8th International OpenFOAM® Workshop

Development and Validation of a Density-Based Implicit Solver Using LU-SGS Algorithm

JaeHeung Gill[1], ByoungYun Kim[1], JiHong Kim[2], HoonBum Shin[2], SungKi Jung[2] and KyuHong Kim[3]

[1] NEXTFoam Co., Ltd.

[2] Korea Aerospace Indutries, LTD.

[3] Seoul National University

Outline

1. **Background**
2. **Implicit Finite Volume Discretization**
3. **LU-SGS Algorithm**
4. **Results**
5. **Concluding Remarks**

Outline

1. Background

2. Implicit Finite Volume Discretization

3. LU-SGS Algorithm

4. Results

5. Concluding Remarks

Background

- Based on *DensityBasedTurbo* by Oliver Borm
  - Density Based Coupled Algorithm
  - Explicit Time Integration
  - Godunov Type Flux Schemes
  - Multi-Dimensional Slope Limiter
  - Local Time Stepping
  - Steady & Transient Solvers

- We have focused on steady state solver
  - Implementation of implicit time integration
  - Implementation of far-field boundary condition
    - Utilizing riemann invariants

Outline

1. Background

2. Implicit Finite Volume Discretization

3. LU-SGS Algorithm

4. Results

5. Concluding Remarks

Implicit Finite Volume Discretization

- Favre-Averaged Navier-Stokes Equations in Integral Form

$$\int_V \frac{\partial \vec{W}}{\partial t} dV + \oint_S \left(\vec{F}_c - \vec{F}_v \right) dS = 0$$

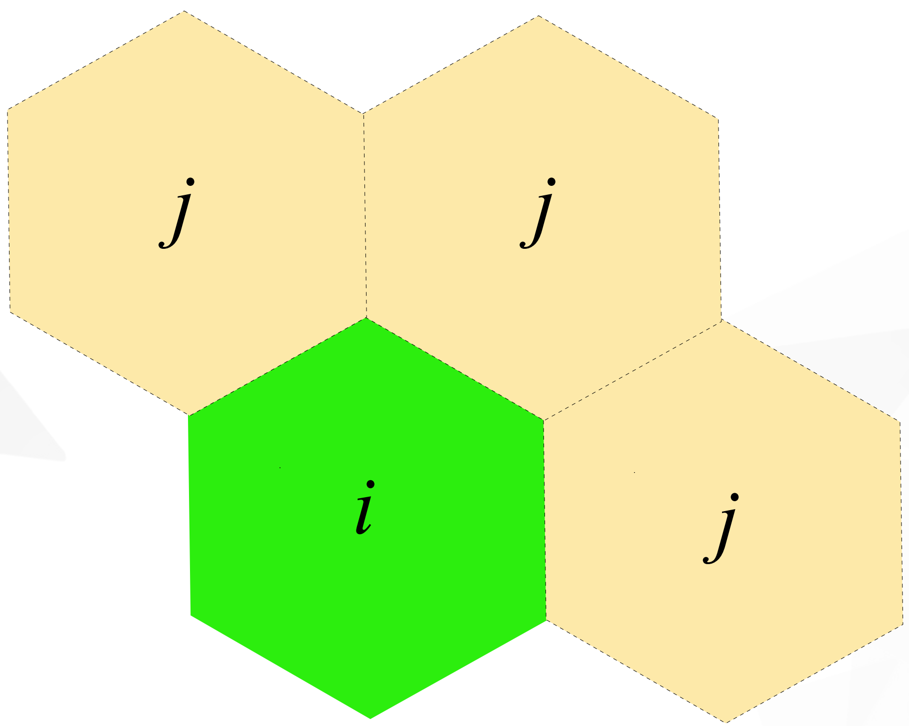
$$\vec{W} = \begin{bmatrix} \rho \\ \rho \vec{U} \\ \rho E \end{bmatrix} \qquad \vec{F}_c = \begin{bmatrix} \left(\rho \vec{U}\right)_f \cdot \vec{n} \\ \left(\rho \vec{U} \otimes \vec{U} + p \bar{I}\right)_f \cdot \vec{n} \\ \left(\rho H \vec{U}\right)_f \cdot \vec{n} \end{bmatrix}$$

$$\vec{F}_v = \begin{bmatrix} 0 \\ \bar{\tau}_f \cdot \vec{n} \\ \left(\bar{\tau} \cdot \vec{n}\right)_f \cdot \vec{n} + \left(\rho \alpha_{eff} \nabla h\right)_f \cdot \vec{n} + \left\{ \left(\mu + \frac{\mu_t}{\sigma_k} \right) \nabla k \right\}_f \cdot \vec{n} \end{bmatrix}$$

Implicit Finite Volume Discretization

- Spatial Discretization

$$V_i \frac{\partial \vec{W}_i}{\partial t} + \sum_{j \in N(i)} \left(\vec{F}_{c,ij} - \vec{F}_{v,ij} \right) S_{ij} = 0$$



Implicit Finite Volume Discretization

- Time Integration
  - Explicit

$$\frac{V_i}{\Delta t_i}\left(\vec{W}_i^{n+1}-\vec{W}_i^{n}\right)+\sum_{j\in N(i)}\left(\vec{F}_{c,ij}^{n}-\vec{F}_{v,ij}^{n}\right)S_{ij}=0$$

  - Implicit (Backward Euler)

$$\frac{V_i}{\Delta t_i}\left(\vec{W}_i^{n+1}-\vec{W}_i^{n}\right)+\sum_{j\in N(i)}\left(\vec{F}_{c,ij}^{n+1}-\vec{F}_{v,ij}^{n+1}\right)S_{ij}=0$$

Implicit Finite Volume Discretization

- **Linearizing Flux Vector**
  - Linearizing both convective and viscous fluxes using Taylor's series expansion.

$$\vec{F}_{ij}^{n+1} \approx \vec{F}_{ij}^{n} + \left(\frac{\partial \vec{F}}{\partial \vec{W}} \right)_{ij} \Delta \vec{W}_{ij}^{n}$$

Result in

$$\frac{V_i}{\Delta t_i} \Delta \vec{W}_i^n + \sum_{j \in N(i)} \left(A_{c,ij} - A_{v,ij} \right) \Delta \vec{W}_{ij}^n S_{ij} = -Res_i^n$$

where

$$\Delta \vec{W}_i^n = \vec{W}_i^{n+1} - \vec{W}_i^n$$

$$A_c = \frac{\partial \vec{F}_c}{\partial \vec{W}} \quad : \quad Convective\ Flux\ Jacobian$$

$$A_v = \frac{\partial \vec{F}_v}{\partial \vec{W}} \quad : \quad Viscous\ Flux\ Jacobian$$

Outline

1. Background
2. Implicit Finite Volume Discretization
3. **LU-SGS Algorithm**
4. Results
5. Concluding Remarks

Lower-Upper Symmetric Gauss-Seidel(LU-SGS) Algorithm

- Evaluation of Flux Jacobian
  - Steger-Warming's Flux Vector Splitting for Convective Flux
  - Thin Shear Layer Approximation(TSL) for Viscous Flux

$$\frac{V_i}{\Delta t_i}\Delta \vec{W}_i^n + \sum_{j\in N(i)} \left(A_{c,i}^{+} + A_{v,i}^{*}\right)\Delta \vec{W}_i^n S_{ij} + \sum_{j\in N(i)} \left(A_{c,j}^{-} - A_{v,j}^{*}\right)\Delta \vec{W}_j^n S_{ij} = -Res_i^n$$

Lower-Upper Symmetric Gauss-Seidel(LU-SGS) Algorithm

- Split off-diagonal term into lower(owner) and upper(neighbor) part

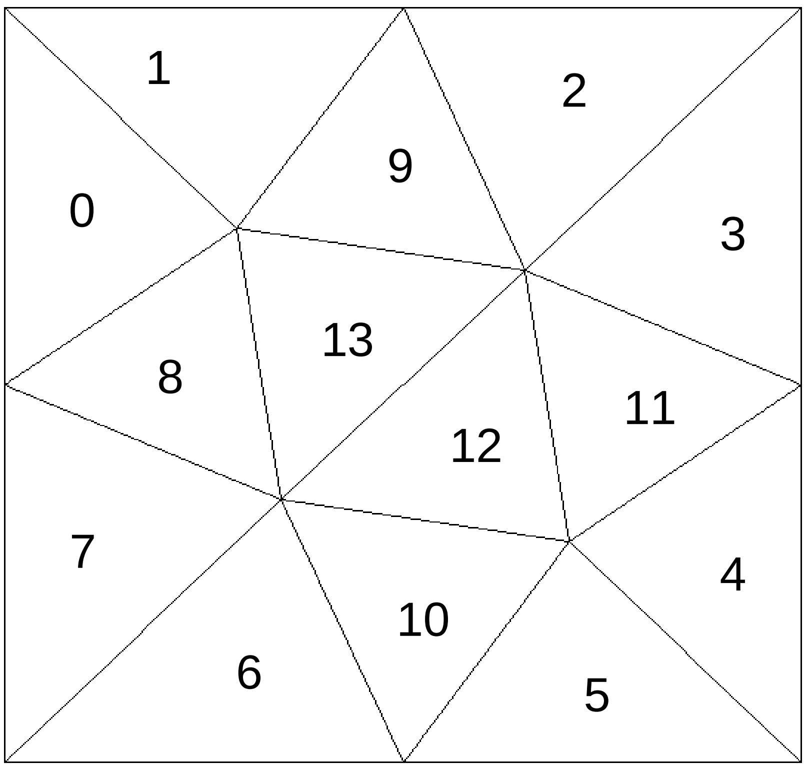
$$\frac{V_i}{\Delta t_i}\Delta \vec{W}_i^n + \sum_{j\in N(i)} \left(A_{c,i}^{+} + A_{v,i}^{*}\right)\Delta \vec{W}_i^n S_{ij}$$
$$+ \sum_{j\in L(i)} \left(A_{c,j}^{-} - A_{v,j}^{*}\right)\Delta \vec{W}_j^n S_{ij}$$
$$+ \sum_{j\in U(i)} \left(A_{c,j}^{-} - A_{v,j}^{*}\right)\Delta \vec{W}_j^n S_{ij} = -Res_i^n$$

  – Block matrix system

$$\left(D+L+U\right)\Delta W^n = -R^n$$

Lower-Upper Symmetric Gauss-Seidel(LU-SGS) Algorithm

- Block Matrix Example



$$
\begin{pmatrix}
D_0 & U_1 & 0 & 0 & 0 & 0 & 0 & 0 & U_8 & 0 & 0 & 0 & 0 & 0 \\
L_0 & D_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & U_9 & 0 & 0 & 0 & 0 \\
0 & 0 & D_2 & U_3 & 0 & 0 & 0 & 0 & 0 & U_9 & 0 & 0 & 0 & 0 \\
0 & 0 & L_2 & D_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & U_{11} & 0 & 0 \\
0 & 0 & 0 & 0 & D_4 & U_5 & 0 & 0 & 0 & 0 & 0 & U_{11} & 0 & 0 \\
0 & 0 & 0 & 0 & L_4 & D_5 & 0 & 0 & 0 & 0 & U_{10} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & D_6 & U_7 & 0 & 0 & U_{10} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & L_6 & D_7 & U_8 & 0 & 0 & 0 & 0 & 0 \\
L_0 & 0 & 0 & 0 & 0 & 0 & 0 & L_7 & D_8 & 0 & 0 & 0 & 0 & U_{13} \\
0 & L_1 & L_2 & 0 & 0 & 0 & 0 & 0 & 0 & D_9 & 0 & 0 & 0 & U_{13} \\
0 & 0 & 0 & 0 & 0 & L_5 & L_6 & 0 & 0 & 0 & D_{10} & 0 & U_{12} & 0 \\
0 & 0 & 0 & L_3 & L_4 & 0 & 0 & 0 & 0 & 0 & 0 & D_{11} & U_{12} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & L_{10} & L_{11} & D_{12} & U_{13} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & L_8 & L_9 & 0 & 0 & L_{12} & D_{13}
\end{pmatrix}
$$

$$D+L+U$$

Lower-Upper Symmetric Gauss-Seidel(LU-SGS) Algorithm

- Approximate Factorization

$$(D+L)D^{-1}(D+U)\Delta W^n = -R^n + LD^{-1}U\Delta W^n$$

  - Factorization error

$$\begin{aligned} &o(\Delta t^2) \quad \text{if} \quad \Delta t \ll \Delta x \\ &o(\Delta t) \quad \text{if} \quad \Delta t \gg \Delta x \end{aligned}$$

Lower-Upper Symmetric Gauss-Seidel(LU-SGS) Algorithm

- Invert the matrix in two steps
  - Forward sweep

$$\left(D+L\right)\Delta W^{*}=-R^{n}$$

$$\begin{pmatrix}
D_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
L_0 & D_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & D_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & L_2 & D_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & D_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & L_4 & D_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & D_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & L_6 & D_7 & 0 & 0 & 0 & 0 & 0 & 0 \\
L_0 & 0 & 0 & 0 & 0 & 0 & 0 & L_7 & D_8 & 0 & 0 & 0 & 0 & 0 \\
0 & L_1 & L_2 & 0 & 0 & 0 & 0 & 0 & 0 & D_9 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & L_5 & L_6 & 0 & 0 & 0 & D_{10} & 0 & 0 & 0 \\
0 & 0 & 0 & L_3 & L_4 & 0 & 0 & 0 & 0 & 0 & 0 & D_{11} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & L_{10} & L_{11} & D_{12} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & L_8 & L_9 & 0 & 0 & L_{12} & D_{13}
\end{pmatrix}
\begin{pmatrix}
\Delta W^*_0 \\ \Delta W^*_1 \\ \Delta W^*_2 \\ \Delta W^*_3 \\ \Delta W^*_4 \\ \Delta W^*_5 \\ \Delta W^*_6 \\ \Delta W^*_7 \\ \Delta W^*_8 \\ \Delta W^*_9 \\ \Delta W^*_{10} \\ \Delta W^*_{11} \\ \Delta W^*_{12} \\ \Delta W^*_{13}
\end{pmatrix}
= -\left[R^n\right]$$

Lower-Upper Symmetric Gauss-Seidel(LU-SGS) Algorithm

– Backward sweep

$$\left(D+U\right)\Delta W^{n}=D\,\Delta W^{*}$$

$$\begin{pmatrix}
D_0 & U_1 & 0 & 0 & 0 & 0 & 0 & 0 & U_8 & 0 & 0 & 0 & 0 & 0 \\
0 & D_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & U_9 & 0 & 0 & 0 & 0 \\
0 & 0 & D_2 & U_3 & 0 & 0 & 0 & 0 & 0 & U_9 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & D_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & U_{11} & 0 & 0 \\
0 & 0 & 0 & 0 & D_4 & U_5 & 0 & 0 & 0 & 0 & 0 & U_{11} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & D_5 & 0 & 0 & 0 & 0 & U_{10} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & D_6 & U_7 & 0 & 0 & U_{10} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & D_7 & U_8 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & D_8 & 0 & 0 & 0 & 0 & U_{13} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & D_9 & 0 & 0 & 0 & U_{13} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & D_{10} & 0 & U_{12} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & D_{11} & U_{12} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & D_{12} & U_{13} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & D_{13}
\end{pmatrix}
\begin{pmatrix}
\Delta W^n_0 \\ \Delta W^n_1 \\ \Delta W^n_2 \\ \Delta W^n_3 \\ \Delta W^n_4 \\ \Delta W^n_5 \\ \Delta W^n_6 \\ \Delta W^n_7 \\ \Delta W^n_8 \\ \Delta W^n_9 \\ \Delta W^n_{10} \\ \Delta W^n_{11} \\ \Delta W^n_{12} \\ \Delta W^n_{13}
\end{pmatrix}
= \left[D\,\Delta W^{*}\right]$$

Outline

1. Background

2. Implicit Finite Volume Discretization

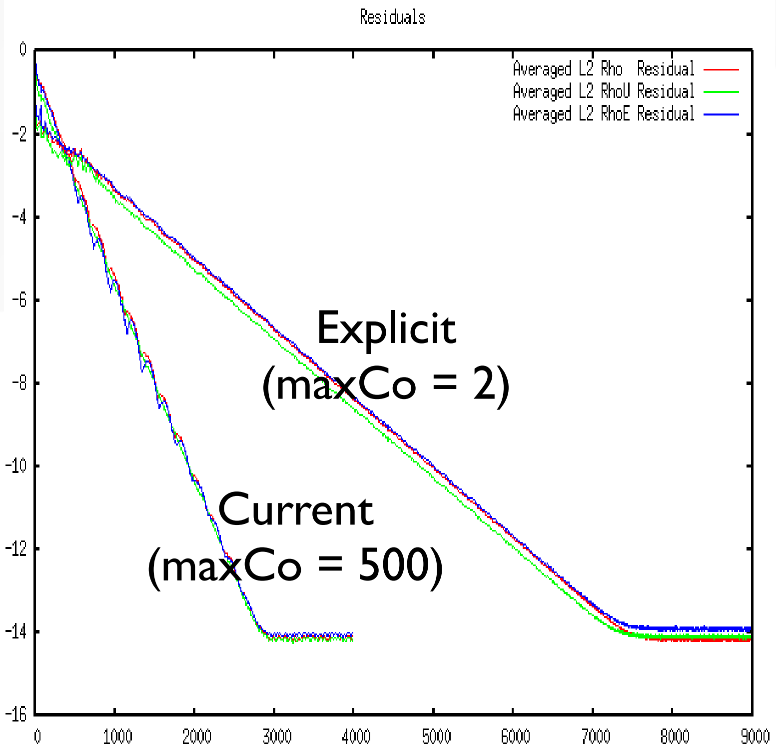
3. LU-SGS Algorithm

4. Results

5. Concluding Remarks

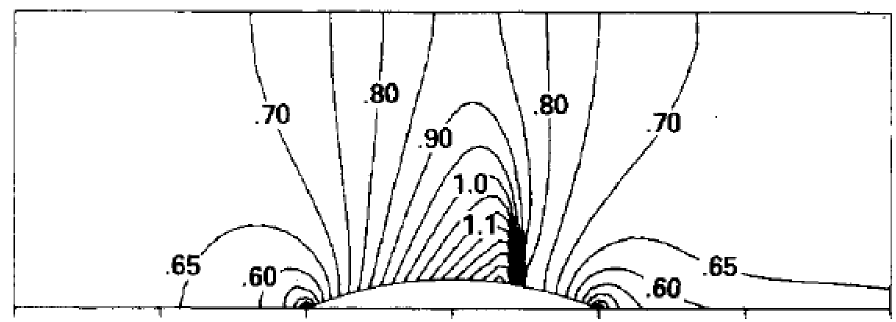
Results – Inviscid Flow(2D)

- Transonic Flow over a Bump in Channel
  - 65 X 17 grid
  - M = 0.675

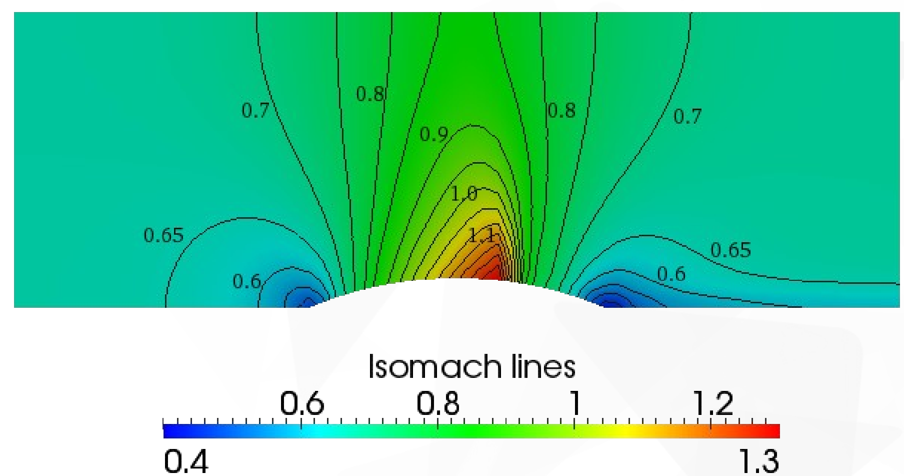


Residual history

From Ni, 1982

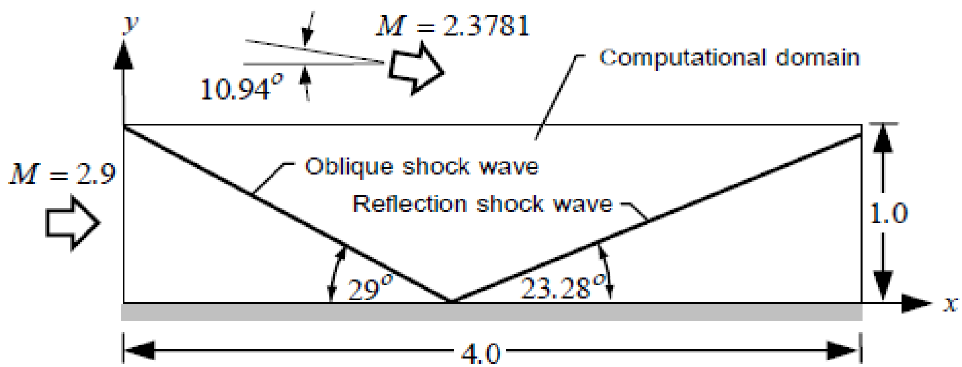


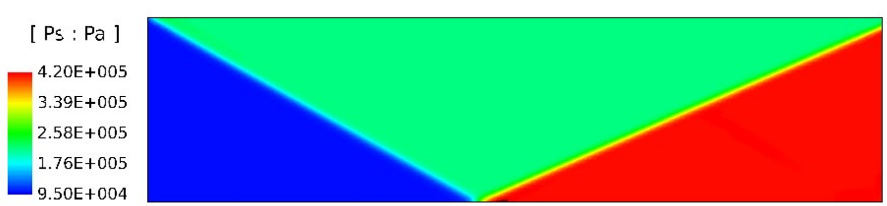
Current

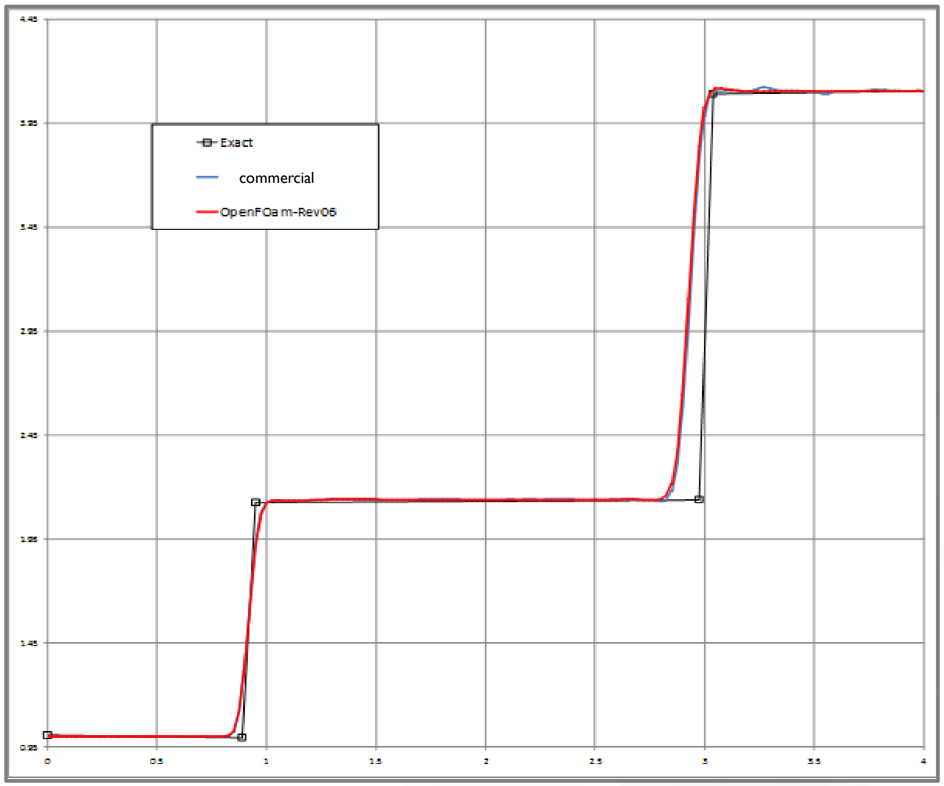


Results – Inviscid Flow(2D)

- Oblique Shock Reflection on a Plane Wall
  - Uniform triangular grid
  - $M_\infty = 2.9$

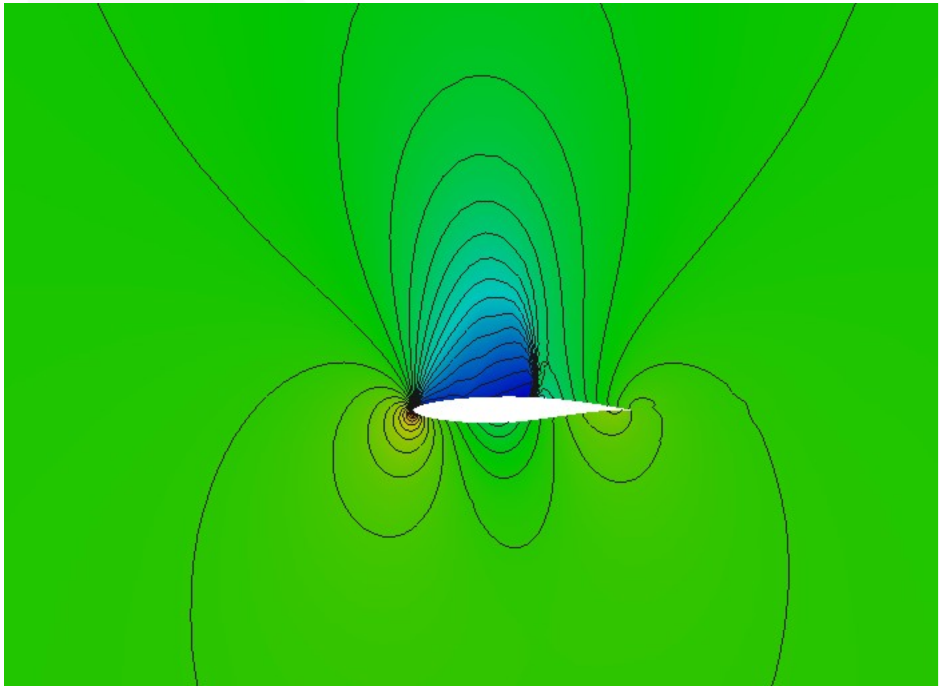


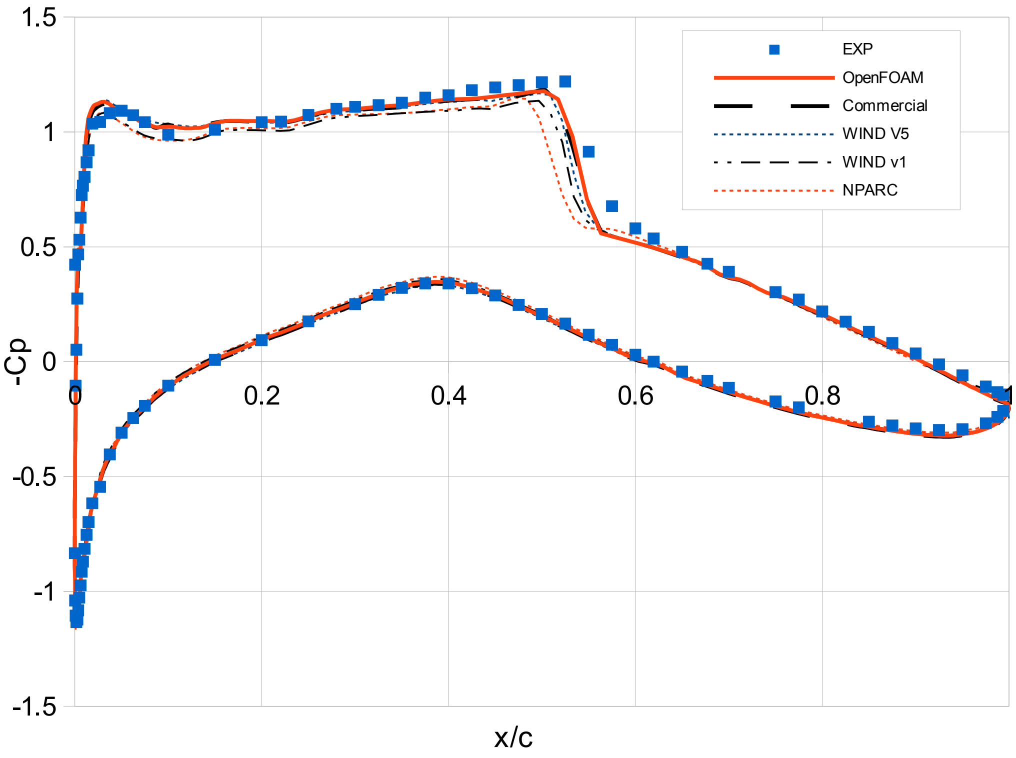




Results – Viscous Turbulent Flow(2D)

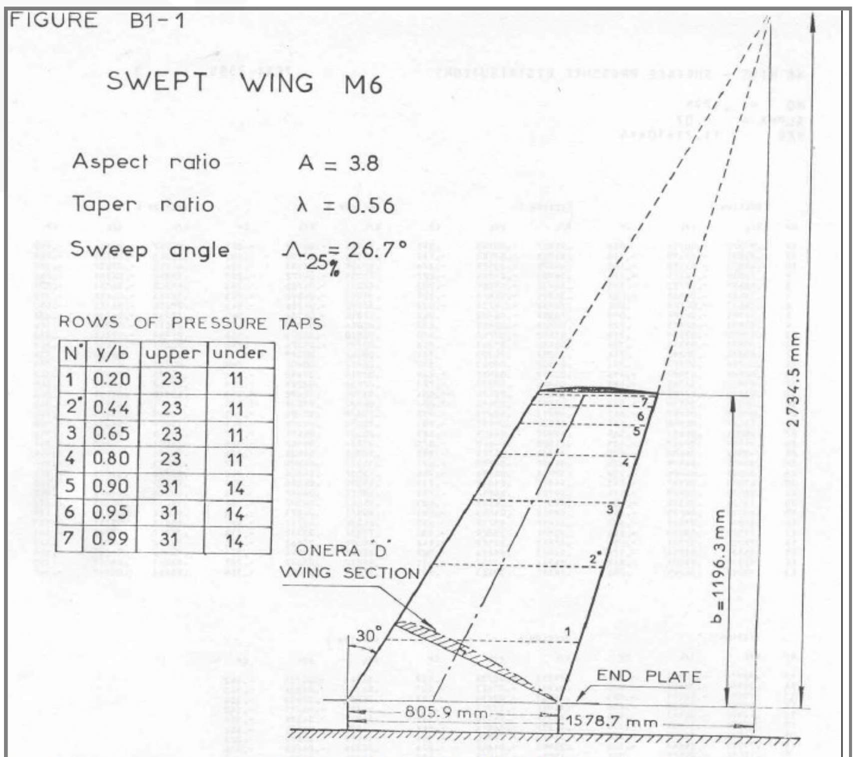
- Transonic Flow Over RAE2822 Airfoil
  - 36,000 cells Hybrid(quad + tri) grid
  - $M_\infty = 0.675$
  - komegaSST turbulence model

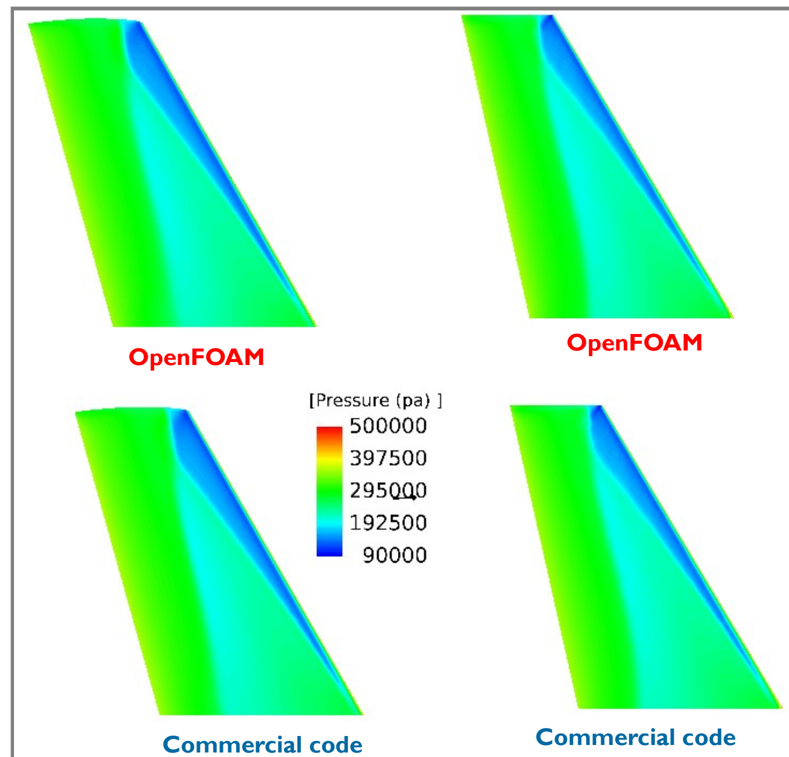




Results – Viscous Turbulent Flow(3D)

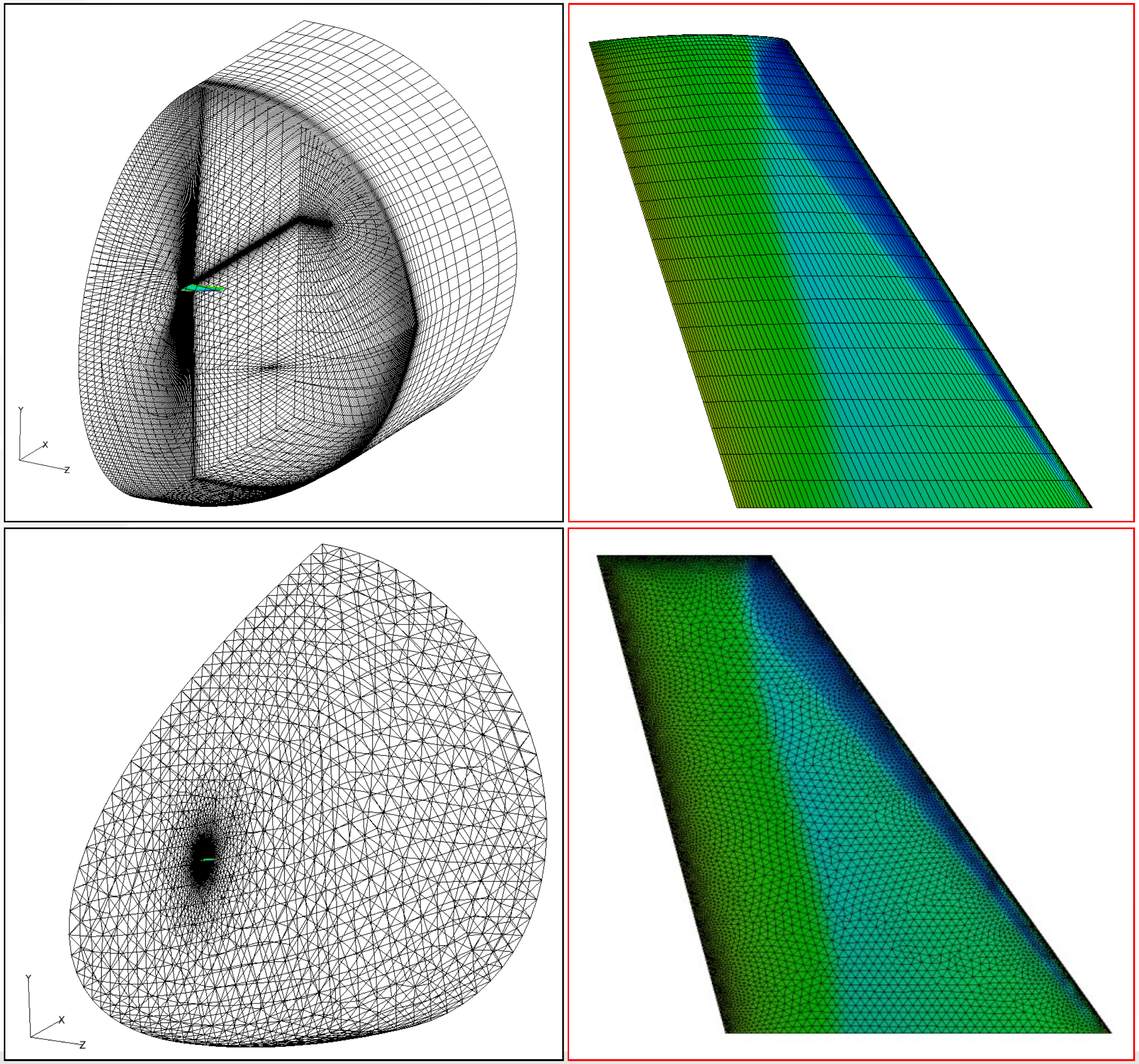
- Transonic Flow over ONERA M6 Wing
  - $M_\infty = 0.8395$, $p_\infty = 315980$pa, $T_\infty = 255.56$K
  - AoA = 3.06
  - Hex(NASA) and hybrid mesh





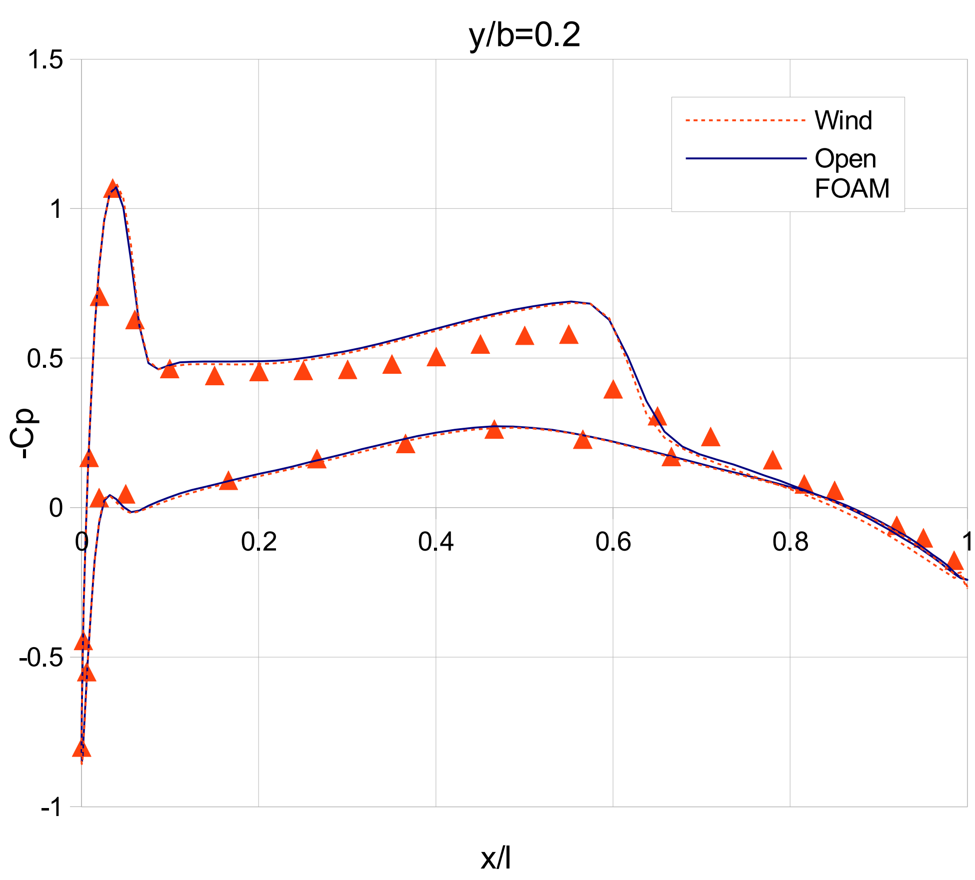
Results – Viscous Turbulent Flow(3D)

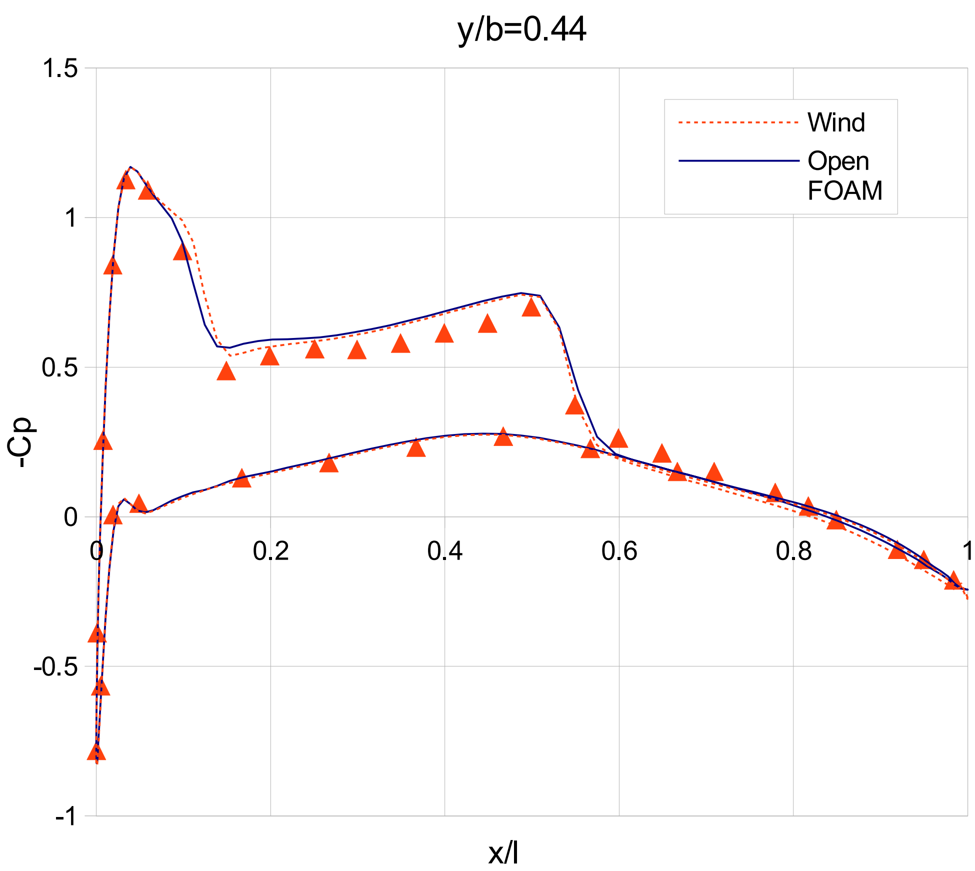
- Transonic Flow over ONERA M6 Wing



Results – Viscous Turbulent Flow(3D)

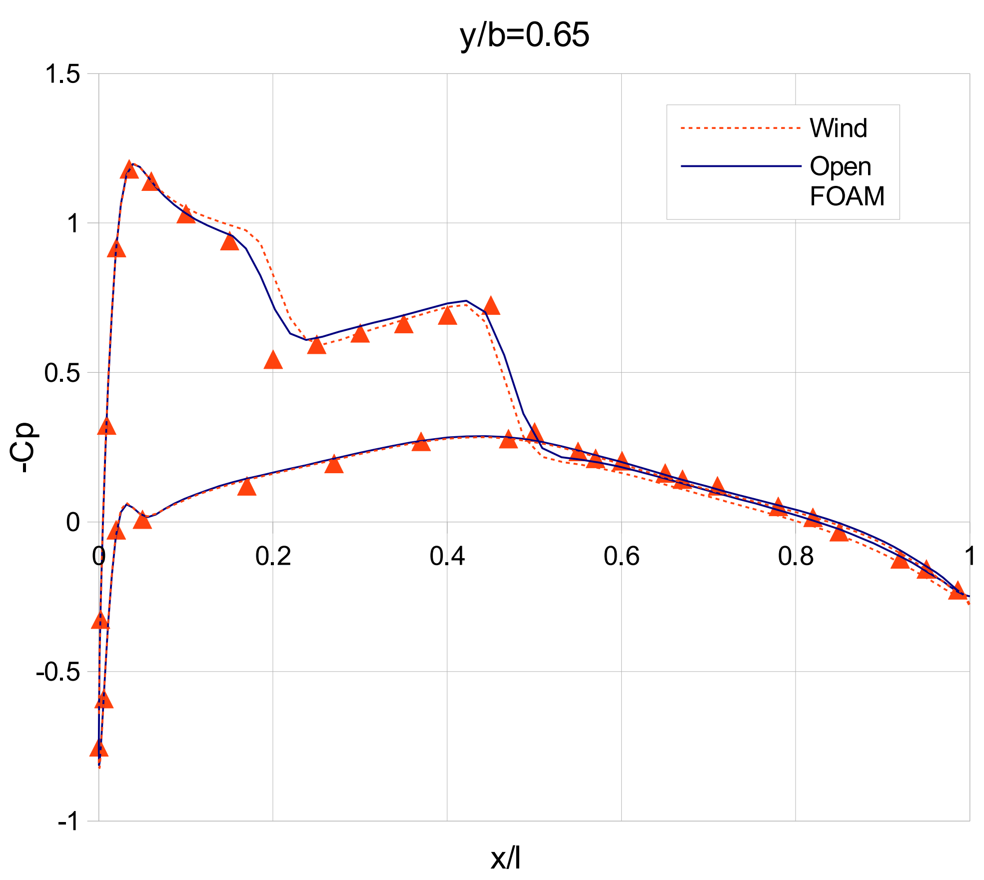
- Transonic Flow over ONERA M6 Wing

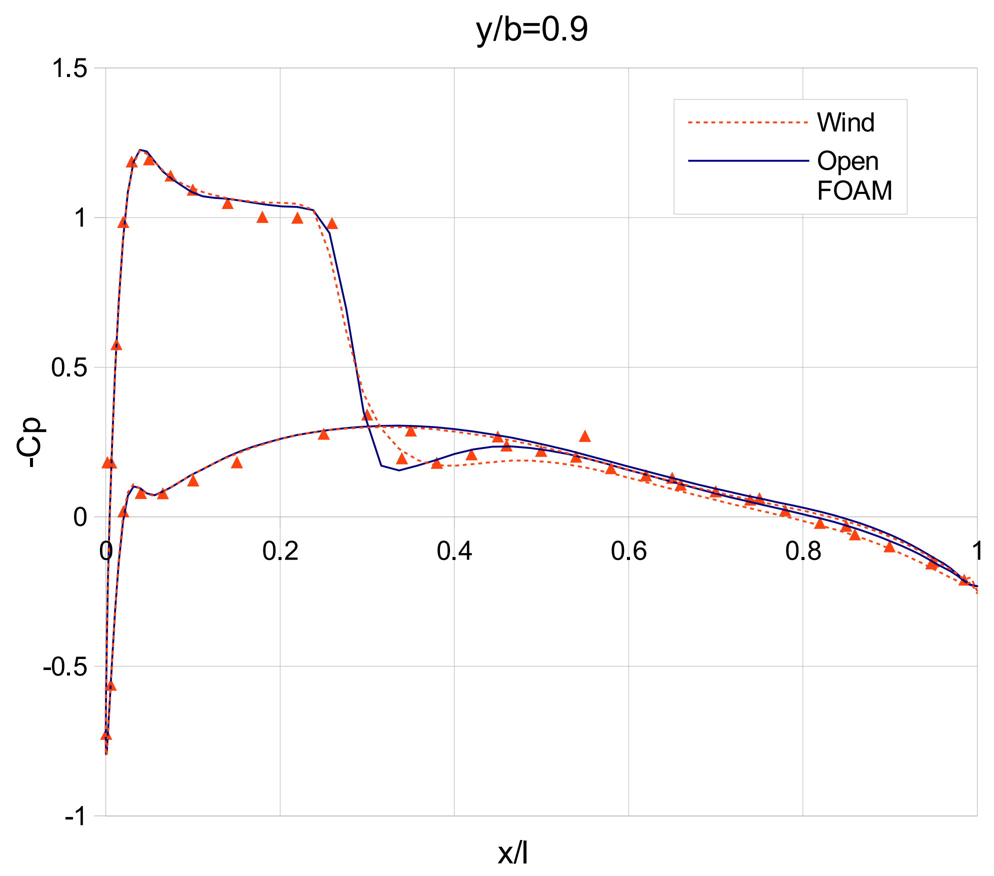




Results – Viscous Turbulent Flow(3D)

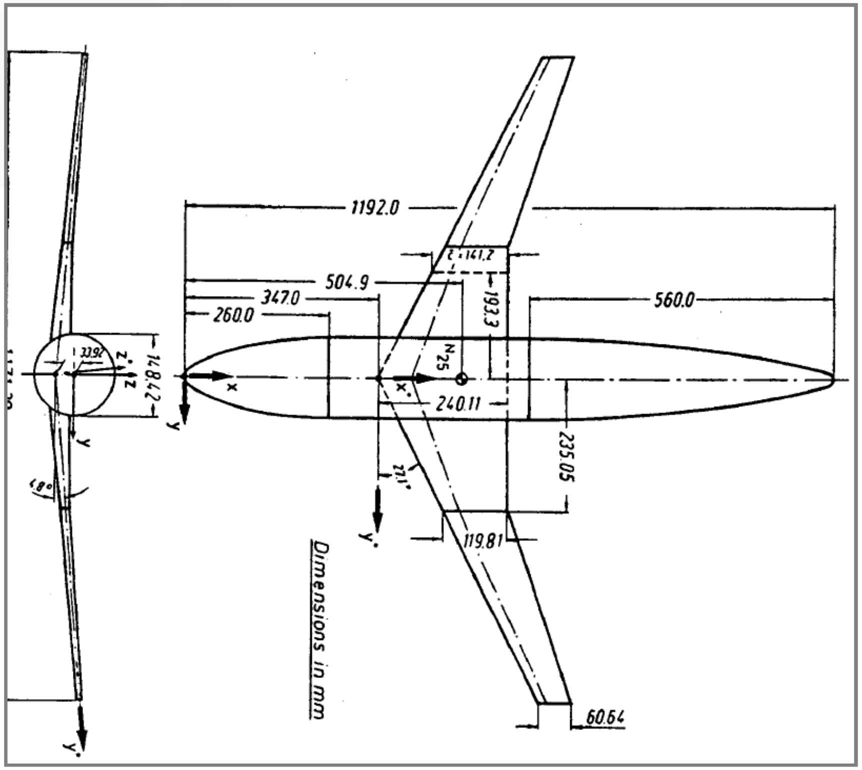
Transonic Flow over ONERA M6 Wing

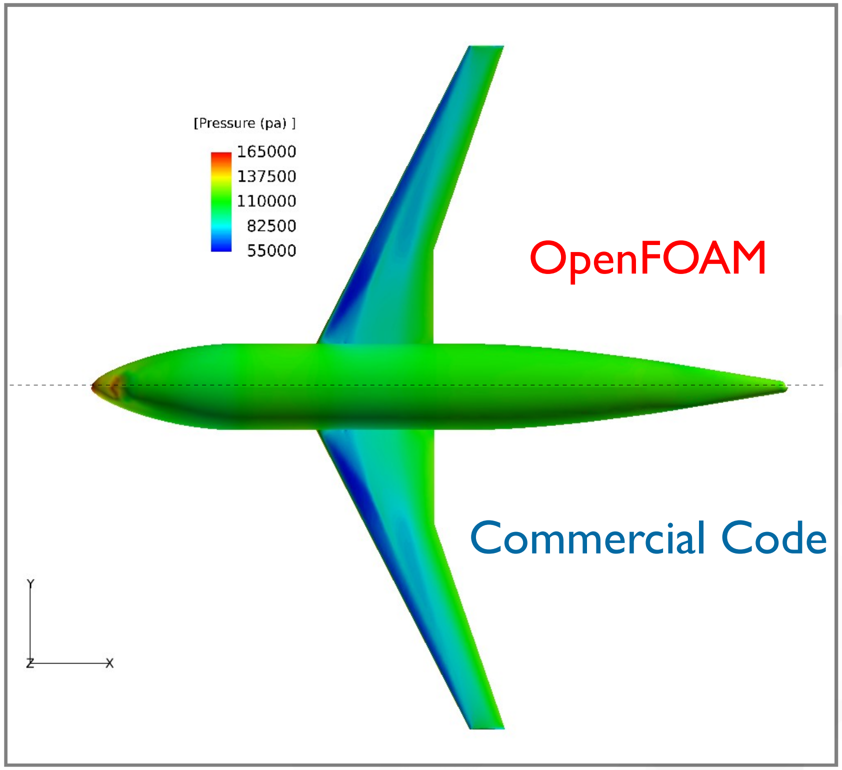




Results – Viscous Turbulent Flow(3D)

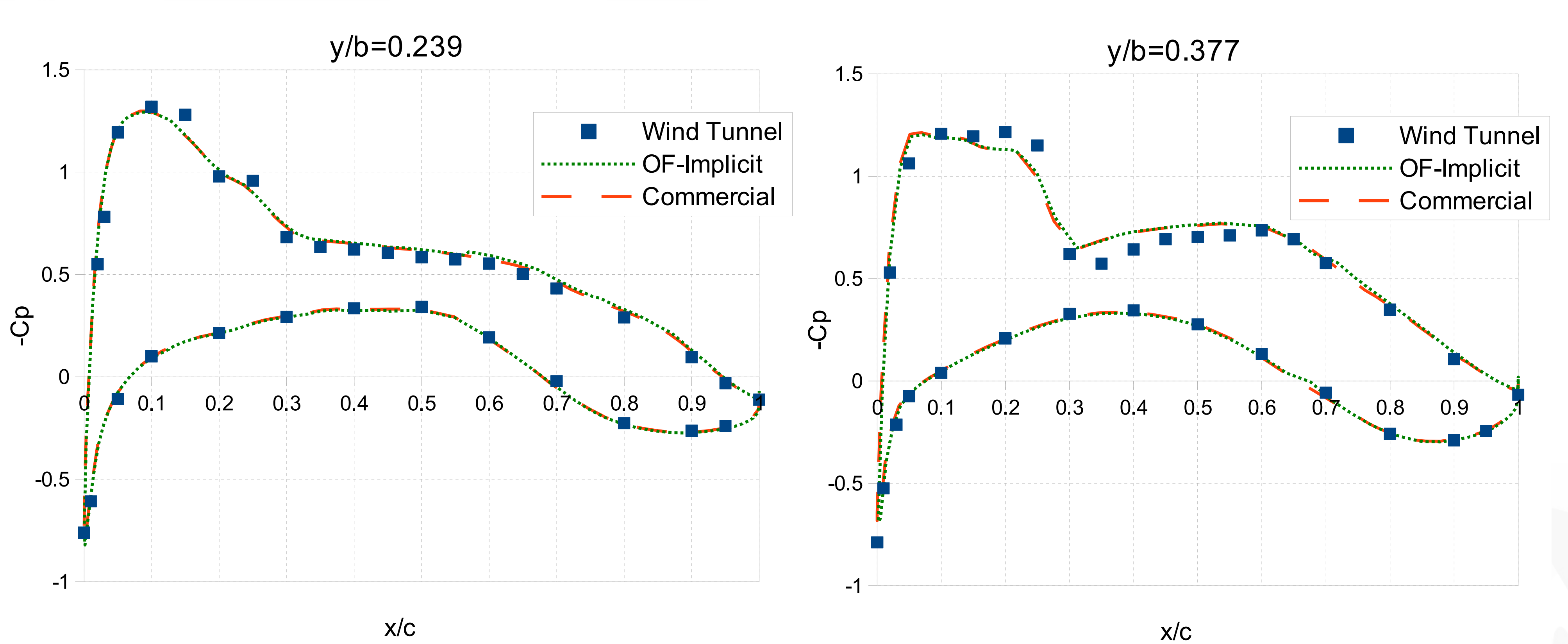
- Transonic Flow around DLR F4 Wing-Body
  - $M_\infty = 0.75$, $p_\infty = 116577\text{pa}$, $T_\infty = 300\text{K}$
  - AoA = 0.49
  - Hybrid mesh from AIAA DPW II (5,200,000 cells)





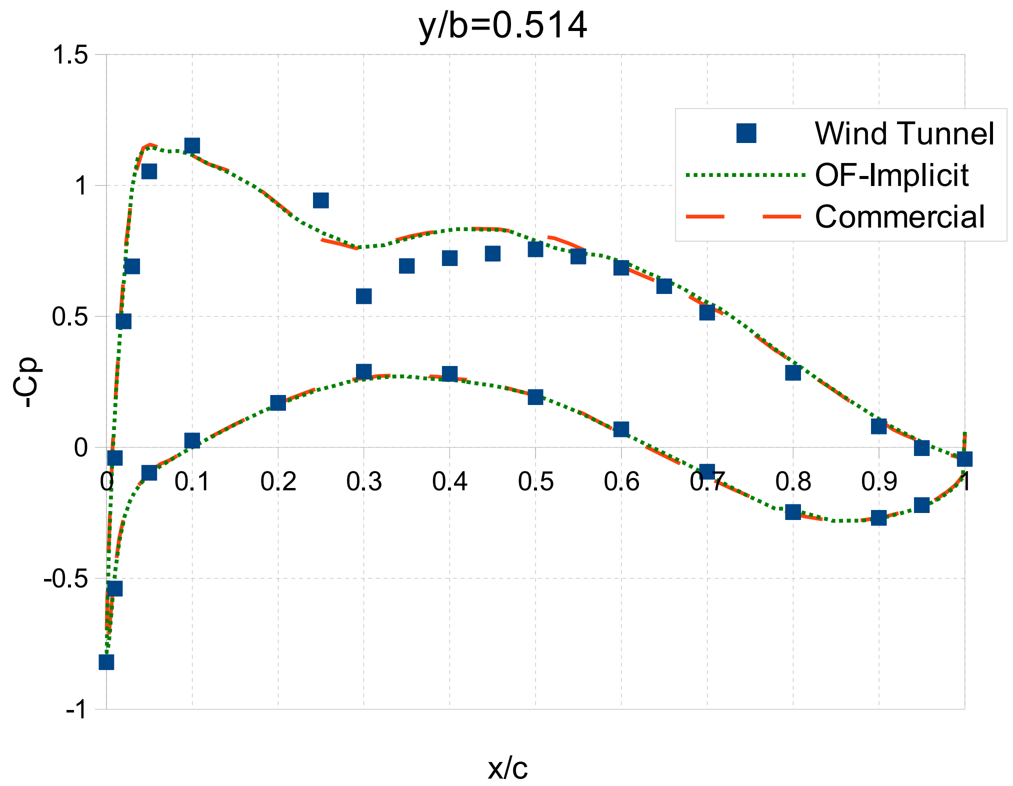
Results – Viscous Turbulent Flow(3D)

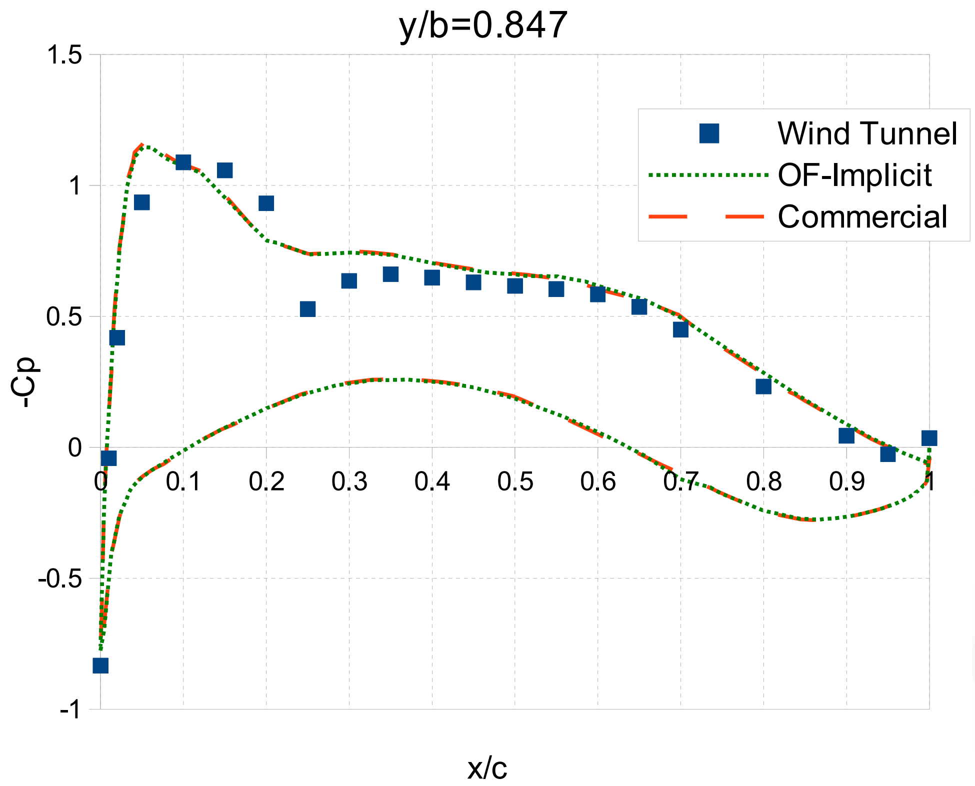
- Transonic Flow around DLR F4 Wing-Body



Results – Viscous Turbulent Flow(3D)

- Transonic Flow around DLR F4 Wing-Body





Outline

Concluding Remarks

- **Summary**
  - Implicit LU-SGS algorithm and new boundary condition have been implemented
  - The numerical results obtained indicate that implicit algorithm leads to an increase in performance over the explicit counter part
  - Solution was comparable to commercial code

- **Future works**
  - More efficient version of LU-SGS
  - Low mach number preconditioning
  - Multigrid

Thank you …

… for your attention!