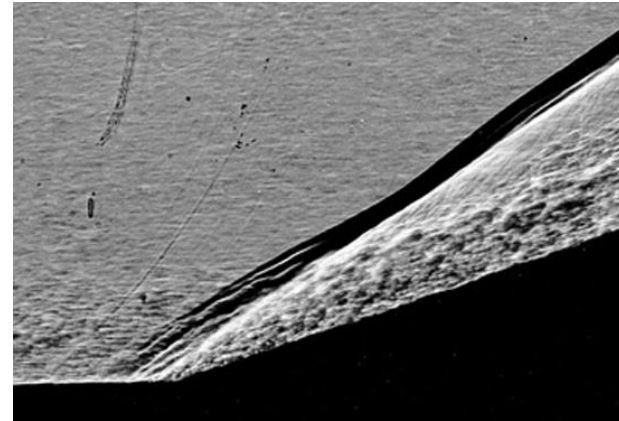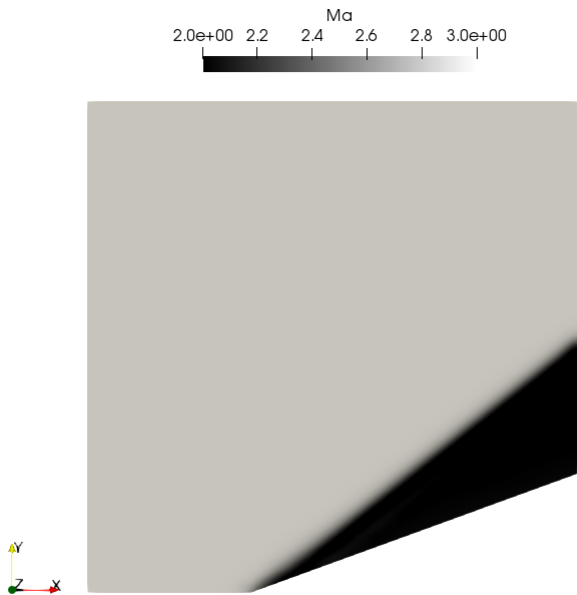# Supersonic flow past a wedge

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 and an inlet Mach number 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

- 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 the `README.FIRST` file. In this file, you will find the general instructions of how to run the case. In this file, you might also find some additional comments.

- You will also find a few additional files (or scripts) with the extension .sh, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on. These files can be used to run the case automatically by typing in the terminal, for example, `sh run_solver`.

- We highly recommend you to open the `README.FIRST` file 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®, use the automatic scripts to run the cases.

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

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*.

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.

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

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.

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 *turbulenceProperties*.
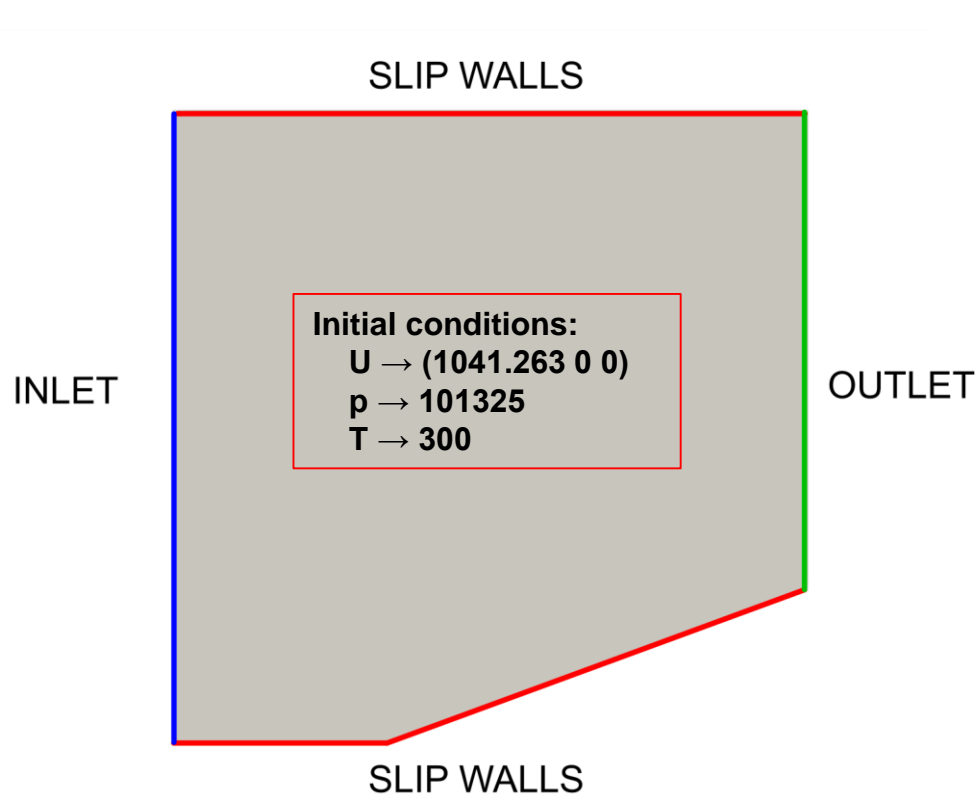
- This dictionary is located in the directory **constant**.

```
simulationType     laminar;
```

- At this point we are done with the physical properties.

- Let us define the boundary conditions.

Selecting turbulence model

- The boundary conditions are defined as follows.

SLIP WALLS

Initial conditions:
U → (1041.263 0 0)
p → 101325
T → 300

INLET

OUTLET

SLIP WALLS

Boundary conditions

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

OUTLET:
    U → inletOutlet
    p → zeroGradient
    T → inletOutlet

SLIP WALLS:
    U, p, T → inletOutlet

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.

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. 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).

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 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 17-20 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*.

Selecting the solution method and corrections

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

- The solution method, corrections and linear solvers are set in the dictionary *fvSolution* located in the directory `system`.

- Hereafter, we will not address how to set linear solvers.

- You can refer to previous tutorials to learn how to set the linear solvers.

- 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).

- In lines 4-6, 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 8, we enable the transonic correction. This option is recommended when the critical Mach number is exceeded (usually for Mach number values above 0.5

Selecting the solution method and corrections

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

- In lines 10-13, 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 15-16 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 19-25, 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

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 are computing the Mach number.

Supersonic flow past a wedge

Visualizing the solution

- At this point, you can visualize the solution using paraview and compare the numerical results with the analytical solution.
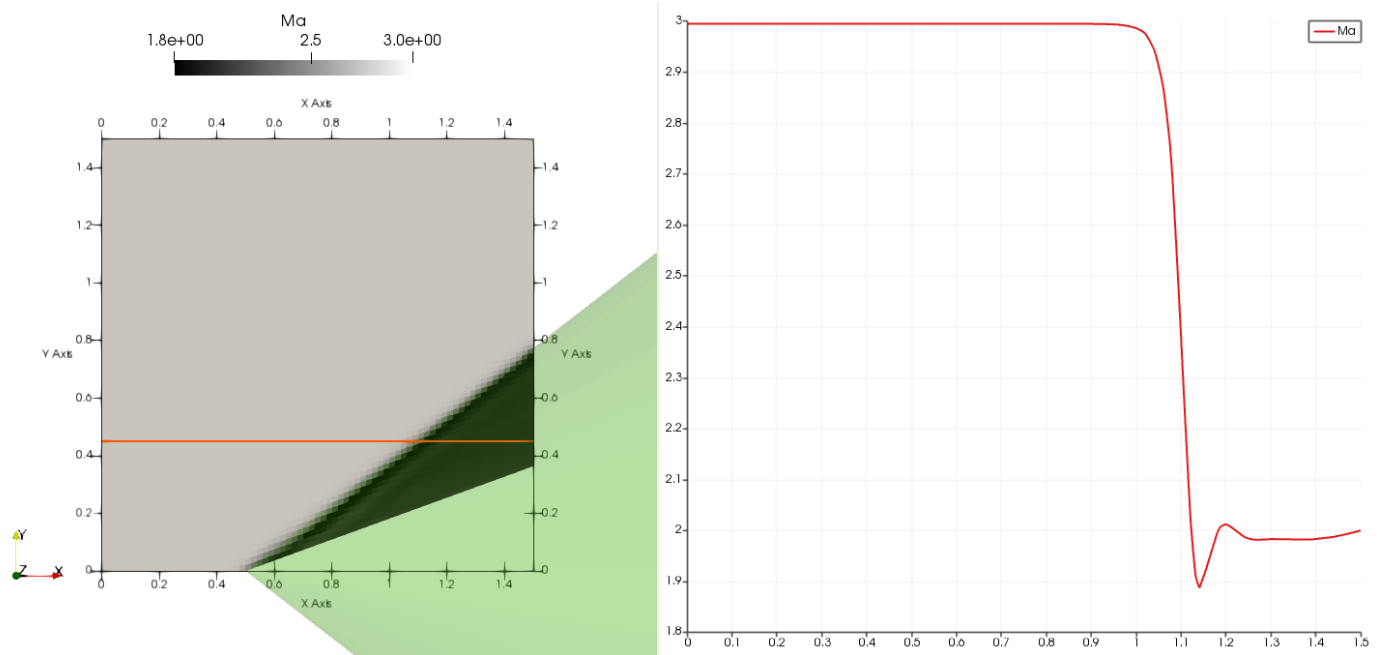
Analytical solution [1, 2]

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, you can use the predefined states, they located in the folder `paraview`.

- After loading the state, proceed as follows,

1. To load the state

2. Select this option

3. Clip on the ellipsis to select the file to load

4. Select the file with the extension .OpenFOAM