reduce the URF to values below 0.3 as it will slow down too much the computation.

Supersonic flow past a wedge

Final remarks – Energy equation formulation

- When solving the enthalpy formulation of the energy equation,

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot (\rho \mathbf{u} h) + \frac{\partial \rho K}{\partial t} + \nabla \cdot (\rho \mathbf{u} K) - \frac{\partial p}{\partial t} = \nabla \cdot (\alpha_{eff} \nabla e) + \rho \mathbf{g} \cdot \mathbf{u} + S$$

the pressure work term $\partial p / \partial t$ can be excluded from the solution.

- This has a stabilizing effect on the solution, specially if you are using steady solvers.
- To turn off the pressure work term $\partial p / \partial t$, set the option dpdt to no (**dpdt no;**) in the `thermophysicalProperties` dictionary.

- Finally, when you work with compressible solvers you use absolute pressure, and the working units are in Pascals.

Supersonic flow past a wedge

Visualizing the solution – Capturing the shock waves

- The easiest way to capture shock waves is by computing the gradient of the density field (also know as the numerical Schlieren).
- You can compute this derived field by using the `postProcess` utility as follows,

```
1.  $> postProcess -func 'grad(rho)' -latestTime
```

- This will create the new field **grad(rho)**.
- Remember, the field **rho** must exist prior to computing its gradient.
- The method used to compute the field **grad(rho)** is the one defined in the dictionary *`fvSchemes`*.

```
gradSchemes
{
    grad(rho)  cellLimited Gauss linear 0.5;
}

gradSchemes
{
    default cellLimited Gauss linear 0.5;
}
```

grad(rho) explicitly defined

If grad(rho) is not defined, the default method will be used to compute grad(rho)

Supersonic flow past a wedge

Visualizing the solution – Capturing the shock waves

- Another way to capture shock waves is by computing the Laplacian of the density field (also know as the numerical shadowgraph).
- Remember, the Laplacian of a quantity ϕ is equivalent to the divergence of the gradient of the quantity ϕ. That is,

$$\Delta\phi = \nabla^2\phi = \nabla \cdot \nabla\phi$$

- You can compute this derived field by using the `postProcess` utility as follows,

1. ```
   $> postProcess -func 'div(grad(rho))' -latestTime
   ```

- This will create the new field **div(grad(rho))**.
- Remember, the field **grad(rho)** must exist prior to computing its divergence.
- The method used to compute the **div(grad(rho))** is the one define in the dictionary *`fvSchemes`*. In this case,

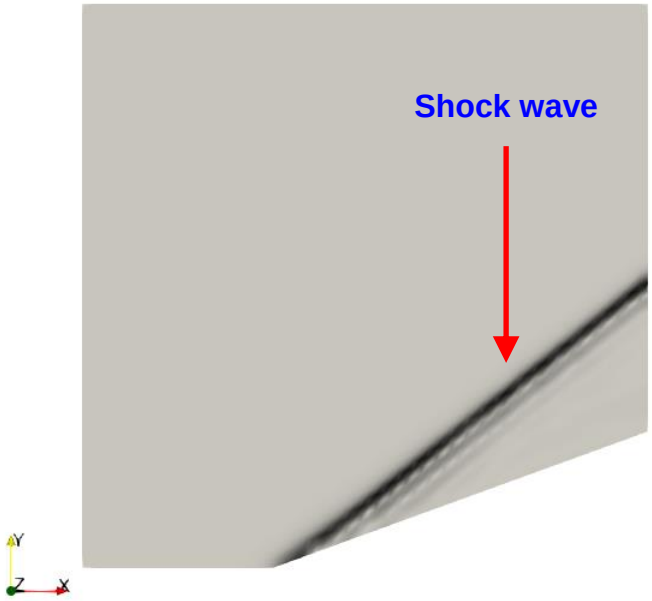
```
divSchemes
{
    div(grad(rho)) Gauss linear;
}
```

div(grad(rho)) explicitly defined
```

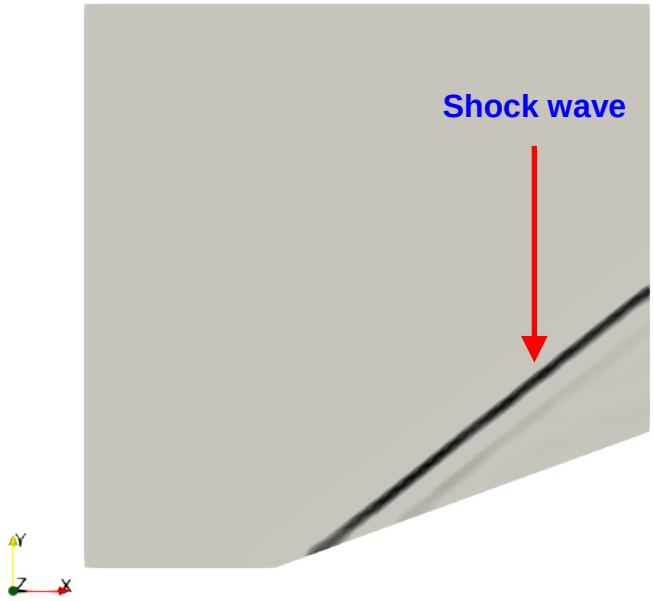
Supersonic flow past a wedge

Visualizing the solution – Capturing the shock waves

- Visualization of the shock wave.
- The shock wave corresponds to the black region.



Gradient of the density field (numerical Schlieren)

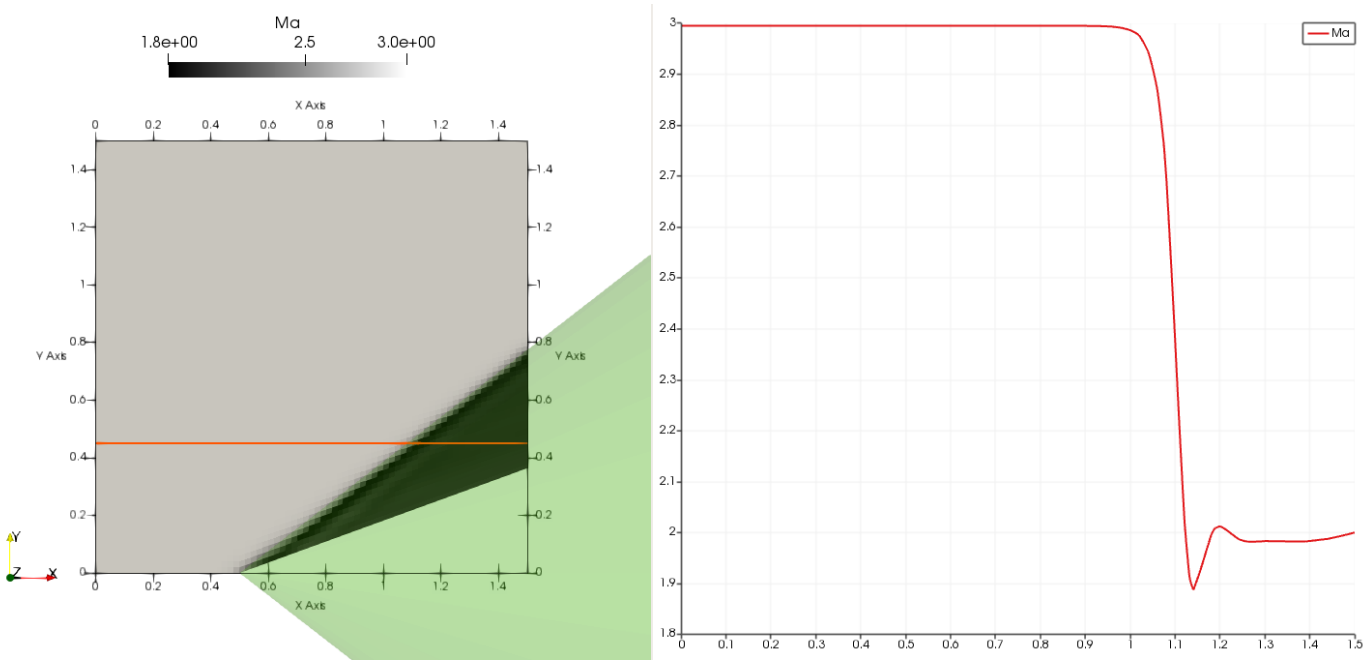


Laplacian of the density field (numerical shadowgraph)

Supersonic flow past a wedge

Visualizing the solution in Paraview

- At this point, you can visualize the solution using Paraview and compare the output with the analytical solution.



Analytical solution [1, 2]

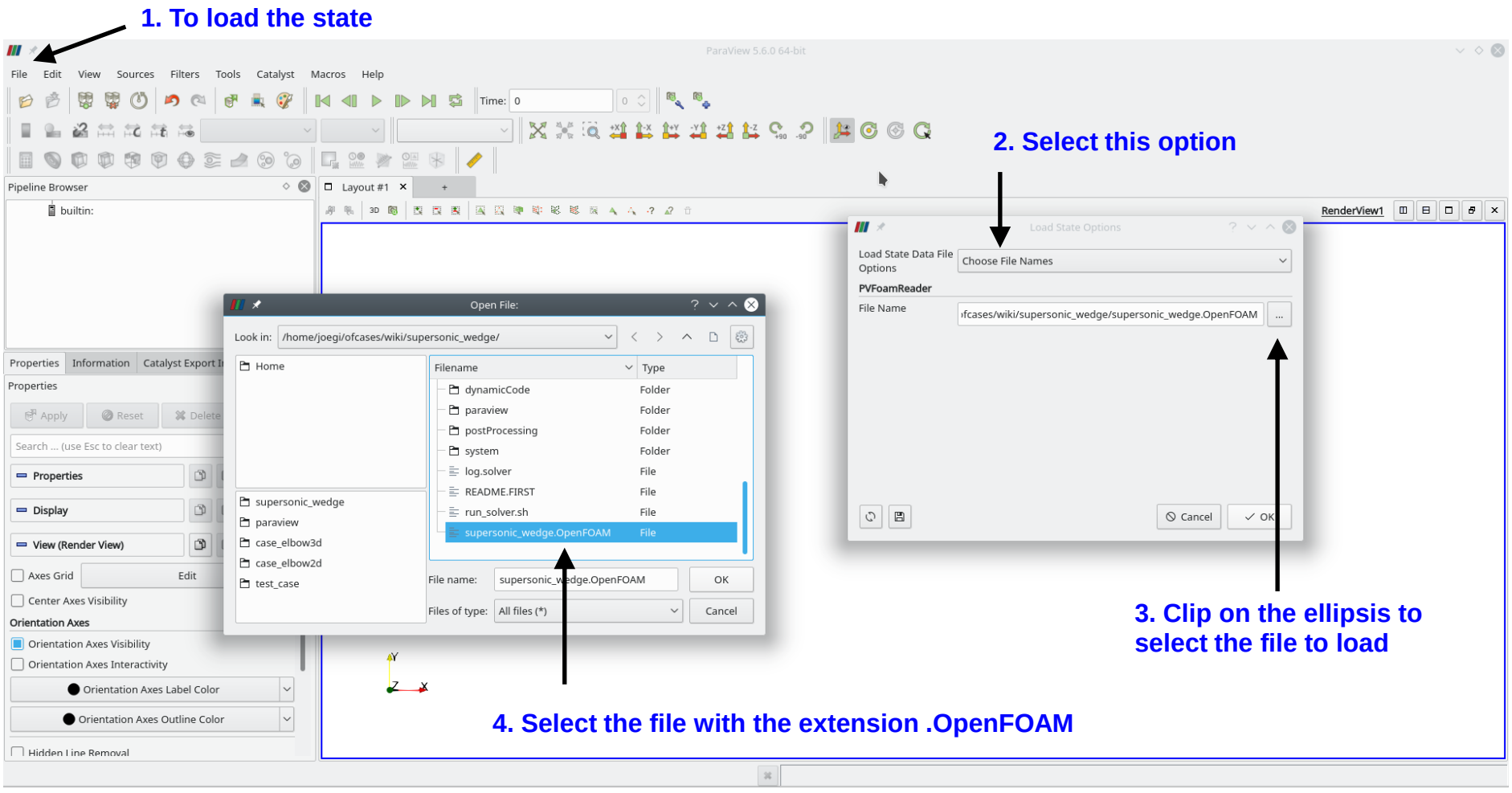
Wedge angle	M_1	M_2	β
20°	3	1.988	37.8°

- [1] Modern compressible flow: with historical perspective. Third edition. McGraw-Hill.
- [2] Compressible aerodynamics calculator. http://www.dept.aoe.vt.edu/~devenpor/aoe3114/calc.html

Supersonic flow past a wedge

Visualizing the solution in Paraview

- In Paraview, you can use the predefined states distributed with this tutorial.
- They are located in the folder **`paraview`**.
- After loading the state, proceed as follows,

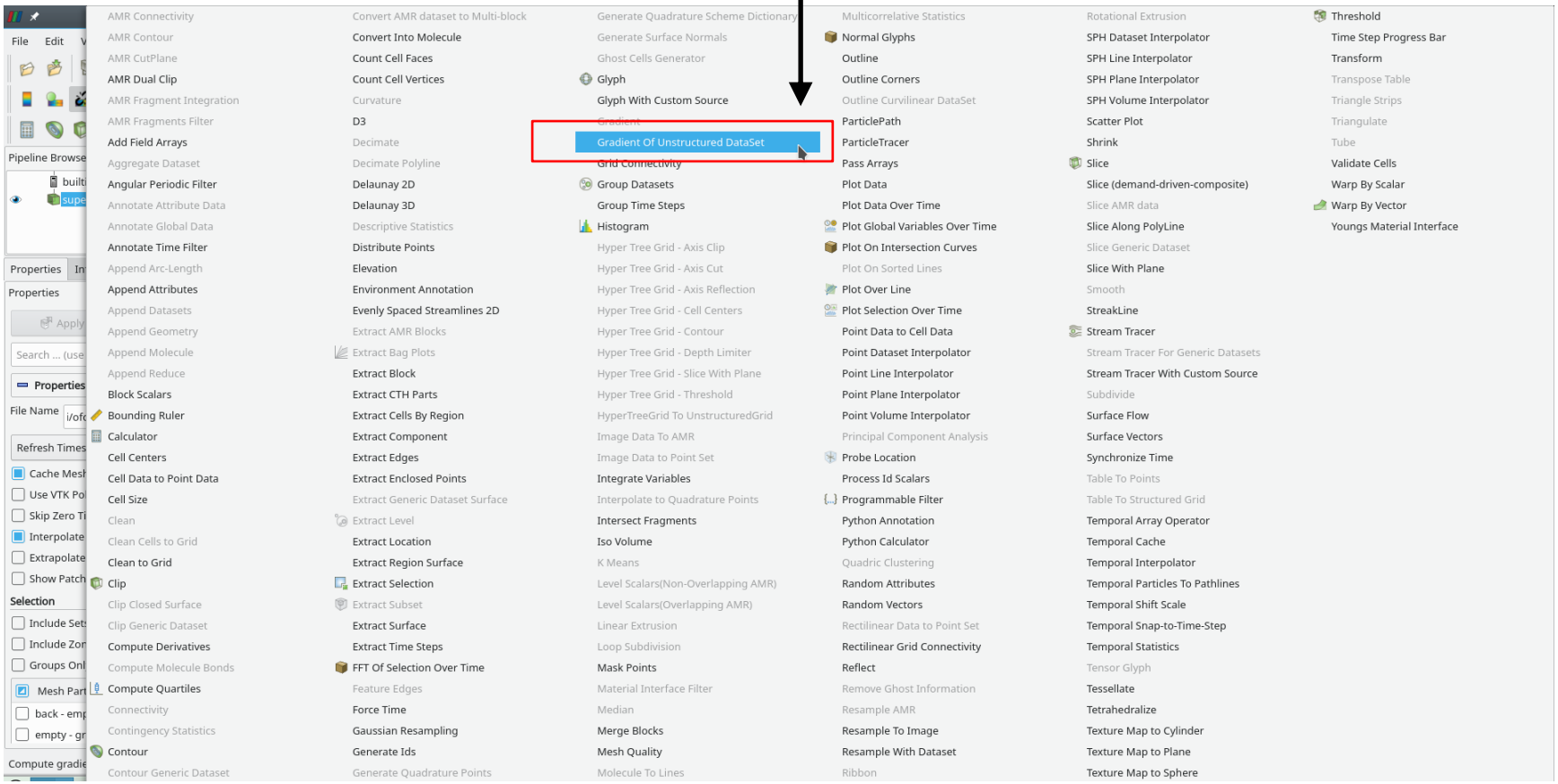


Supersonic flow past a wedge

Visualizing the solution in Paraview

- To compute the gradient of a field in Paraview, you can use the filter `Gradient Of Unstructured DataSet`

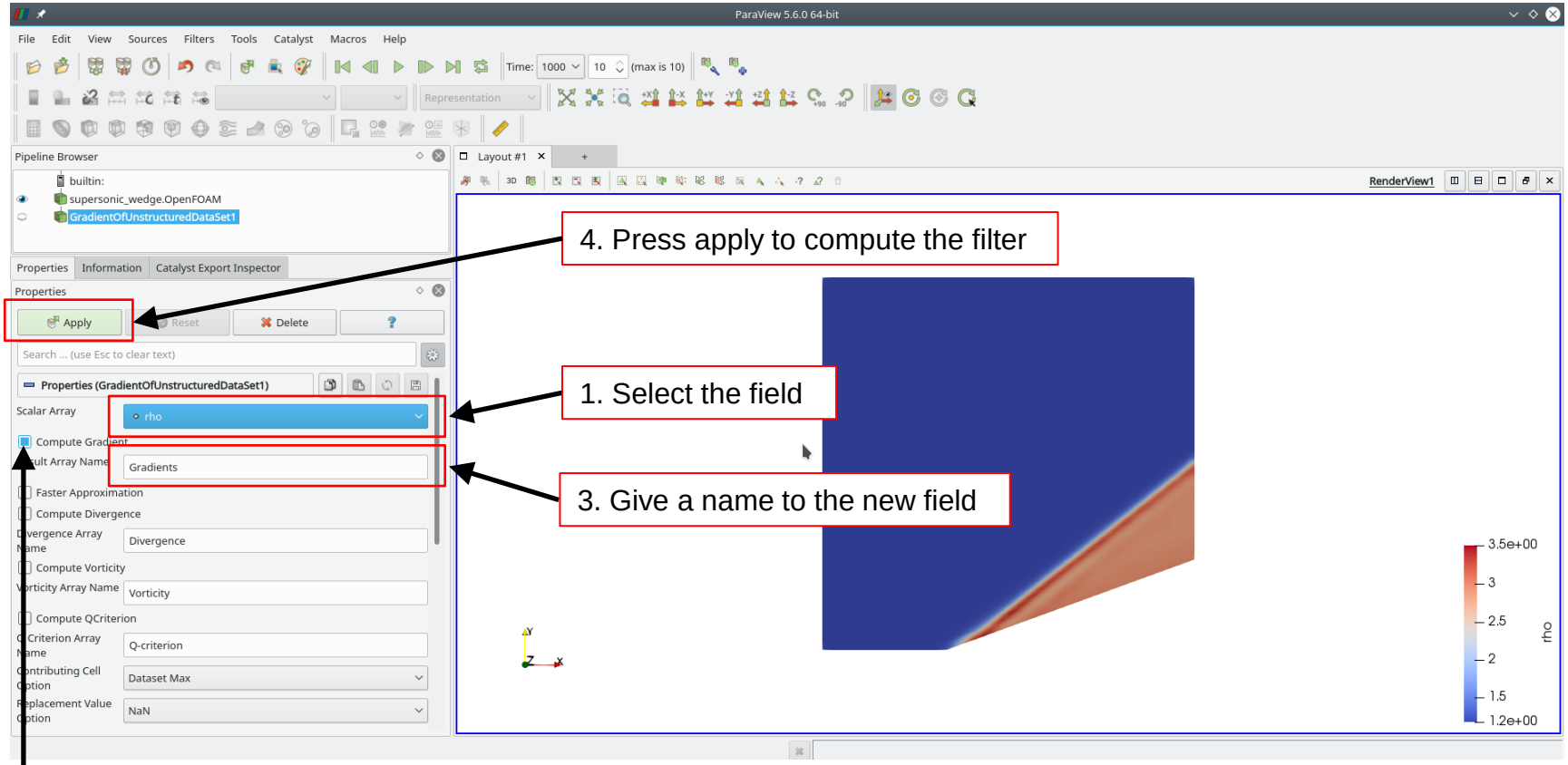
`Gradient Of Unstructured DataSet` filter



Supersonic flow past a wedge

Visualizing the solution in Paraview

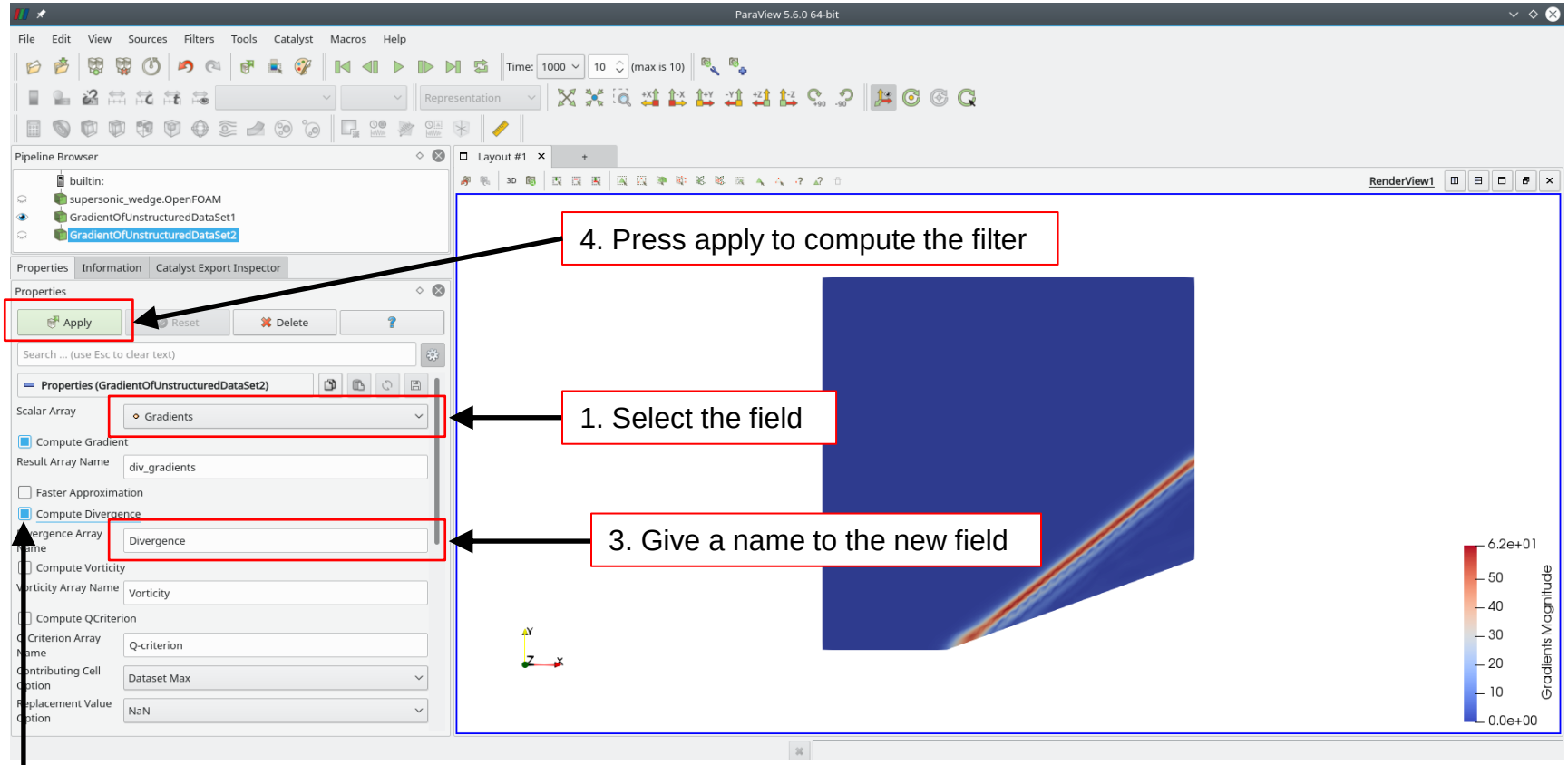
- After selecting the filter `Gradient Of Unstructured DataSet`, follow these steps to compute the gradient of field,



Supersonic flow past a wedge

Visualizing the solution in Paraview

- After selecting the filter `Gradient Of Unstructured DataSet`, follow these steps to compute the divergence of the gradient of a field, that is, the Laplacian,

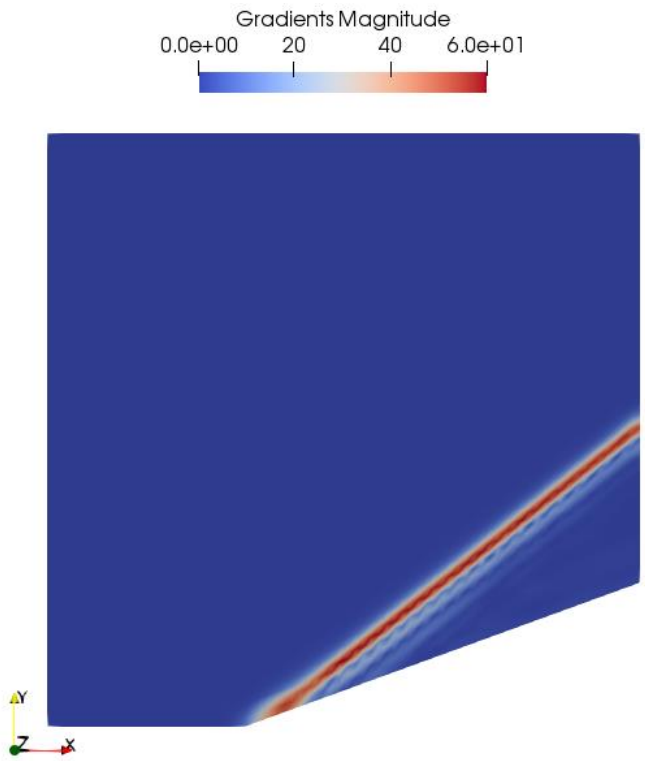


2. Select the option `Compute Divergence`

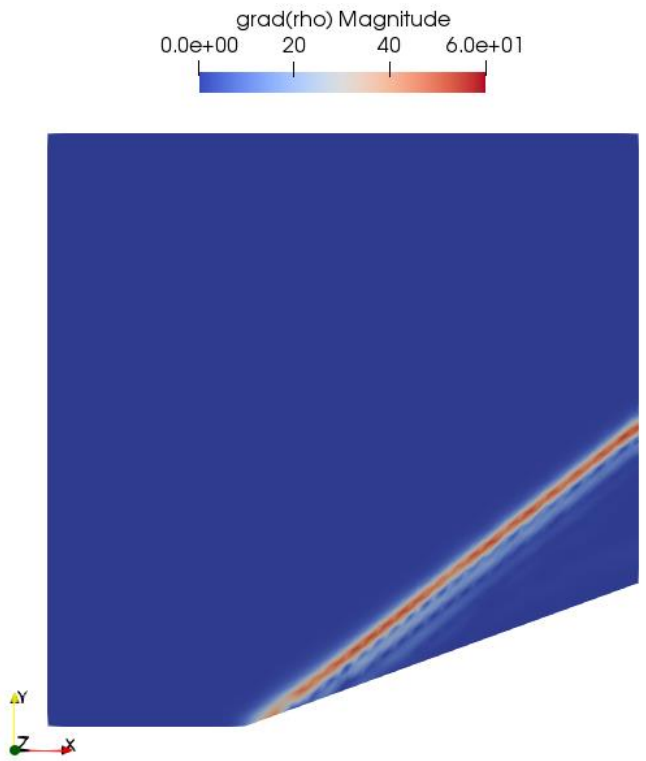
Supersonic flow past a wedge

Visualizing the solution in Paraview

- Paraview vs. OpenFOAM – Comparison of numerical Schlieren computation.



Paraview

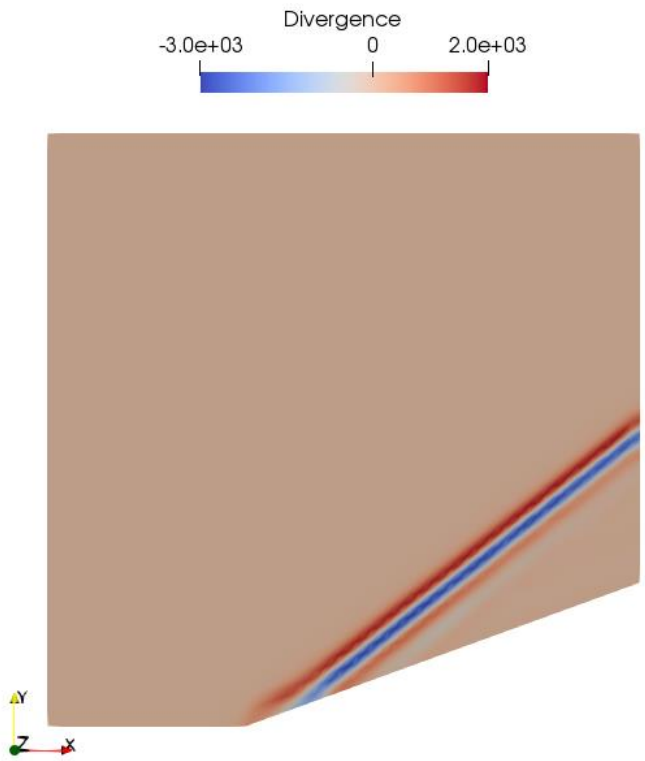


OpenFOAM

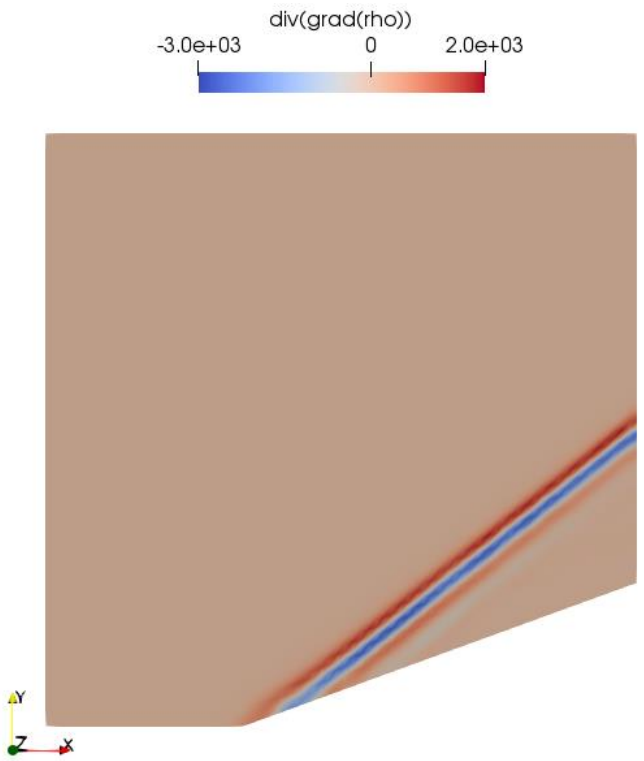
Supersonic flow past a wedge

Visualizing the solution in Paraview

- Paraview vs. OpenFOAM – Comparison of numerical shadowgraph computation.



Paraview



OpenFOAM

Supersonic flow past a wedge

Running the case

- Let us first generate the mesh using the meshing utility `blockMesh`.
- In the terminal window type:

```
1. $> foamCleanTutorials
2. $> rm -rf 0 > /dev/null 2>&1
3. $> cp -r 0_org 0 > /dev/null 2>&1
4. $> blockMesh
5. $> checkMesh
```

- The dictionary *`blockMeshDict`* has been already parametrized.
- In this case we are using a medium mesh and the wedge angle is 20 degrees.
- If you want to try different meshes or wedge angles, feel free to modify the dictionary *`blockMeshDict`*.

Supersonic flow past a wedge

Running the case

- You will find this tutorial in the directory **`$PTOFC/supersonic_wegde`**
- In the terminal window type:

```
1. $> cp -r 0_empty 0
2. $> renumberMesh -overwrite
3. $> rhoPimpleFoam | tee log.solver
4. $> rhoPimpleFoam -postProcess -func MachNo
5. $> paraFoam
```

- In line 4 we compute the Mach number.
- Have in mind that the pseudo-transient formulation can be used only with transient solvers (`rhoPimpleFoam` in this case).

Supersonic flow past a wedge

Running the case

- You will find this tutorial in the directory **`$PTOFC/supersonic_wegde`**
- To compute the numerical Schlieren, in the terminal window type:

1. ```
   $> postProcess -func 'grad(rho)' -latestTime
   ```

- To compute the numerical shadowgraph, in the terminal window type:

1. ```
   $> postProcess -func ‘div(grad(rho))' -latestTime
   ```