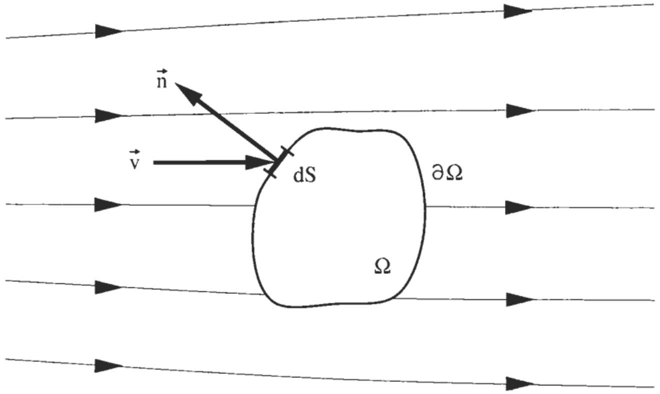
Part-II: Compressible FVM methods in OF

The Conservation Equations in Integral Form



$$\frac{\partial}{\partial t}\int_{\Omega} \vec{W}\, d\Omega + \oint_{\partial\Omega} (\vec{F}_c - \vec{F}_v)\, dS = \int_{\Omega} \vec{Q}\, d\Omega\,.$$

$$\vec{W} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix}.$$

$$\vec{F}_c = \begin{bmatrix} \rho V \\ \rho u V + n_x p \\ \rho v V + n_y p \\ \rho w V + n_z p \\ \rho H V \end{bmatrix}$$

$$V \equiv \vec{v}\cdot\vec{n} = n_x u + n_y v + n_z w$$

$$\vec{F}_v = \begin{bmatrix} 0 \\ n_x\tau_{xx} + n_y\tau_{xy} + n_z\tau_{xz} \\ n_x\tau_{yx} + n_y\tau_{yy} + n_z\tau_{yz} \\ n_x\tau_{zx} + n_y\tau_{zy} + n_z\tau_{zz} \\ n_x\Theta_x + n_y\Theta_y + n_z\Theta_z \end{bmatrix},$$

$$\Theta_x = u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + k\frac{\partial T}{\partial x}$$

$$\Theta_y = u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + k\frac{\partial T}{\partial y}$$

$$\Theta_z = u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + k\frac{\partial T}{\partial z}$$

$$\vec{Q} = \begin{bmatrix} 0 \\ \rho f_{e,x} \\ \rho f_{e,y} \\ \rho f_{e,z} \\ \rho \vec{f}_e\cdot\vec{v} + \dot{q}_h \end{bmatrix}.$$

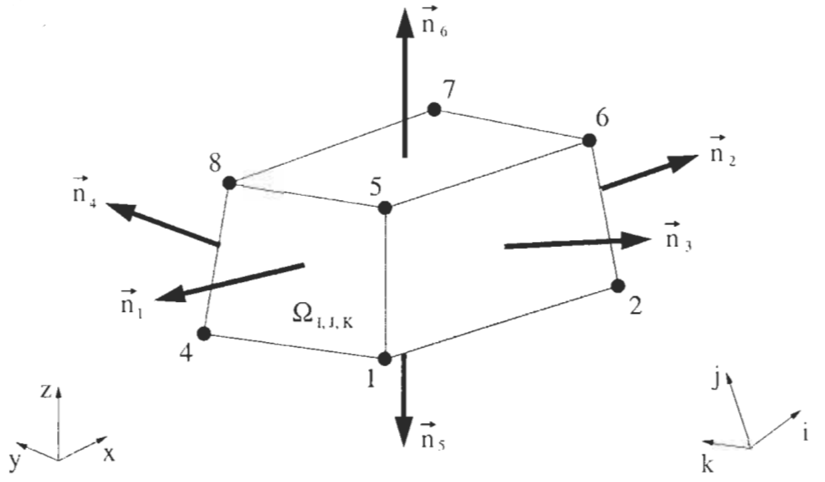
Integration of time derivative

$$\boxed{\frac{\partial}{\partial t}\int_{\Omega}\vec{W}\,d\Omega} + \oint_{\partial\Omega}(\vec{F}_c - \vec{F}_v)\,dS = \int_{\Omega}\vec{Q}\,d\Omega\,.$$

Step 2

$$\frac{\partial}{\partial t}\int_{\Omega}\vec{W}\,d\Omega = \Omega\,\frac{\partial\vec{W}}{\partial t}\,.$$

Step 1



$$\frac{\partial\vec{W}}{\partial t} = -\frac{1}{\Omega}\left[\oint_{\partial\Omega}(\vec{F}_c - \vec{F}_v)\,dS - \int_{\Omega}\vec{Q}\,d\Omega\right].$$

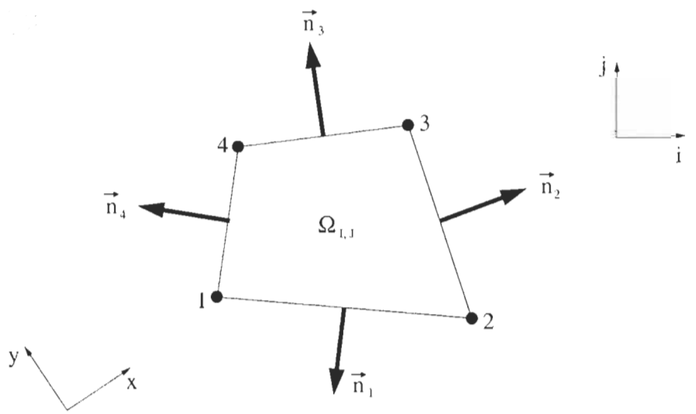
Step 3

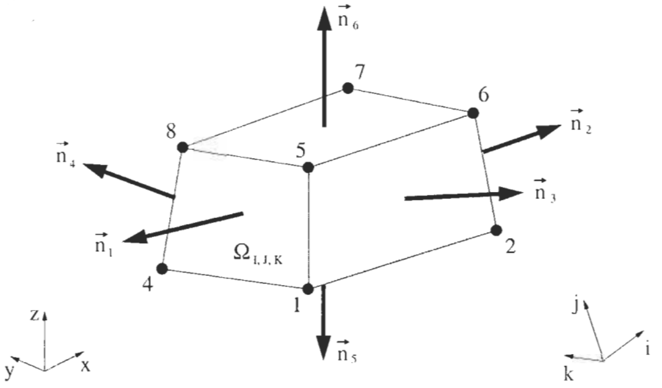
Step 4

$$\frac{d\vec{W}_{I,J,K}}{dt} = -\frac{1}{\Omega_{I,J,K}}\left[\sum_{m=1}^{N_F}(\vec{F}_c - \vec{F}_v)_m\,\Delta S_m - (\vec{Q}\Omega)_{I,J,K}\right]$$

$$\frac{d\vec{W}_{I,J,K}}{dt} = -\frac{1}{\Omega_{I,J,K}}\,\vec{R}_{I,J,K}$$

Geometrical Quantities of a Control Volume





Volume of cell

$$\Omega_{I,J} = \frac{b}{2}\left[(x_1 - x_3)(y_2 - y_4) + (x_4 - x_2)(y_1 - y_3)\right]$$

Face area of cell

$$\vec{S}_m = \begin{bmatrix} S_{x,m} \\ S_{y,m} \end{bmatrix} = \vec{n}_m \, \Delta S_m \,.$$

$$\vec{S}_1 = b\begin{bmatrix} y_2 - y_1 \\ x_1 - x_2 \end{bmatrix}$$

$$\vec{S}_2 = b\begin{bmatrix} y_3 - y_2 \\ x_2 - x_3 \end{bmatrix}$$

$$\vec{S}_3 = b\begin{bmatrix} y_4 - y_3 \\ x_3 - x_4 \end{bmatrix}$$

$$\vec{S}_4 = b\begin{bmatrix} y_1 - y_4 \\ x_4 - x_1 \end{bmatrix}.$$

$$\vec{n}_m = \frac{\vec{S}_m}{\Delta S_m}$$

$$\Delta S_m = |\vec{S}_m| = \sqrt{S_{x,m}^2 + S_{y,m}^2}\,.$$

Face area of cell

$$\Delta x_A = x_8 - x_1\,, \quad \Delta x_B = x_5 - x_4\,,$$

$$\Delta y_A = y_8 - y_1\,, \quad \Delta y_B = y_5 - y_4\,,$$

$$\Delta z_A = z_8 - z_1\,, \quad \Delta z_B = z_5 - z_4$$

$$\vec{S}_1 = \frac{1}{2}\begin{bmatrix} \Delta y_A\,\Delta z_B - \Delta z_A\,\Delta y_B \\ \Delta z_A\,\Delta x_B - \Delta x_A\,\Delta z_B \\ \Delta x_A\,\Delta y_B - \Delta y_A\,\Delta x_B \end{bmatrix}.$$

$$\Delta S_m = \sqrt{S_{x,m}^2 + S_{y,m}^2 + S_{z,m}^2}$$

Volume of cell

$$\Omega_{I,J,K} = \frac{1}{3}\sum_{m=1}^{m=6}(\vec{r}_{\rm mid}\cdot\vec{S})_m$$

$$\vec{r}_{\rm mid,1} = \frac{1}{4}(\vec{r}_1 + \vec{r}_5 + \vec{r}_8 + \vec{r}_4)$$

Discretization of Convective Terms

- Various methods available for discretization are

❑ Central,

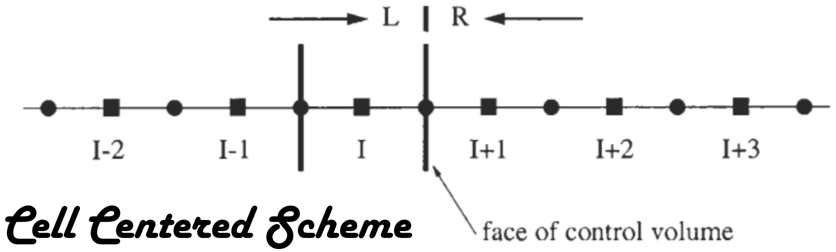
❑ Flux-Vector Splitting

❑ Flux-difference Splitting

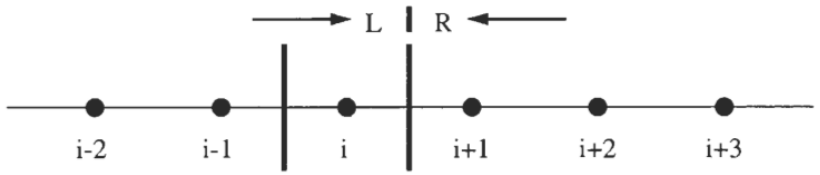
❑ Total Variation Diminishing (TVD) and

❑ Fluctuation splitting

Left & Right States and Stencil



Cell Centered Scheme



Cell Vertex Scheme

Central Scheme

- *Uses same number of cells left and right of interface*
- *Flux is "centered" by symmetric cell distribution across cell interface*

Upwinding

- *Uses cells based on information propagation direction based on characteristics*
- *Asymmetric usage of cells across cell interface*

$$U_R = U_{I+1} - \frac{\epsilon}{4}\left[(1+\hat{\kappa})\Delta_- + (1-\hat{\kappa})\Delta_+\right] U_{I+1}$$

$$U_L = U_I \quad + \frac{\epsilon}{4}\left[(1+\hat{\kappa})\Delta_+ + (1-\hat{\kappa})\Delta_-\right] U_I \,.$$

$$\Delta_+ U_I = U_{I+1} - U_I$$

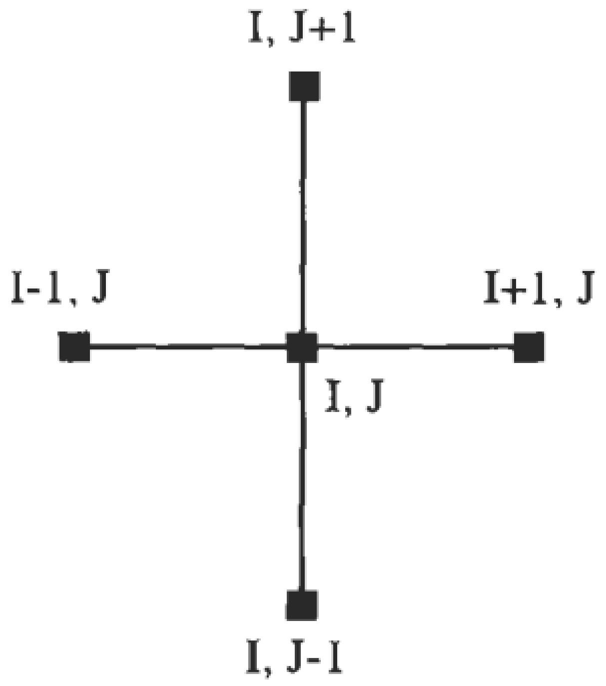
$$\Delta_- U_I = U_I - U_{I-1} \,.$$

Based on values of ϵ *and* $\hat{\kappa}$ *several upwind schemes can be defined such as*

- *First order upwind*
- *Second order upwind*
- *Upwind biased linear interpolation*
- *Second order central Scheme*

Union of cells involved in computation of derived quantities is Stencil

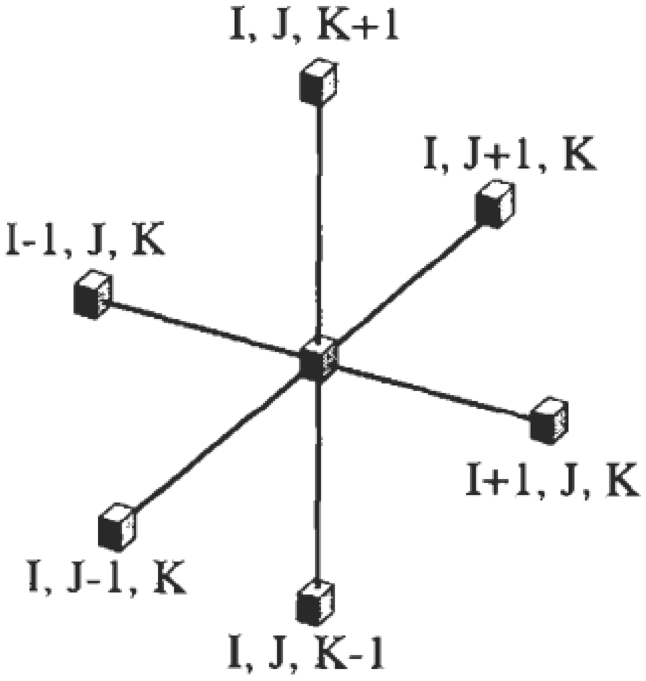
(a)



2D

$$(I, J) \quad (I+1, J) \quad (I, J+1) \quad (I-1, J) \quad (I, J-1)$$

(b)



3D

$$(I, J, K) \quad (I+1, J, K) \quad (I, J+1, K) \quad (I-1, J, K)$$
$$(I, J-1, K) \quad (I, J, K-1) \quad (I, J, K+1) .$$

Flux Vector Splitting Scheme

1. Van Leer's Scheme

$$\vec{F}_c = \vec{F}_c^{+} + \vec{F}_c^{-} ,$$

$$(M_n)_{I+1/2} = \left(\frac{V}{c}\right)_{I+1/2} ,$$

$$(M_n)_{I+1/2} = M_L^{+} + M_R^{-} ,$$

$$M_L^{+} = \begin{cases} M_L & \text{if } M_L \geq +1 \\ \frac{1}{4}(M_L+1)^2 & \text{if } |M_L| < 1 \\ 0 & \text{if } M_L \leq -1 \end{cases} ,$$

$$M_R^{-} = \begin{cases} 0 & \text{if } M_R \geq +1 \\ \frac{1}{4}(M_R-1)^2 & \text{if } |M_R| < 1 \\ M_R & \text{if } M_R \leq -1 \end{cases} .$$

$$M_L = \frac{V_L}{c_L} , \quad M_R = \frac{V_R}{c_R} .$$

$$\vec{F}_c^{\pm} = \begin{bmatrix} f_{\text{mass}}^{\pm} \\ f_{\text{mass}}^{\pm}\left[n_x(-V \pm 2c)/\gamma + u\right] \\ f_{\text{mass}}^{\pm}\left[n_y(-V \pm 2c)/\gamma + v\right] \\ f_{\text{mass}}^{\pm}\left[n_z(-V \pm 2c)/\gamma + w\right] \\ f_{\text{energy}}^{\pm} \end{bmatrix} .$$

$|M_n| < 1$

$$f_{\text{mass}}^{+} = +\rho_L c_L \frac{(M_L+1)^2}{4}$$

$$f_{\text{mass}}^{-} = -\rho_R c_R \frac{(M_R-1)^2}{4}$$

$$f_{\text{energy}}^{\pm} = f_{\text{mass}}^{\pm}\left\{\frac{[(\gamma-1)V \pm 2c]^2}{2(\gamma^2-1)} + \frac{u^2+v^2+w^2-V^2}{2}\right\}_{L/R} .$$

$|M_n| \geq 1$

$$\vec{F}_c^{+} = \vec{F}_c \quad \vec{F}_c^{-} = 0 \quad \text{if } M_n \geq +1$$

$$\vec{F}_c^{+} = 0 \quad \vec{F}_c^{-} = \vec{F}_c \quad \text{if } M_n \leq -1 .$$

2. Advection Upstream Splitting Method (AUSM)

$$\vec{F}_c = V \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho H \end{bmatrix} + \begin{bmatrix} 0 \\ n_x p \\ n_y p \\ n_z p \\ 0 \end{bmatrix}.$$

$$(\bullet)_{L/R} = \begin{cases} (\bullet)_L & \text{if } M_{I+1/2} \geq 0 \\ (\bullet)_R & \text{otherwise} \end{cases}$$

Convective flux spilitting same as Van Leer

$$p_{I+1/2} = p_L^+ + p_R^-$$

$$p_L^+ = \begin{cases} p_L & \text{if } M_L \geq +1 \\ \dfrac{p_L}{4}(M_L+1)^2(2-M_L) & \text{if } |M_L| < 1 \\ 0 & \text{if } M_L \leq -1 \end{cases},$$

$$p_R^- = \begin{cases} 0 & \text{if } M_R \geq +1 \\ \dfrac{p_R}{4}(M_R-1)^2(2+M_R) & \text{if } |M_R| < 1 \\ p_R & \text{if } M_R \leq -1 \end{cases}.$$

Some other convective flux discretization schemes continued.....

Various Computational Schemes for Euler Equations

Central Schemes

1. **Combined Space-Time Integration**
   (a) **Explicit Schemes**
   - Lax-Friendrichs – First order (1954)
   - Lax-Wendroff – Second order (1960)

   (b) **Two-Step Explicit Schemes**
   - Richtmyer and Morton (1967)
   - MacCormack (1969)
   - LeRat and Peyret (1974)

   (c) **Implicit Schemes**
   - MacCormack (1981)
   - Casier, Deconinck, Hirsch (1983)
   - LeRat (1979, 1983)
2. **Separate Space-Time Integration**
   (a) **Implicit Schemes**
   - Briley and McDonald (1975)
   - Beam and Warming (1976)

   (b) **Explicit Schemes (Multistage Runge-Kutta)**
   - Jameson, Schmidt, Turkel (1981)

First Order Upwind Schemes

1. **Flux Vector Splitting**
   - Courant, Isaacson, and Reeves (1952)
   - Moretti (1979)
   - Steger and Warming (1981)
   - VanLeer (1982)
2. **Godunov Methods-Riemann Solvers**
   (a) **Exact Riemann Solvers**
   - Godunov (1959) – First order
   - VanLeer (1979) – Second order
   - Woodward and Colella (1984)
   - Ben-Artzi and Falcovitz (1984)

   (b) **Approximate Riemann Solvers**
   - Roe (1981)
   - Enquist and Osher (1980)
   - Osher (1982)
   - Harten, Lax, Van Leer (1983)

Second Order Upwind Schemes

1. **Extrapolation**
   (a) **Variable Extrapolation (MUSCL)**
   - Van Leer (1979)

   (b) **Flux Extrapolation**
   - Van Leer (1979)
2. **Explicit TVD Upwind**
   - VanLeer (1974)
   - Harten (1983)
   - Osher (1984)
   - Osher and Chakravarthy (1984)
3. **Implicit TVD Upwind**
   - Yee (1986)
4. **Central TVD Implicit or Explicit**
   - Davis (1984)
   - Roe (1985)
   - Yee (1985)
5. **Essentially Nonoscillatory Scheme**
   - Harten and Osher (1987)
6. **Flux Corrected Transport**
   - Boris and Book (1973)

Implementation of Central Scheme in OpenFoam: rhoCentralFoam

Features of rhoCentralFoam

- Finite Volume, density based solver.
- Non-staggered/collocated field variables.
- Central/Central-upwind scheme.
- Applicable on polyhedral Mesh.
- Suitable for high speed viscous flows.

Some of the Important Solution Methodology for High Speed Flows

- Notable methods for producing accurate non-oscillatory solutions are (convective terms discretization of governing equations)
- Monotone upstream-centred scheme for conservation laws
- Piece wise parabolic method (PPM)
- Essentially non-oscillatory scheme (ENO)
- Weighted ENO (WENO)

The above methods are complicated as they involve Riemann solvers, characteristic decomposition and Jacobian evaluation.

They are complex to implement !!

Compressible FVM: CFD

- To avoid the difficulty of implementation and to reduce cost of computation, Central schemes were devised which was first proposed by Nessyahu and Tadmor as second order accurate version of Lax-Friedrichs scheme.

- The more advanced versions of central scheme widely used to solve industrial problems are Kurganov-Tadmor (KT) and Kurganov-Noelle-Petrova (KNP) schemes

- These schemes are implemented in OpenFOAM package, with "rhoCentralFoam" as the solver name

Governing Equations : Differential Form

Continuity Equation

$$\frac{\partial \rho}{\partial t} + \nabla.[\boldsymbol{u}\rho] = 0$$

Momentum Equation

$$\frac{\partial(\rho\boldsymbol{u})}{\partial t} + \nabla.[\boldsymbol{u}(\rho\boldsymbol{u})] + \nabla p + \nabla.\boldsymbol{T} = 0$$

Conservation of Total Energy

$$\frac{\partial(\rho E)}{\partial t} + \nabla.[\boldsymbol{u}(\rho E)] + \nabla.(\boldsymbol{u}p) + \nabla.(\boldsymbol{T}\boldsymbol{u}) + \nabla.\boldsymbol{j} = 0$$

Compressible FVM: CFD

- Here ρ is the mass density **u** is the fluid velocity, p is the pressure,

Total energy density $E=e+0.5|\mathbf{u}|^2$ with 'e' as specific internal energy

j as the diffusion heat flux vector, **T** is the viscous shear stress tensor (defined positive in compression)

$\mathbf{T}=-2\mu\,\mathrm{dev}(\mathbf{D})$

μ is the dynamic viscosity (Sutherland's law or constant)

$\boldsymbol{D} = \frac{1}{2}[\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^{\boldsymbol{T}}]$ (Deformation gradient tensor)

Deviatoric component $\mathrm{dev}(\mathbf{D})=\mathbf{D} - \left(\frac{1}{3}\right) tr(\boldsymbol{D})\boldsymbol{I}$ $\quad \boldsymbol{I}$ is the unit tensor

The diffusion heat flux vector defined by

$j = -k\nabla T$ (Fourier's Law)

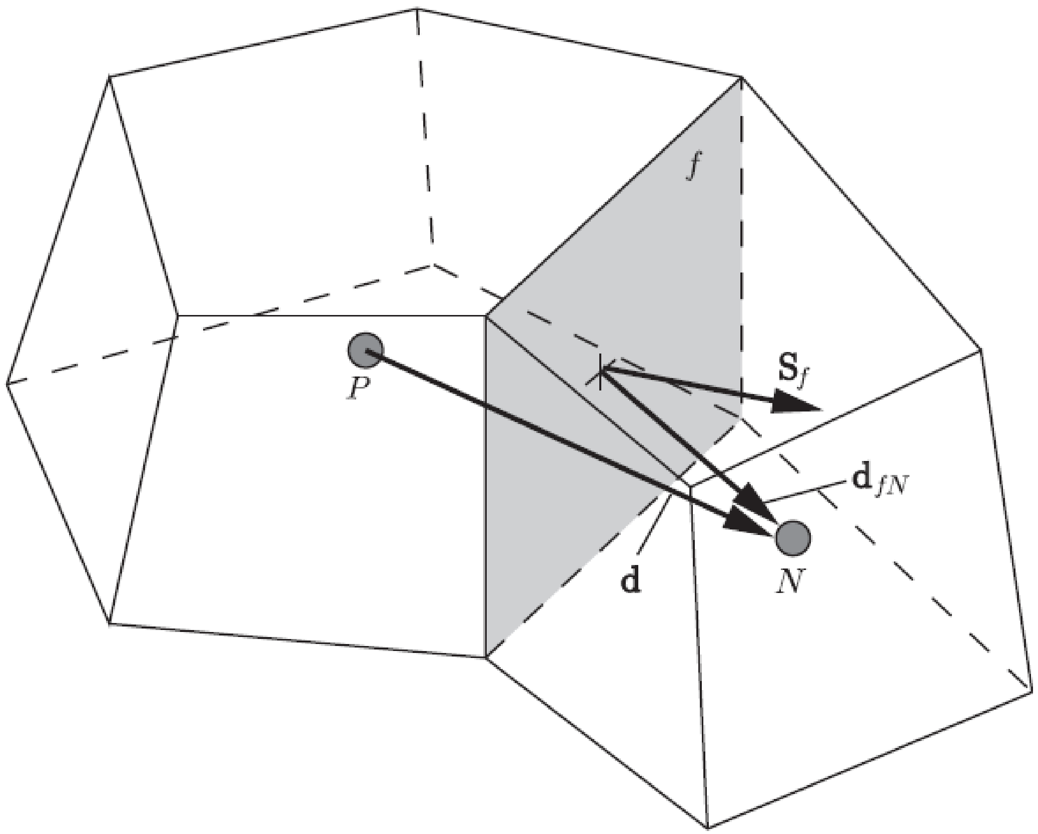
k= thermal conductivity

T= temperature

For calorically perfect gas, $p=\rho RT$ and $e=CvT=(\gamma-1)RT$

$\gamma=Cp/Cv$, R is characteristic gas constant

Computational Method



d_{fN} vector connecting face centre and cell centre of neighbouring cell

d Vector connecting cell centre **P** and cell centre **N**

$\boldsymbol{S_f}$ face area vector (outwards is positive)

All variables are stored at cell centres (**P**,**N**….)

Start

↓

Governing Equation in differential form to Integral Form on cells (Gauss divergence theorem)

↓

For flux calculation,
Cell center data interpolation onto face center
Use limited interpolation schemes (eg. TVD based)

↓

Use KT/KNP method to obtain final algebraic discretized governing equations on each cells

↓

Stop

The KT/KNP convective flux discretization method shall be explained in upcoming slides……

Procedure to obtain discretized algebraic governing equation on collocated finite volume grid

The Algebraic Discretized Equation

- The general variable $\boldsymbol{\psi}$ (tensor of any rank) can be convected in the flow field, the general term associated with convection in GE is:

$$\nabla.(\boldsymbol{u}\boldsymbol{\psi})$$

Application of Gauss divergence theorem

$$\iiint \nabla.(\boldsymbol{u}\boldsymbol{\psi})dV = \iint d\boldsymbol{S}.[\boldsymbol{u}\boldsymbol{\psi}] = \sum_f S_f.[\boldsymbol{u}_f\boldsymbol{\psi}_f] = \sum_f \phi_f \boldsymbol{\psi}_f$$

Here, ϕ_f is the volumetric flux

Convective Terms

- The general variable **ψ** (tensor of any rank) can be convected in the flow field, the general term associated with convection in GE is:

$$\nabla.(\boldsymbol{u}\psi)$$

Application of Gauss divergence theorem

$$\iiint \nabla.(\boldsymbol{u}\psi)dV = \iint d\boldsymbol{S}.[\boldsymbol{u}\psi] = \sum_f S_f.[\boldsymbol{u}_f\boldsymbol{\psi}_f] = \sum_f \phi_f\boldsymbol{\psi}_f$$

Here, ϕ_f is the volumetric flux

Compressible FVM: CFD

$$\alpha = \begin{cases} \dfrac{1}{2} & for\ KT\ method \\ \dfrac{\lambda_{f+}}{\lambda_{f+} + \lambda_{f-}} & for\ KNP\ method \end{cases}$$

$$\lambda_{f+} = \max(c_{f+}|\boldsymbol{S_f}| + \phi_{f+}, c_{f-}|\boldsymbol{S_f}| + \phi_{f-}, 0)$$
$$\lambda_{f-} = \max(c_{f+}|\boldsymbol{S_f}| - \phi_{f+}, c_{f-}|\boldsymbol{S_f}| - \phi_{f-}, 0)$$

$$\omega_f = \begin{cases} \alpha \max(\lambda_{f+}, \lambda_{f-}) & for\ KT\ method \\ \alpha(1-\alpha)(\lambda_{f+} + \lambda_{f-}) & for\ KNP\ method \end{cases}$$

Gradient Terms

$$\iiint \nabla(\boldsymbol{\psi})dV = \iint d\boldsymbol{S}\boldsymbol{\psi} = \sum_f \boldsymbol{S}_f \boldsymbol{\psi}_f$$

$$\sum_f \boldsymbol{S}_f \boldsymbol{\psi}_f = \sum_f [\alpha \boldsymbol{S}_f \boldsymbol{\psi}_{f^+} + (1-\alpha)\boldsymbol{S}_f \boldsymbol{\psi}_{f^-}]$$

Laplacian Terms

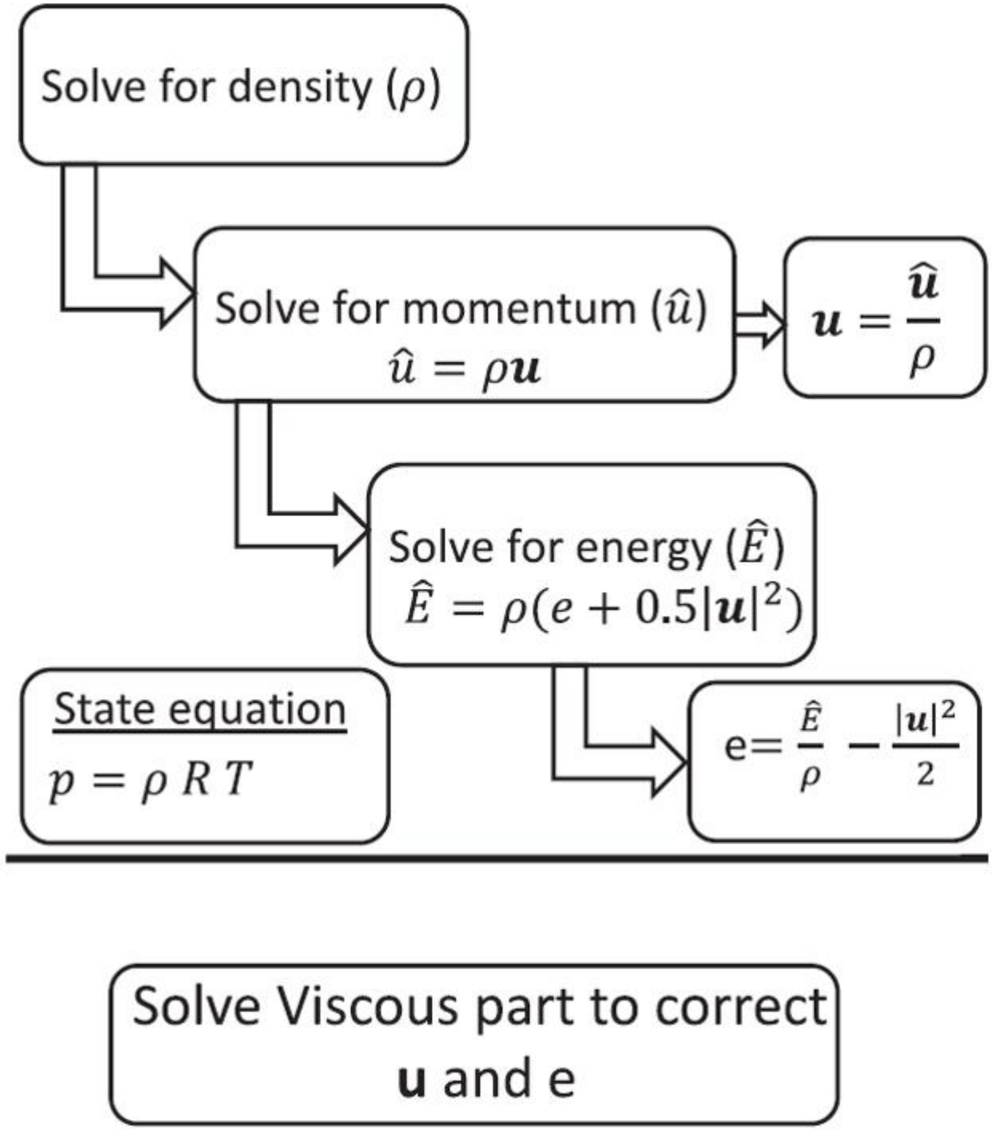
$$\iiint \nabla.(\Gamma\nabla\boldsymbol{\psi})dV = \iint d\boldsymbol{S}(\Gamma\nabla\boldsymbol{\psi}) = \sum_f \Gamma_f \boldsymbol{S}_f.(\nabla\boldsymbol{\psi})_f$$

For Non-orthogonal Mesh,

$$\boldsymbol{S}_f.(\nabla\boldsymbol{\psi})_f = A(\boldsymbol{\psi}_N - \boldsymbol{\psi}_P) + \boldsymbol{a}.(\nabla\boldsymbol{\psi})_f$$

$$A = {|\boldsymbol{S}_f|^2}/{\boldsymbol{S}_f.\boldsymbol{d}}$$

$$\boldsymbol{a} = \boldsymbol{S}_f - A\boldsymbol{d}$$



Solution Algorithm of rhoCentralFoam

OpenFoam File : Implementation

```
surfaceScalarField rho_pos(interpolate(rho, pos));
surfaceScalarField rho_neg(interpolate(rho, neg));

surfaceVectorField rhoU_pos(interpolate(rhoU, pos, U.name()));
surfaceVectorField rhoU_neg(interpolate(rhoU, neg, U.name()));

volScalarField rPsi("rPsi", 1.0/psi);
surfaceScalarField rPsi_pos(interpolate(rPsi, pos, T.name()));
surfaceScalarField rPsi_neg(interpolate(rPsi, neg, T.name()));

surfaceScalarField e_pos(interpolate(e, pos, T.name()));
surfaceScalarField e_neg(interpolate(e, neg, T.name()));

surfaceVectorField U_pos("U_pos", rhoU_pos/rho_pos);
surfaceVectorField U_neg("U_neg", rhoU_neg/rho_neg);

surfaceScalarField p_pos("p_pos", rho_pos*rPsi_pos);
surfaceScalarField p_neg("p_neg", rho_neg*rPsi_neg);

surfaceScalarField phiv_pos("phiv_pos", U_pos & mesh.Sf());
// Note: extracted out the orientation so becomes unoriented
phiv_pos.setOriented(false);
surfaceScalarField phiv_neg("phiv_neg", U_neg & mesh.Sf());
phiv_neg.setOriented(false);
```

```
volScalarField c("c", sqrt(thermo.Cp()/thermo.Cv()*rPsi));
surfaceScalarField cSf_pos
(
    "cSf_pos",
    interpolate(c, pos, T.name())*mesh.magSf()
);

surfaceScalarField cSf_neg
(
    "cSf_neg",
    interpolate(c, neg, T.name())*mesh.magSf()
);
```

Interpolation of variables from left and right side of interface of control volume

Compressible FVM: CFD

```
surfaceScalarField ap
(
    "ap",
    max(max(phiv_pos + cSf_pos, phiv_neg + cSf_neg), v_zero)
);

surfaceScalarField am
(
    "am",
    min(min(phiv_pos - cSf_pos, phiv_neg - cSf_neg), v_zero)
);

surfaceScalarField a_pos("a_pos", ap/(ap - am));

surfaceScalarField amaxSf("amaxSf", max(mag(am), mag(ap)));

surfaceScalarField aSf("aSf", am*a_pos);

if (fluxScheme == "Tadmor")
{
    aSf = -0.5*amaxSf;
    a_pos = 0.5;
}

surfaceScalarField a_neg("a_neg", 1.0 - a_pos);

phiv_pos *= a_pos;
phiv_neg *= a_neg;

surfaceScalarField aphiv_pos("aphiv_pos", phiv_pos - aSf);
surfaceScalarField aphiv_neg("aphiv_neg", phiv_neg + aSf);
```

Calculations for scalar dissipation coefficient For KT or KNP scheme based on user input flag (fluxScheme==“Tadmor”)

```
phi = aphiv_pos*rho_pos + aphiv_neg*rho_neg;

surfaceVectorField phiU(aphiv_pos*rhoU_pos + aphiv_neg*rhoU_neg);
// Note: reassembled orientation from the pos and neg parts so becomes
// oriented
phiU.setOriented(true);

surfaceVectorField phiUp(phiU + (a_pos*p_pos + a_neg*p_neg)*mesh.Sf());

surfaceScalarField phiEp
(
    "phiEp",
    aphiv_pos*(rho_pos*(e_pos + 0.5*magSqr(U_pos)) + p_pos)
  + aphiv_neg*(rho_neg*(e_neg + 0.5*magSqr(U_neg)) + p_neg)
  + aSf*p_pos - aSf*p_neg
);
```

Calculation of Convective Fluxes of Governing Equations

Compressible FVM: CFD

```
volScalarField muEff("muEff", turbulence->muEff());
volTensorField tauMC("tauMC", muEff*dev2(Foam::T(fvc::grad(U))));
```

Calculation of viscosity and shear stress tensor

```
// --- Solve density
solve(fvm::ddt(rho) + fvc::div(phi));
```

Solves density equation

```
// --- Solve momentum
solve(fvm::ddt(rhoU) + fvc::div(phiUp));
```

Predictor step of momentum equation

```
U.ref() =
    rhoU()
   /rho();
U.correctBoundaryConditions();
rhoU.boundaryFieldRef() == rho.boundaryField()*U.boundaryField();
```

Update Boundary Condition on U

```
if (!inviscid)
{
    solve
    (
        fvm::ddt(rho, U) - fvc::ddt(rho, U)
      - fvm::laplacian(muEff, U)
      - fvc::div(tauMC)
    );
    rhoU = rho*U;
}
```

Viscous Corrector step of momentum equation

```
// --- Solve energy
surfaceScalarField sigmaDotU
(
    "sigmaDotU",
    (
        fvc::interpolate(muEff)*mesh.magSf()*fvc::snGrad(U)
      + fvc::dotInterpolate(mesh.Sf(), tauMC)
    )
  & (a_pos*U_pos + a_neg*U_neg)
```

Calculation of work done by viscous forces

```
solve
(
    fvm::ddt(rhoE)
  + fvc::div(phiEp)
  - fvc::div(sigmaDotU)
);
```

Predictor step of energy equation

```
e = rhoE/rho - 0.5*magSqr(U);
e.correctBoundaryConditions();
thermo.correct();
```

Update Boundary Condition on e
Extract temperature from e

```
rhoE.boundaryFieldRef() ==
    rho.boundaryField()*
    (
        e.boundaryField() + 0.5*magSqr(U.boundaryField())
    );
```

Update boudanry field of rhoE

```
if (!inviscid)
{
    solve
    (
        fvm::ddt(rho, e) - fvc::ddt(rho, e)
      - fvm::laplacian(turbulence->alphaEff(), e)
    );
    thermo.correct();
    rhoE = rho*(e + 0.5*magSqr(U));
}
```

Viscous Corrector step of energy equation

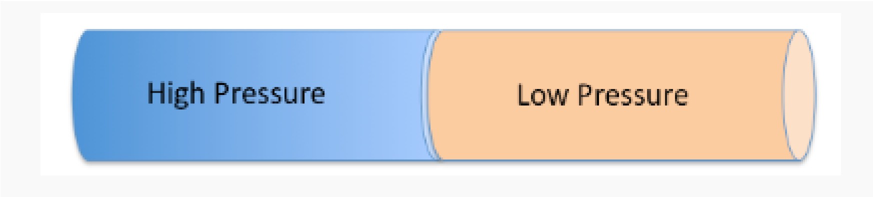
```
p.ref() =
    rho()
   /psi();
p.correctBoundaryConditions();
rho.boundaryFieldRef() == psi.boundaryField()*p.boundaryField();

turbulence->correct();

runTime.write();

runTime.printExecutionTime(Info);
```

Update value of pressure field and continue to the next time loop



- **The Sod's Shock Tube Problem**

Initial Conditions

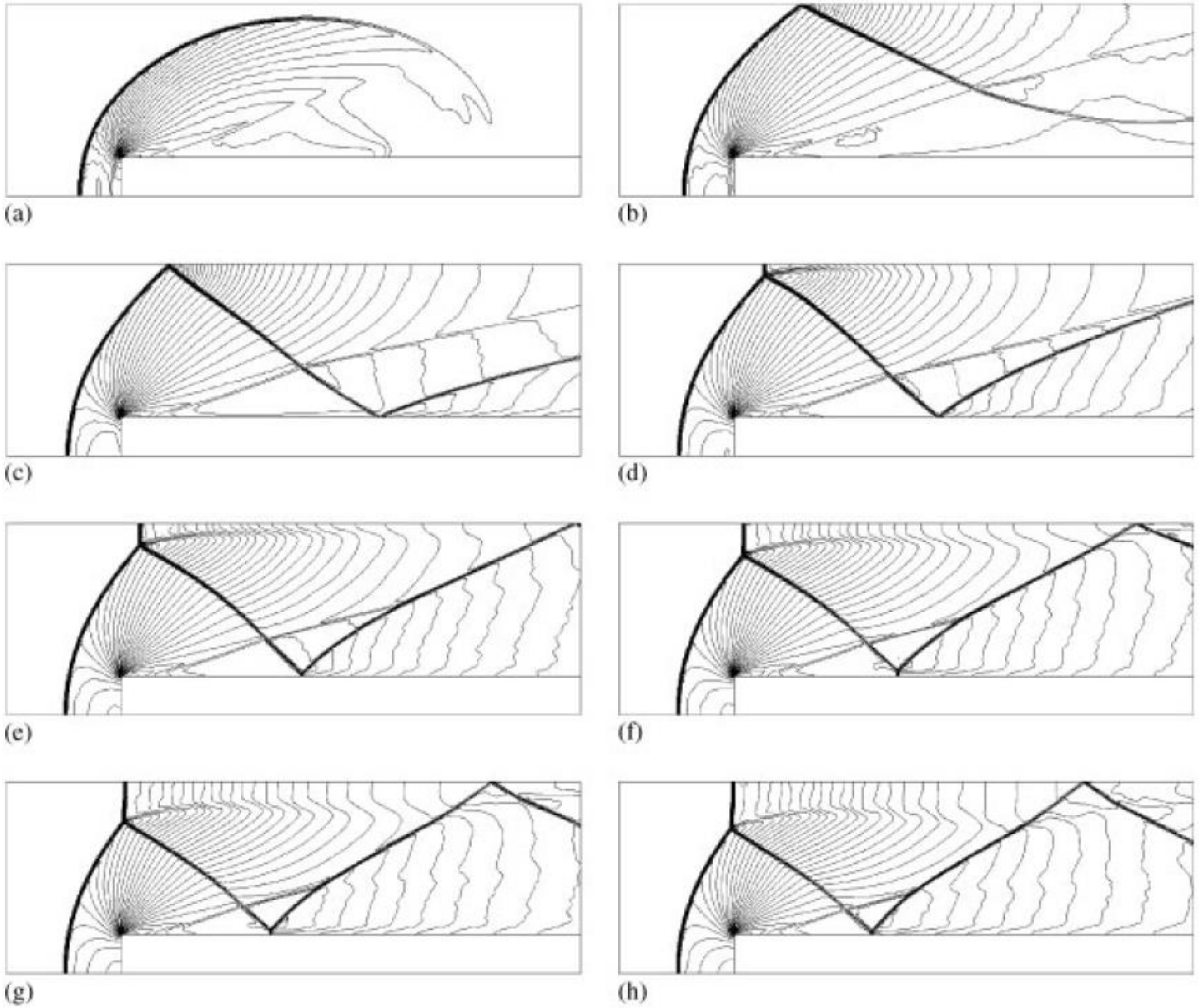
The state on the left side of the diaphragm is denoted by L

$$\rho_L = 1.0\ \mathrm{kg/m^3},\ p_L = 100000\ \mathrm{Pa},\ T_L = 348.4\ \mathrm{K}$$

The state on the left side of the diaphragm is denoted by L

$$\rho_R = 0.125\ \mathrm{kg/m^3},\ p_R = 10000\ \mathrm{Pa},\ T_R = 278.7\ \mathrm{K}$$

- Forward Step (Woodward and Collela)



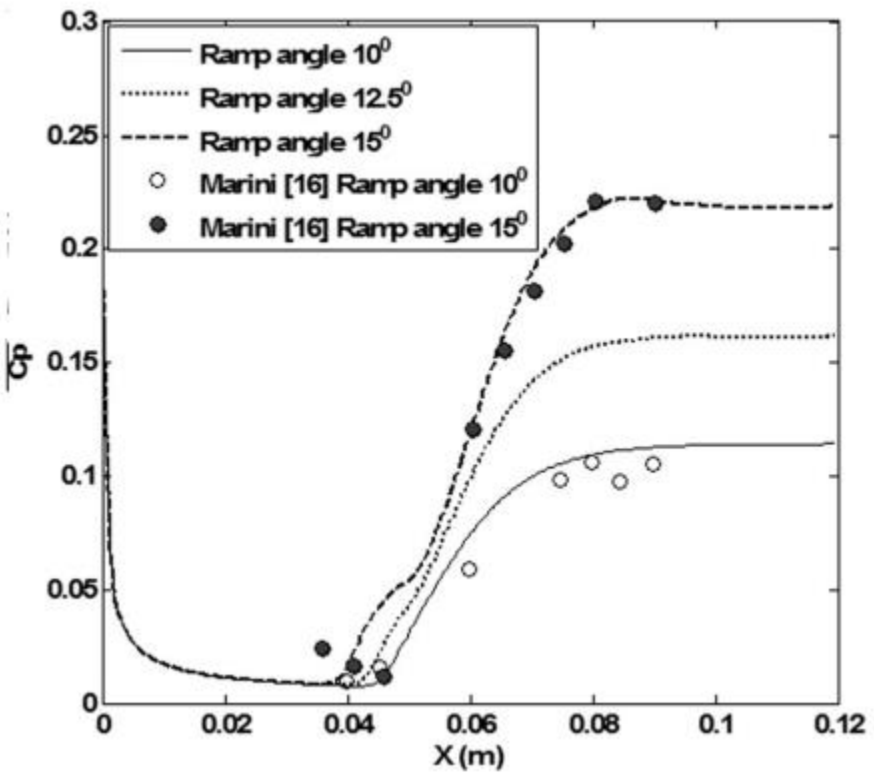
Unsteady flow over the forward step
placed in supersonic stream of air

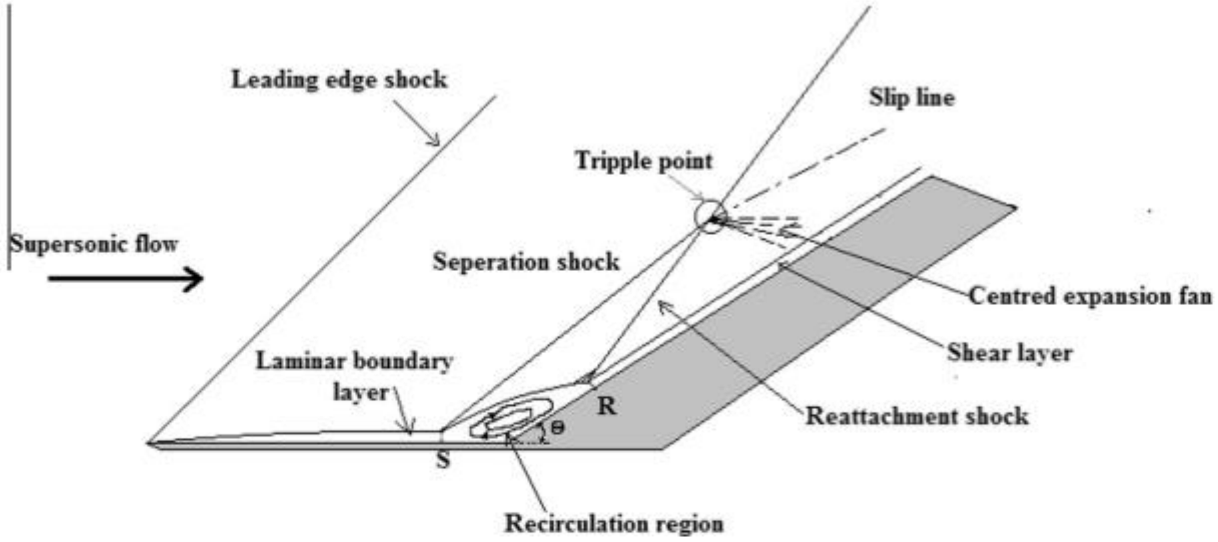
Mach 3 flow

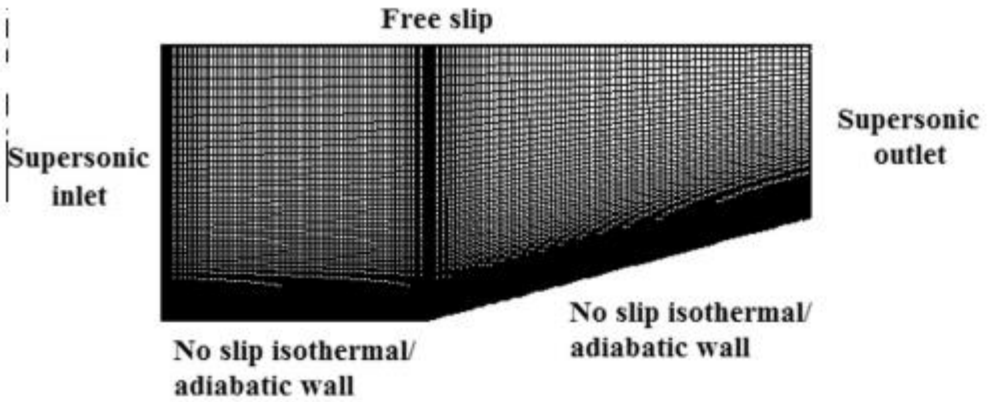
A standard benchmark test case to access the
Performance of various numerical schemes

Demo examples in OF: CFD

- Shock wave boundary layer interaction on compression corner







U=1380m/s
P=191.4345 Pa
T=131.7
Twall = 300 K


danke
謝謝
ngiyabonga
teşekkür ederim
спасибо
thank you
gracias
dank je
mochchakkeram
dziękuje
bedankt
obrigado
sukriya
kop khun krap
go raibh maith agat
arigato
dakujem
terima kasih
grazie
мерси
merci
감사합니다


Q&A
HOW
Why
Who
What
Where
When