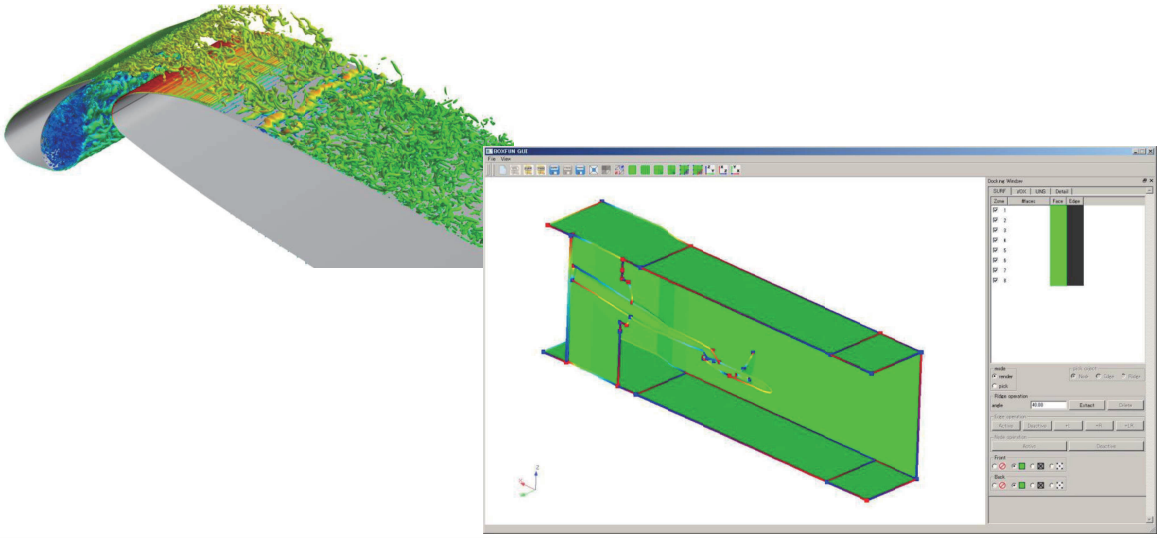
Fifth Aerodynamics Prediction Challenge (APC-V)

Flow Simulation around 30P30N with BOXFUN grid



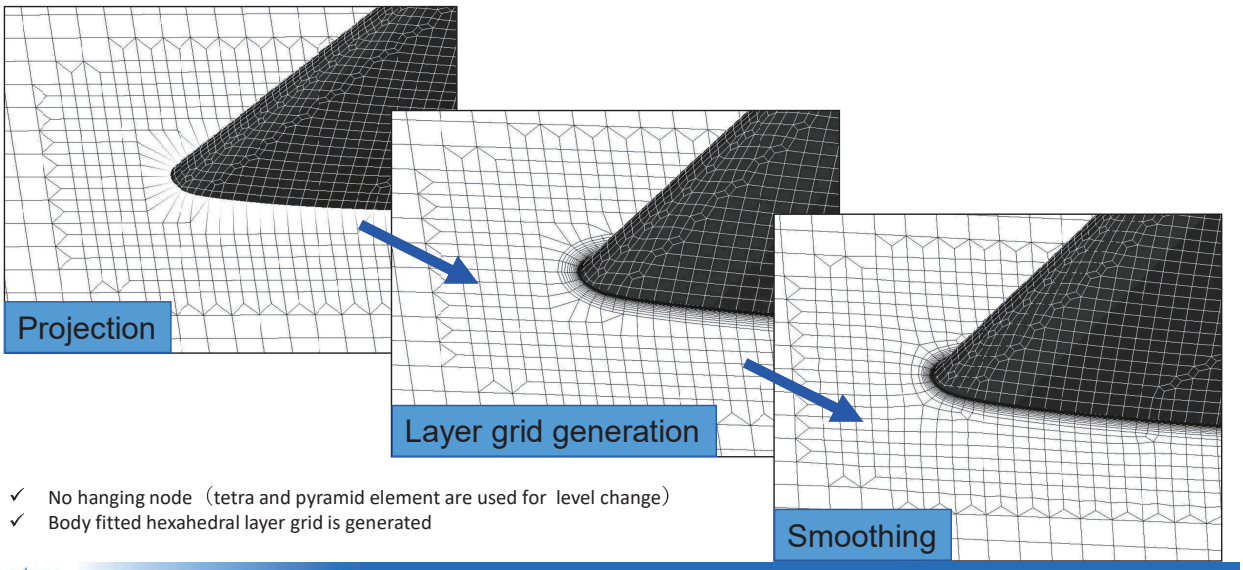
ISHIDA Takashi
Numerical Simulation Research Unit
Japan Aerospace Exploration Agency

Outline

- Background
  - HexaGrid to BOXFUN
  - Introduction of BOXFUN
- Some results of 30P30N simulation
- Conclusion

Development of HexaGrid

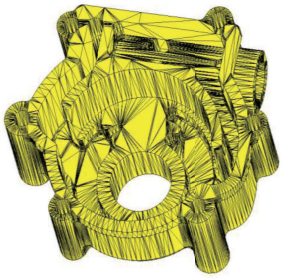
- Automatic grid generation tool based on hexahedral element
  - Up to 1 month by hand⇒1~2 hours by HexaGrid
  - Cartesian grid based unstructured grid⇒quick, easy adaptation for complex geom.

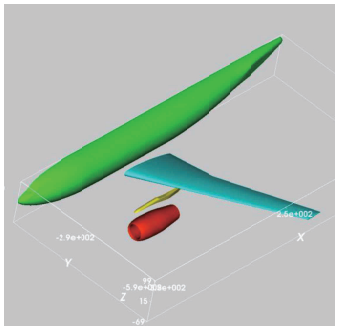


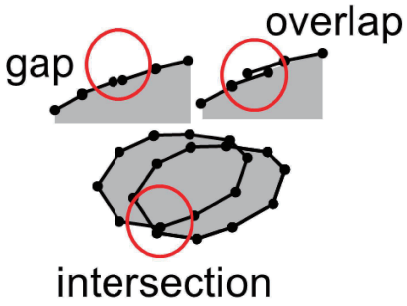
✓ No hanging node（tetra and pyramid element are used for level change）
✓ Body fitted hexahedral layer grid is generated

Characteristics of HexaGrid

- Handling dirty STL data ⇒ reduce CAD data fix
- Handling multi-component data ⇒ easy to replace parts



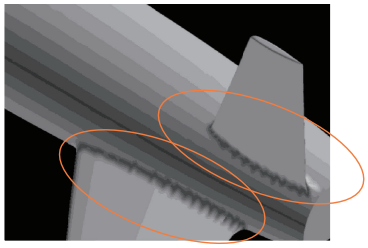




※HexaGrid can handle small gaps, intersections, and overlaps in STL data

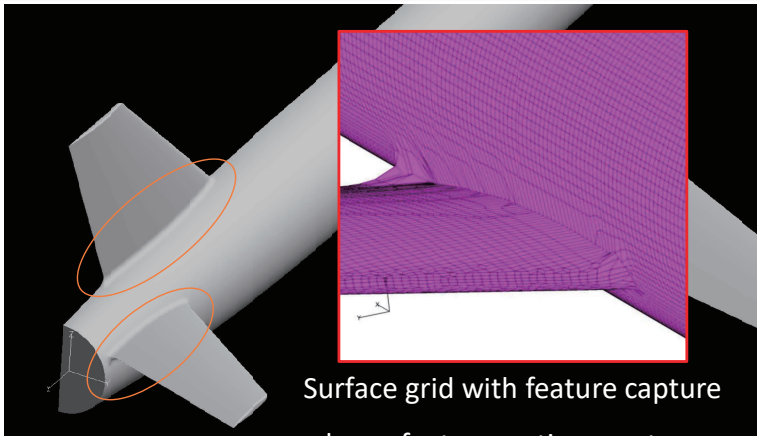
- Automatic grid generation with few parameters ⇒ user friendly
  - Size of domain (min/max coordinate of x, y, z)
  - Min/max cell size of wall boundary
  - Min grid spacing and expansion ratio for layer grid

Problems of HexaGrid



Surface grid without feature capture

hexa_feature_action: smooth

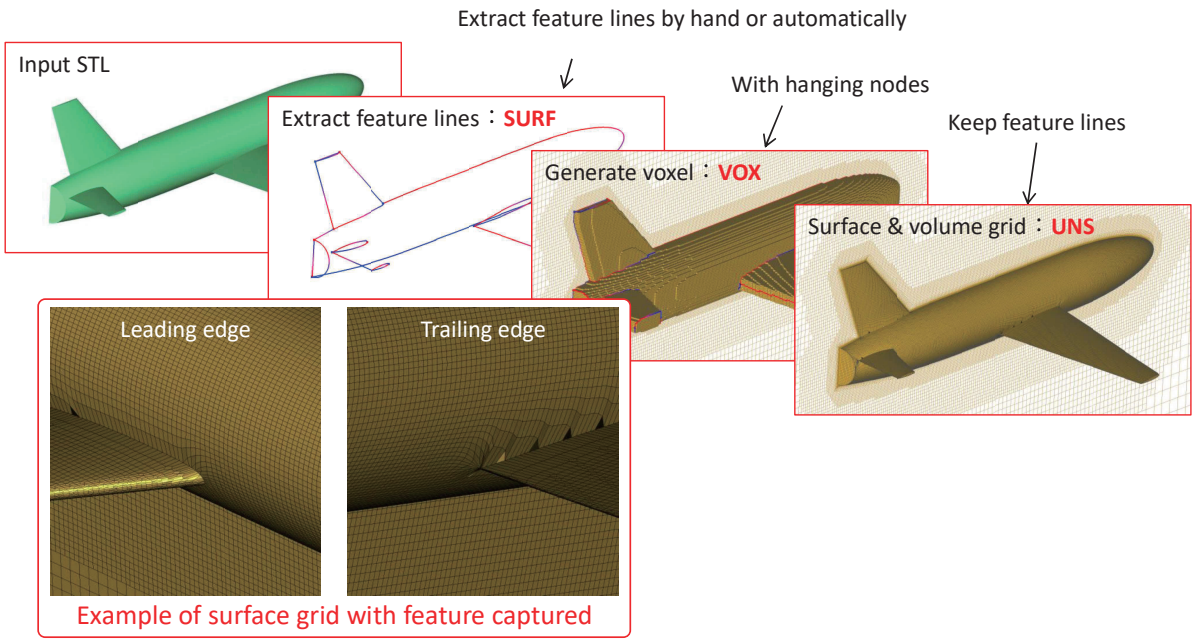


- HexaGrid often fails to capture geometric feature lines.
  ⇒not good at capturing concave shapes
- HexaGrid is a single process program and has some limitations.
  ⇒not suitable for large scale grid generation

Development of BOXFUN

- Development of new grid generator named BOXFUN has been started to overcome the problems of HexaGrid.
  - Feature lines ⇒implement manual function
  - Large-scale grid ⇒utilize the framework of Building-Cube Method
- BOXFUN
  Block-based vOXel for Fine UNstructured grid
- Development environment
  - language : C++, Visual Studio 2017
  - Visualization library : OpenGL, GLUT
  - GUI framework : Qt
- Operating environment
  - Windows
  - Linux (JSS2 remote desktop)
- Program module
  - BOXFUN SURF : extract feature lines
  - BOXFUN VOX : generate voxel
  - BOXFUN UNS : convert voxel to unstructured data

Procedure of grid generation



Details of voxel data structure

From Riken framework

```
template<size_t D>
OctreeNode{
    OctreeIPos<D>          m_pos;
    union{
        OctreeNode<D>      *m_child;
        OctreeLeaf<D>      *m_leaf;
    };
};

template<size_t D>
OctreeInfo{
    unsigned short         depth;
    unsigned short         pdg[D];
};
```

m_pos : 2+2D=2(D+1)[byte]
m_child : 4[byte]
Total : 2(D+4)[byte]

BOXFUN

```
template<size_t D>
OmnitreeNode{
    OmnitreeIPos<D>        m_pos;
    union{
        OmnitreeNode<D>    *m_child;
        OmnitreeLeaf<D>    *m_leaf;
    };
};

template<size_t D>
OmnitreeInfo{
    unsigned short         flags;
    unsigned short         depth;
    unsigned int           pdg[D];
};
```

m_pos : 2+2+4D=4(D+1)[byte]
m_child : 4[byte]
Total : 4(D+2)[byte]

Bit layout

OmnitreeInfo::depth															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
flag	Z depth					Y depth					X depth				

OmnitreeInfo::flags															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID								flag					dimension		
domain number for MPI								leaf	wall	fluid	agg.	div.	Z	Y	X

unsigned short type : 2[byte] =16[bit]

unsigned int type : 4[byte] =32[bit]

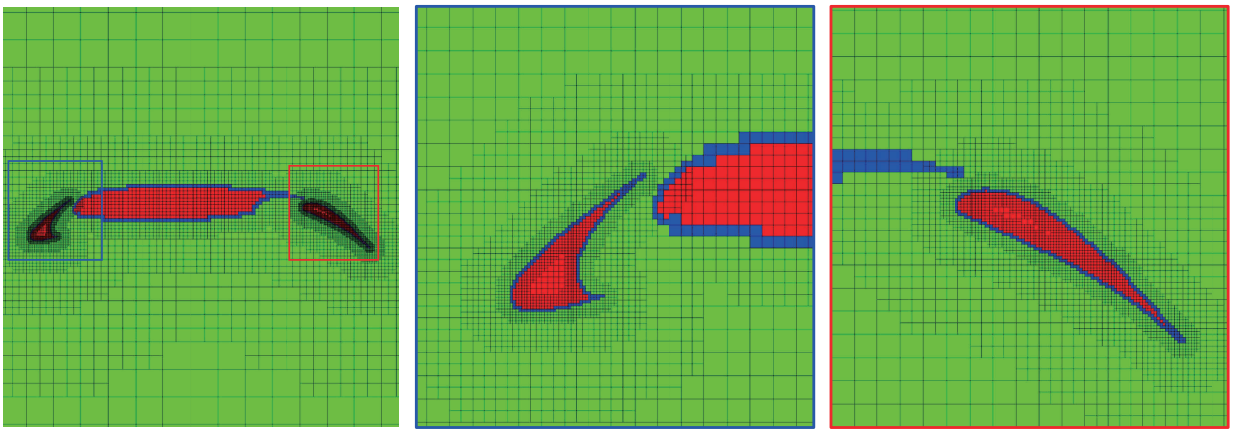
Depth in each direction : 5[bit] =0~31

Resolution : 2^{31} $=2147483648$

($4.7 \times 10^{-8} L_\infty$ with outer boundary size= $100 L_\infty$)

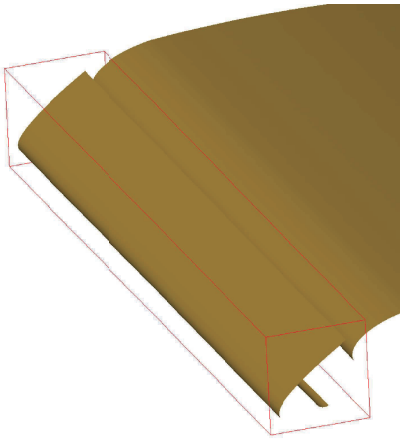
Parameter setting for each component

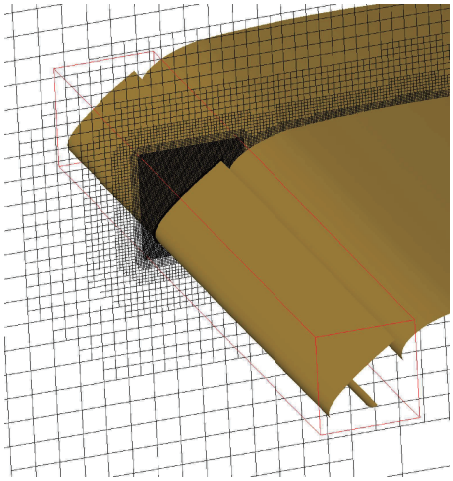
- User set min/max depth of tree
- Example of grid generation around 30P30N
  - Slat/flap : depth 15
  - main : depth 13



Refinement Box

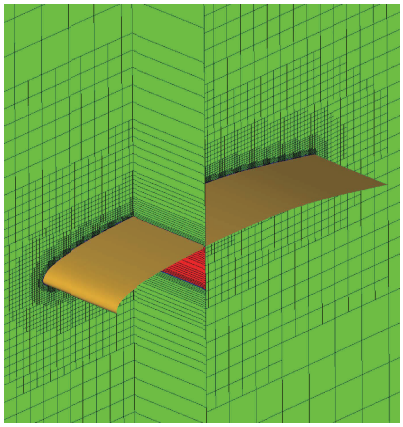
- Similar function with HexaGrid
- Set box information（coordinate, size, depth）



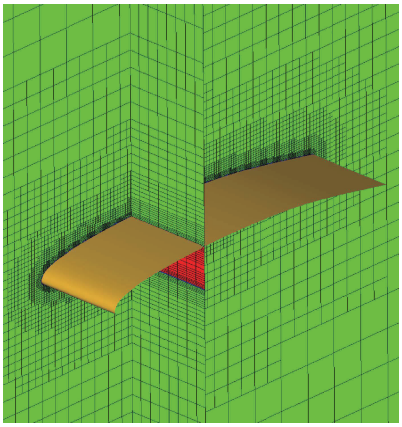


2.5D grid generation

- Pseudo 3D grid generation function (not available in HexaGrid)
  - position matching in spanwise direction for periodic boundary condition
  - # of division in spanwise direction has 2^n restriction
  - All voxel is divided in spanwise direction at the same time.



Spanwise:1cell

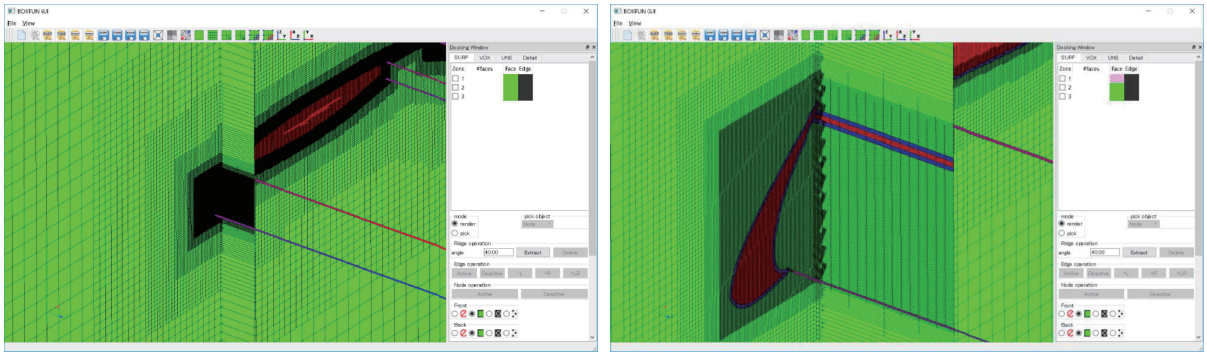


Spanwise:8cells in everywhere

2.5D grid generation

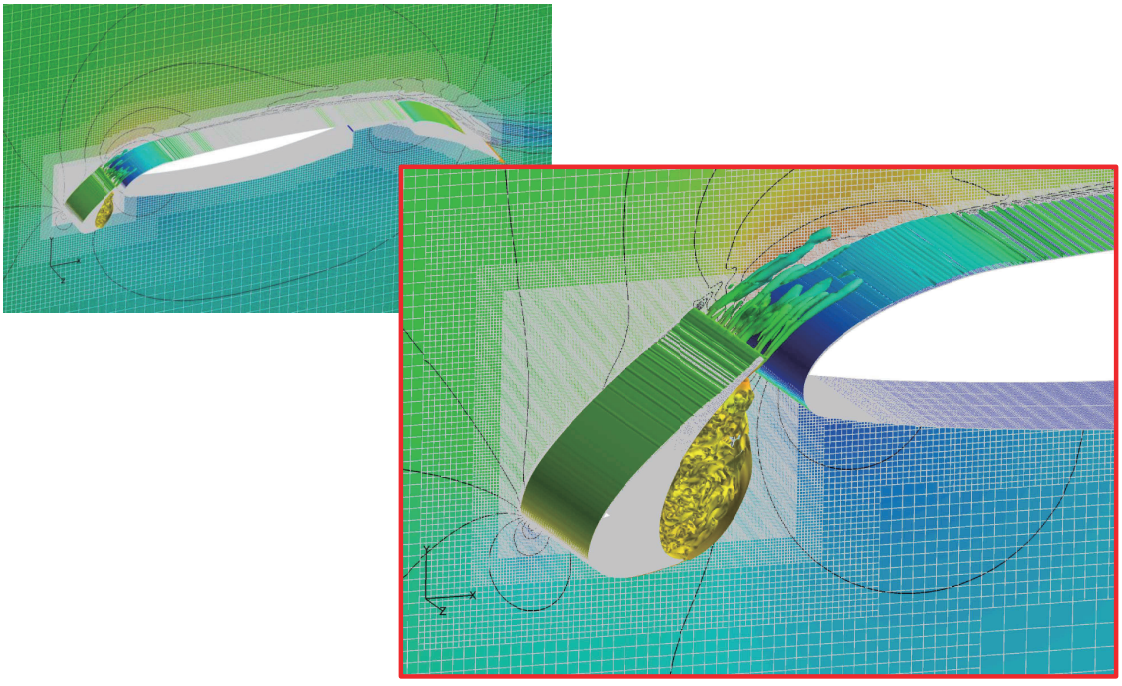
- Code modification has conducted.
- New module could reduce total grid points.

	2D grid	2.5D grid (128 cells, org)	2.5D grid (128 cells, mod1)	2.5D grid (128 cells, mod2)
# of voxel	184756	23,648,768	16,160,471	5,521,317
Reduction rate	-	-	68%	23%



Visualization of mod2 grid

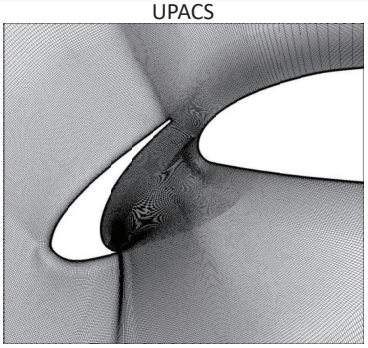
Still simulating 3D…



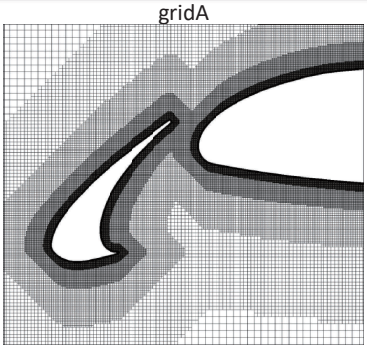
2D RANS:Grid

Slat

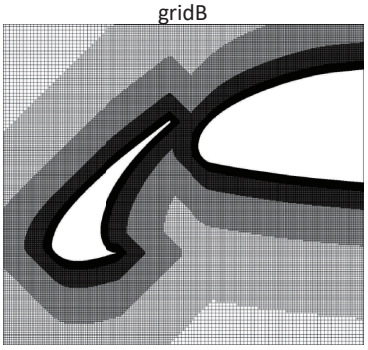
UPACS



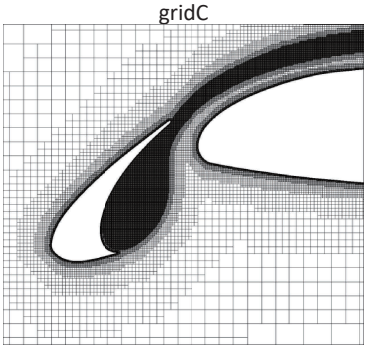
gridA



gridB



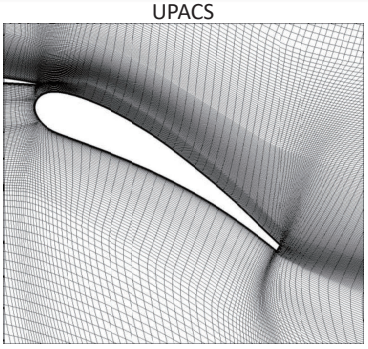
gridC



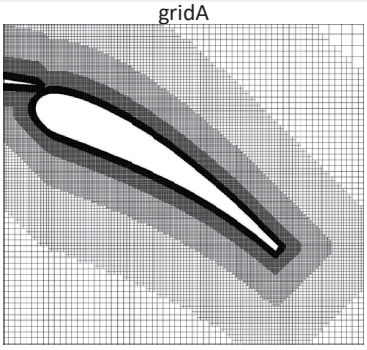
2D RANS:Grid

Flap

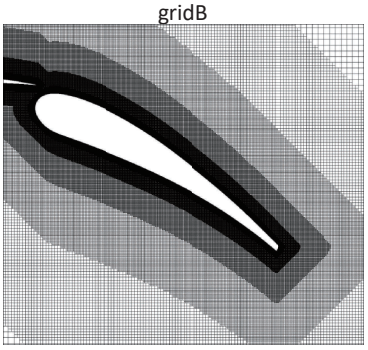
UPACS



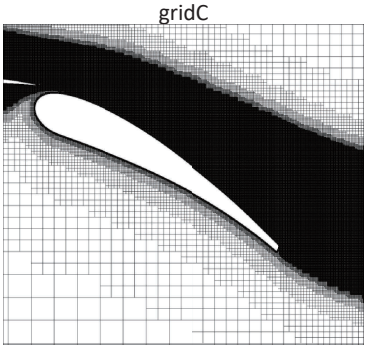
gridA



gridB



gridC

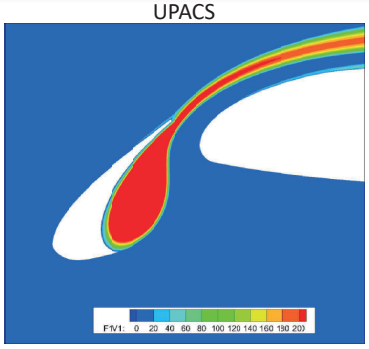


Takashi ISHIDA, JAXA/NSRU

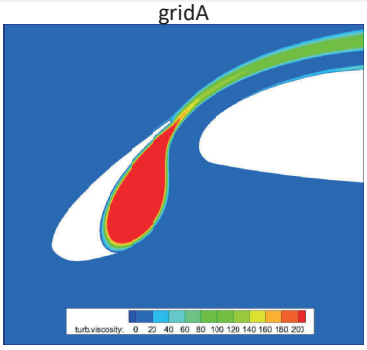
2D RANS: ν

Slat

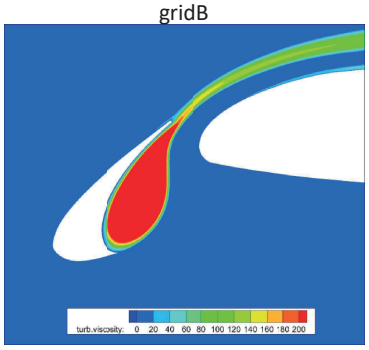
UPACS



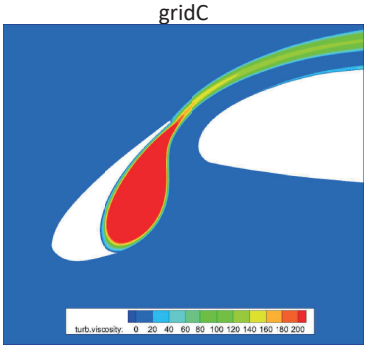
gridA



gridB



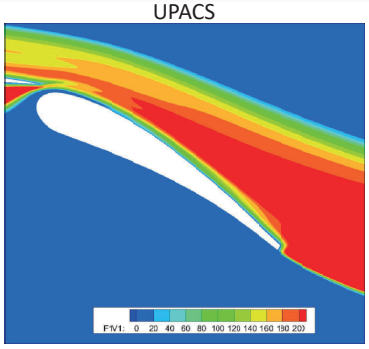
gridC



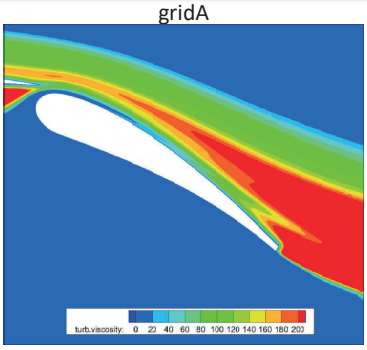
2D RANS: ν

Flap

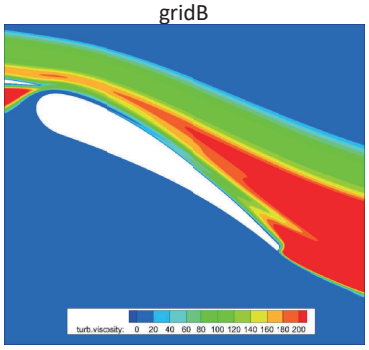
UPACS



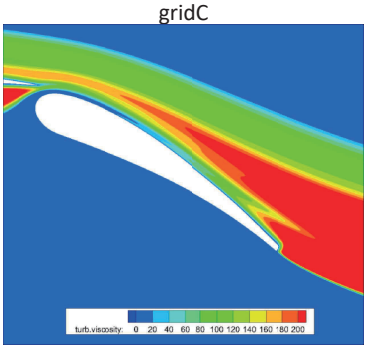
gridA



gridB

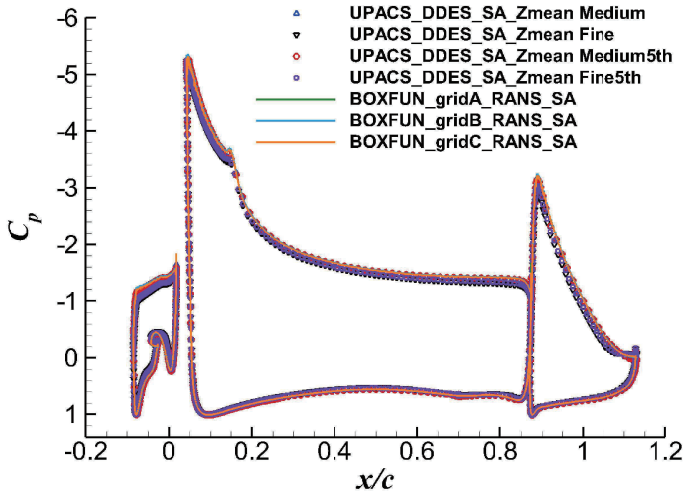


gridC



Takashi ISHIDA, JAXA/NSRU

Comparison of Cp



	α°	CD	CL	CM
gridA	5.5	0.04127	2.89378	0.61982
gridB	5.5	0.04084	2.89519	0.61985
gridC	5.5	0.04337	2.87959	0.61782

Summary

- BOXFUN was modified and used for 30P30N simulation.
  - We will update unsteady data ASAP.
- We would like to make meshing guideline for high lift configuration through APC-IV and V.
  - What is “good” grid for simulating high-lift configuration?
  - How fine mesh do we need around high lift configuration?
  - What turbulence model should we use?
  - Where is convergent result?
  - 2D preliminary result suggested that wake refinement only is not enough…
  - Initial value problem, hysteresis, etc