Development of a High-Order Discontinuous Galerkin Fluid Solver Within SU2

Edwin van der Weide
Department of Mechanical Engineering
University of Twente

Thomas D. Economon, Juan J. Alonso, Jae hwan Choi, Carlos da Silva
Department of Aeronautics and Astronautics
University of Stanford

Dheevatsa Mudigere, Alexander Heinecke, Gaurav Bansal
Intel Corporation

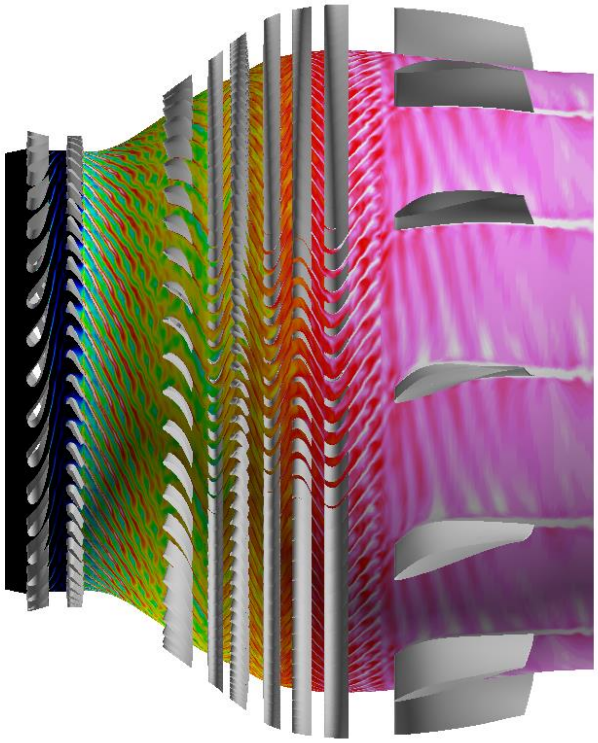
Ramesh Balakrishnan
Argonne National Laboratory

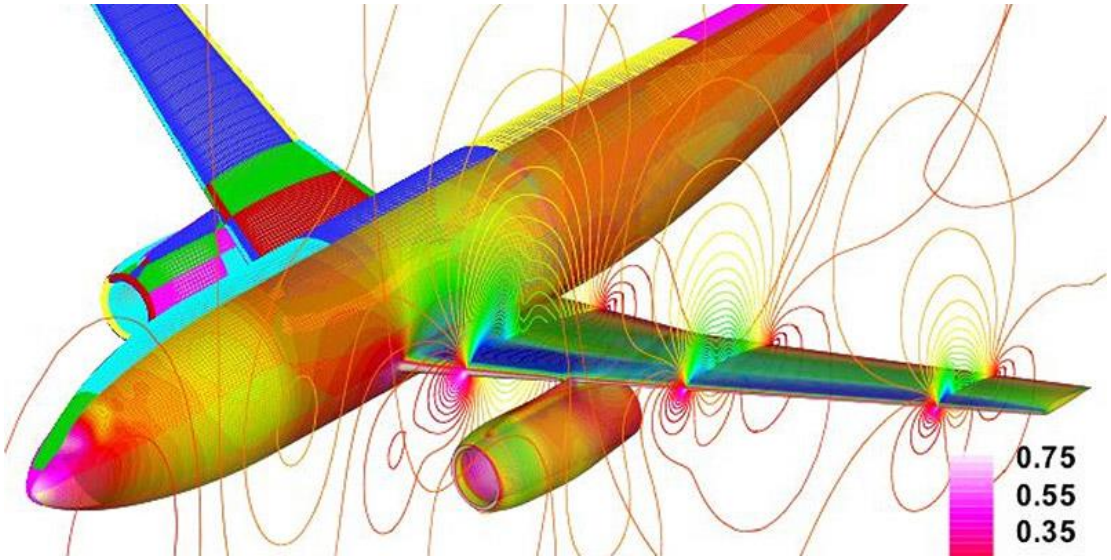
Outline

- Motivation for high order schemes
- Pros and cons of different discretization techniques
- DG-FEM, the basic principles
- Implementation in SU2
- Performance optimization
- High order grid generation
- Preliminary results
- Next things to be done

Why do we need high order schemes?

2nd order schemes have been extremely successful, certainly for RANS applications



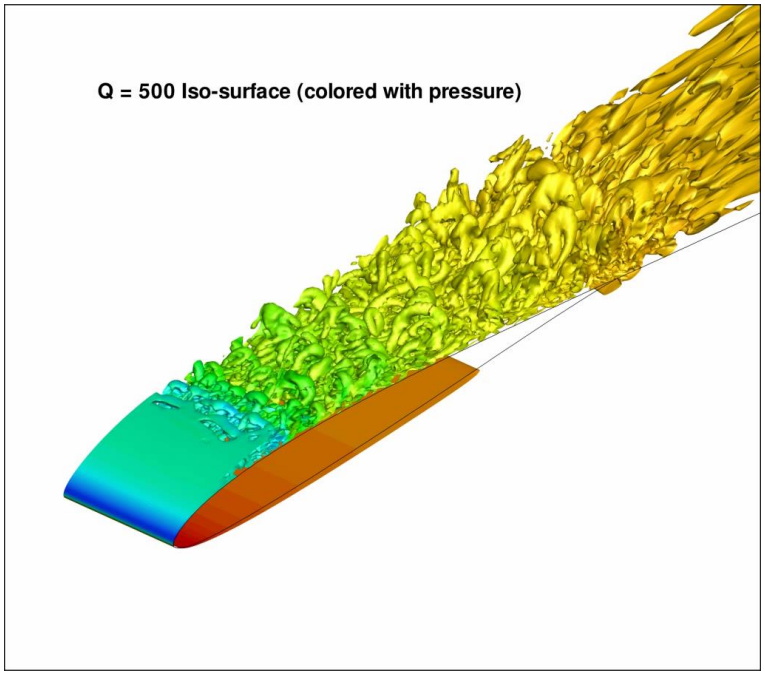


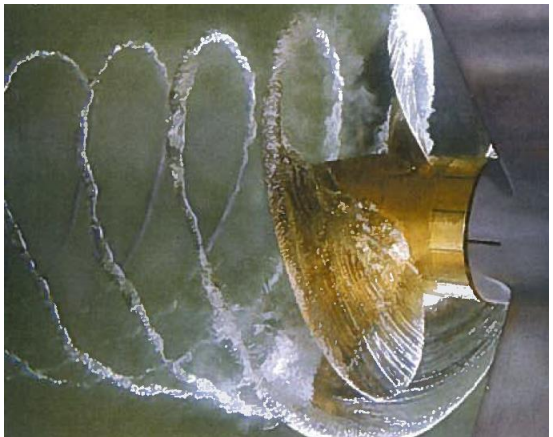
But

For some applications 2nd order accuracy may not be enough

Examples

- Wake and vortex flows
- Noise prediction
- LES/DNS



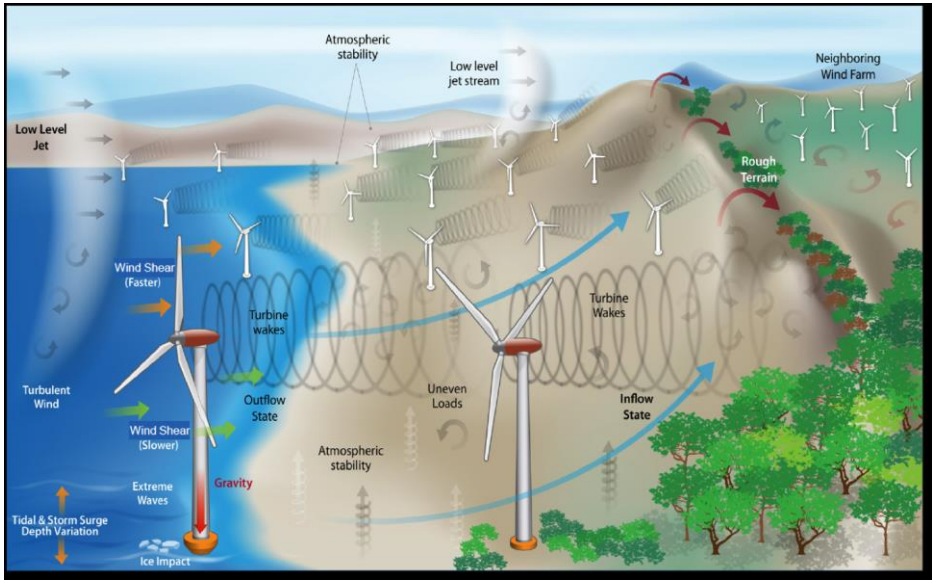


ALCF Theta Early Science Program

Collaboration Stanford – Argonne

Scale Resolving Simulations of Wind Turbines with SU2

Horns Rev wind farm (Denmark)



Wakes must be resolved

Ideally suited for high order schemes

Spatial discretization methods for hyperbolic systems (most commonly used)

- Finite Difference Method (FDM)
- Finite Volume Method (FVM)
- (Discontinuous Galerkin) Finite Element Method (DG-FEM)

Pros and cons of the methods

	Complex geometries	High-order accuracy and *hp*-adaptivity	Explicit semi-discrete form	Conservation laws	Elliptic problems
FDM	×	✓	✓	✓	✓
FVM	✓	×	✓	✓	(✓)
FEM	✓	✓	×	(✓)	✓
DG-FEM	✓	✓	✓	✓	(✓)

× : Not suited

✓ : Suited

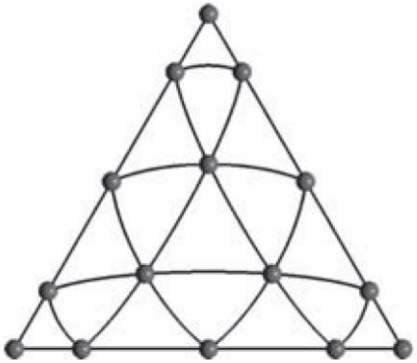
(✓) : Suited, possibly with modifications, but not the most natural (or efficient) choice

Taken from Hesthaven and Warburton, 2008

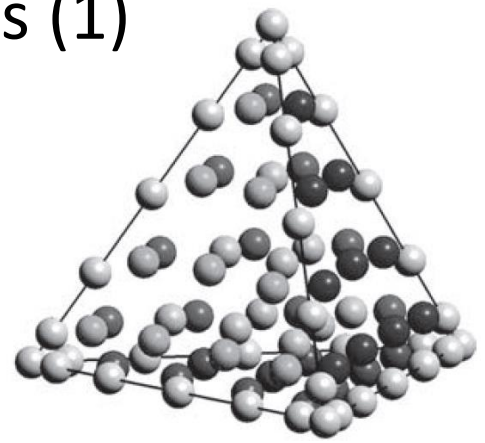
2nd order: FVM is the best choice

High order: DG-FEM is the best choice

Nodal DG-FEM: the basic principles (1)



$$\text{Element } k: \; U(x_i) = \sum_{j=1}^{N_p} U_j^k \varphi_j^k(x_i)$$



Hyperbolic system of PDE's
Weak formulation

$$\frac{\partial U}{\partial t} + \frac{\partial F_i}{\partial x_i} = 0 \quad \Rightarrow$$

$$\iint_{V_k} \frac{\partial U}{\partial t} \varphi_m^k \, dV - \iint_{V_k} F_i \frac{\partial \varphi_m^k}{\partial x_i} \, dV + \oint_{\partial V_k} F_i n_i \varphi_m^k \, d\Omega_k = 0, \quad m = 1, \ldots, N_p$$

Nodal DG-FEM: the basic principles (2)

Contribution from the contour integral

Solution at the interfaces is multiply defined and discontinuous

$$\oint_{\partial V_k} F_i n_i \varphi_m^k \, d\Omega_k \Rightarrow \oint_{\partial V_k} F_i\left(U_L, U_R\right) n_i \varphi_m^k \, d\Omega_k$$

Riemann problem: Any approximate Riemann solver can be used => stabilizes the discretization

1st order DG-FEM equals 1st order FVM!!!

Nodal DG-FEM: the basic principles (3)

Nonlinear equations and/or curved elements

Integrals must be computed with high order quadrature rules to avoid aliasing.
Expensive!!!

Diffusion problems, 2^{nd} derivatives

Discontinuous basis functions not suited => must be repaired
Even more expensive!!!

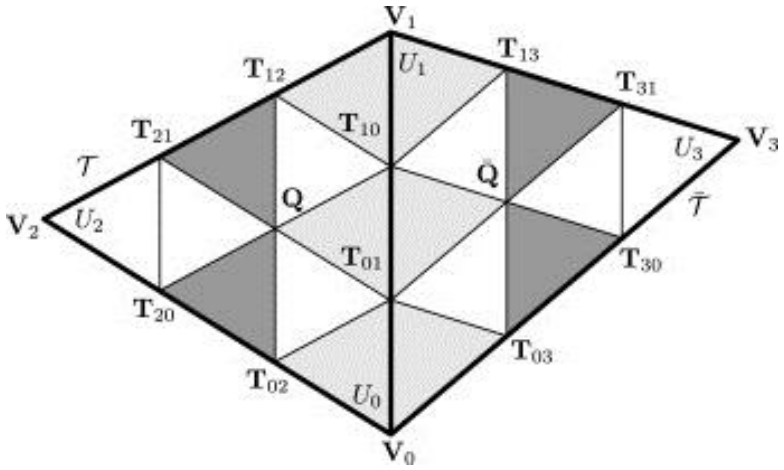
However: most operations are local to an element.
Extremely well-suited for modern computer hardware

Implementation in SU2 (1)
together with Tom

- Framework of SU2 is very flexible => high level data structures can be used for any solver, so also DG-FEM
- Input parameter structure can be reused entirely
- FVM output functionality could be reused (after some modifications)
- Still a lot of work was required for other low level functions
  - Partitioning of the grid (element wise)
  - Preprocessing is completely different from FVM
  - Standard elements and standard orientation of elements are introduced (see next slide)
  - Spatial discretization is completely new

Implementation in SU2 (2)

Tricky part: Relative orientation of elements
Needed for the discretization of the viscous terms
2D hybrid grid: 37 different cases
3D hybrid grid: 248 different cases



Solution: Renumber connectivity of adjacent elements for each face.
Similar to the CGNS transformation matrix between blocks.
=> One general implementation for the contour integral

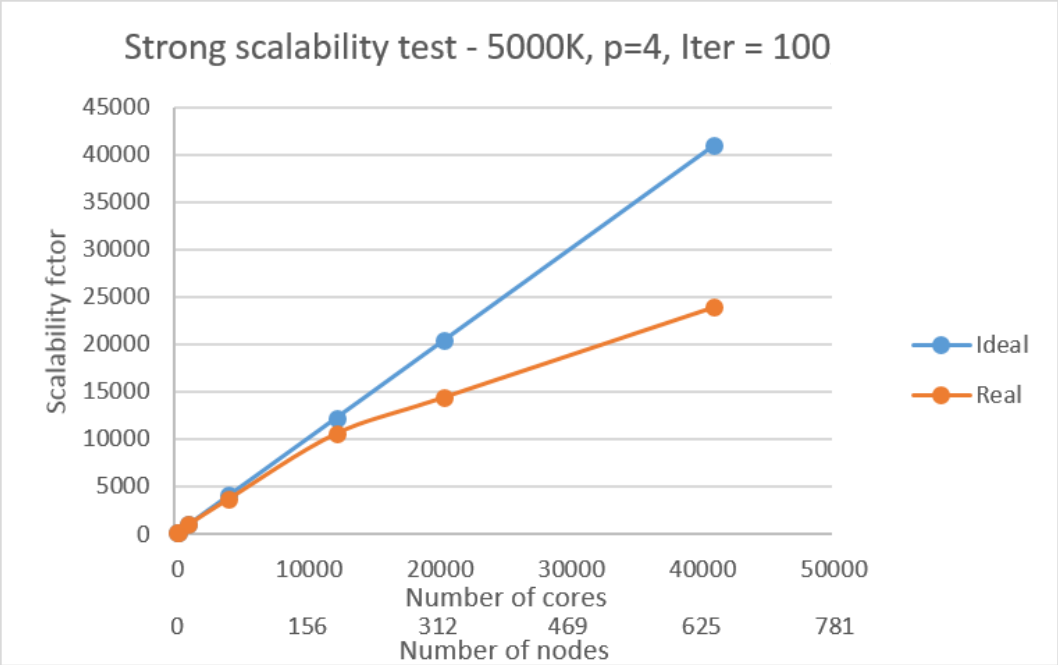
Performance optimization (1)

Collaboration with Intel

- Target architectures: Intel Knights Landing (MIC architecture)
  Intel Xeon
- Hybrid MPI/OpenMP parallelization
- MPI
  - Domain decomposition (one halo layer of elements)
  - Overlap computation and communication
  - Use of persistent communication
- OpenMP
  - Aim: Parallelization on the for-loop level (sufficient for tests on Xeon)
  - Current data structures are designed for this approach
- Optimized BLAS/LAPACK/LIBXSMM functions must be used to get good performance for matrix multiplications
  - Large contiguous chunks of memory for data storage (not the case for the FVM solver)

Performance optimization (2)

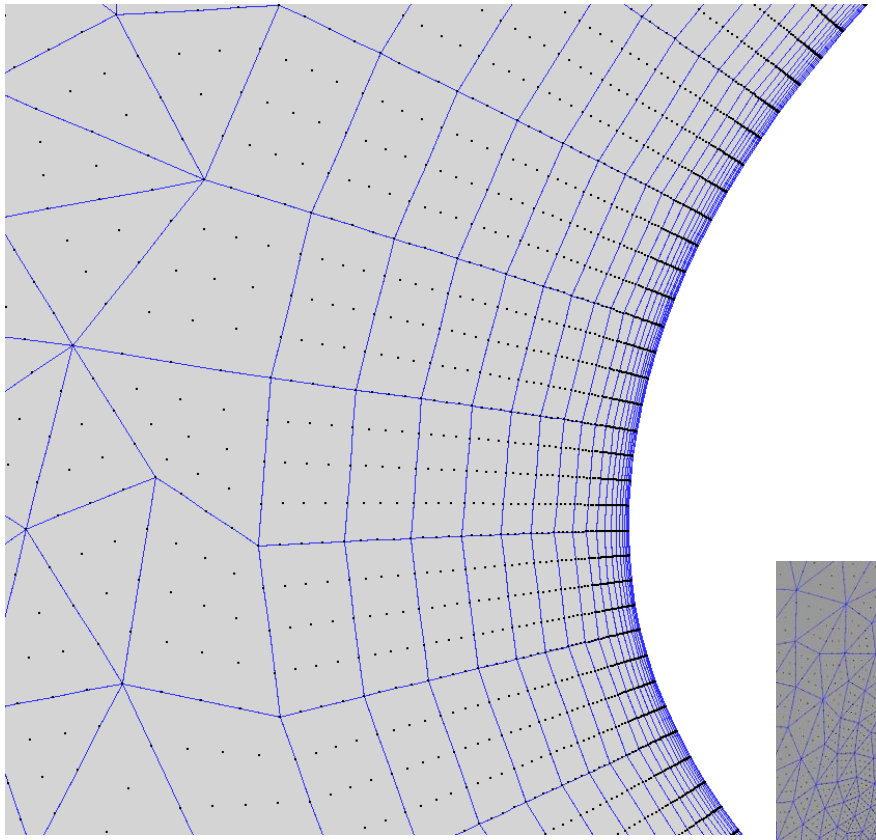
Motivation for hybrid parallelization approach
Flat MPI does not seem to work too well on KNL
Preliminary results on Theta (Argonne)

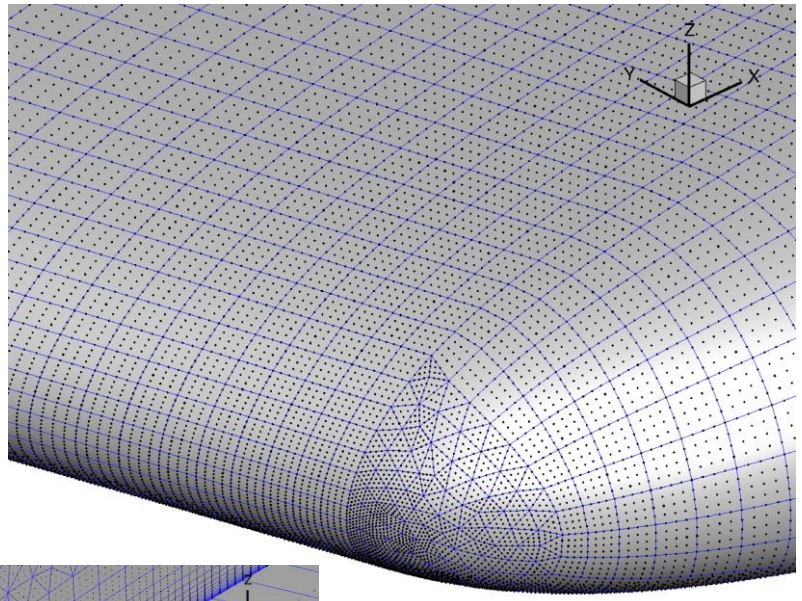


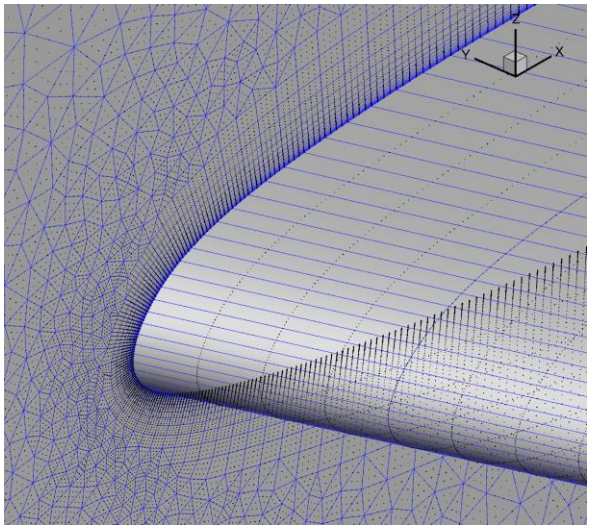
Elem/Core	122	244	407	1.22K	4.88K	19.5K	39.1K
DOF/Core	4.27K	8.55K	14.2K	42.7K	0.17M	0.68M	1.37M
Efficiency(%)	58.4	70.6	86.6	91.0	99.2	100	N/A

High order grid generation (1)

High order elements must be curved near solid wall boundaries







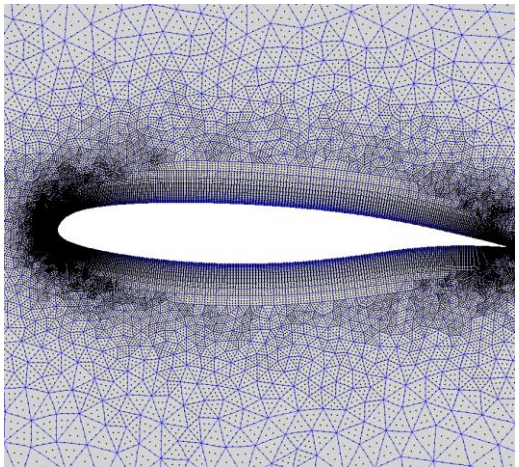
High order grid generation (2)

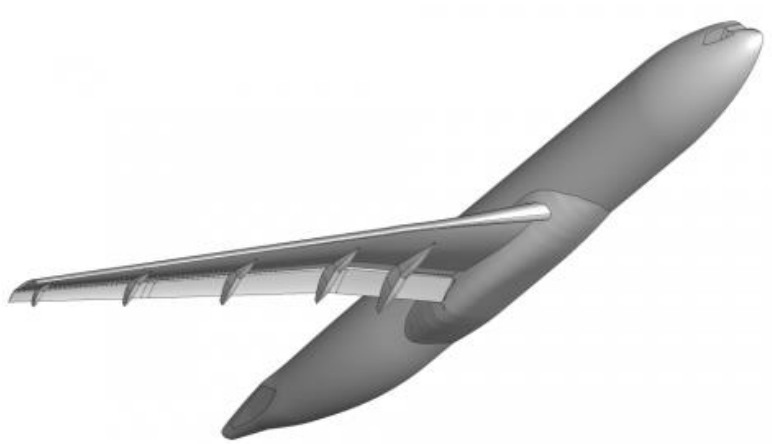
- Two possibilities
  - Truly high order grid generation
  - Post processing of linear grids => seems to be the preferred choice
- Not a lot of activity from commercial vendors
  - Centaursoft: Quadratic elements via post processing
  - Pointwise in collaboration with several universities (including us)
- Open source software
  - Gmsh

A lot more effort should be put into high order grid generation in order not to become a show stopper !!

High order grid generation within SU2
Collaboration with Pointwise

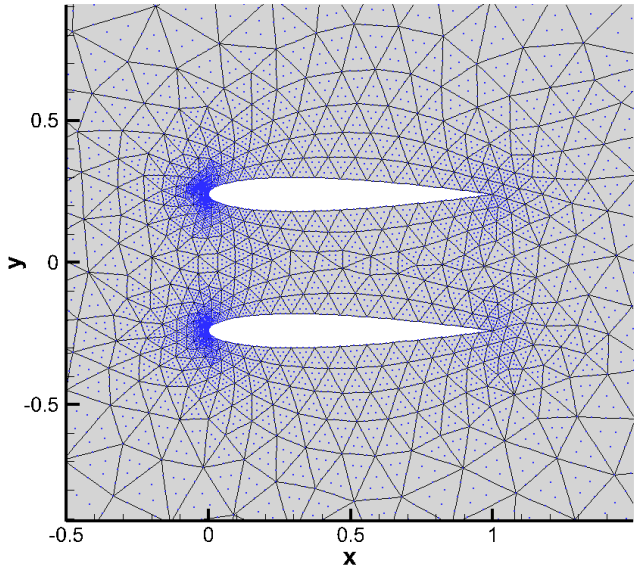
- Post processing of linear grids (quality linear grid is crucial)
- Geometry kernel of Pointwise (GE Lite) is used for projection onto the CAD geometry
- Correction due to surface projection is propagated into the domain
  - Heuristic algorithm (only for structured normal direction): very fast
  - SU2 mesh deformation algorithm: works for all element types
- Works very well for not too complicated geometries
- To be tested for really complicated cases (problems expected)

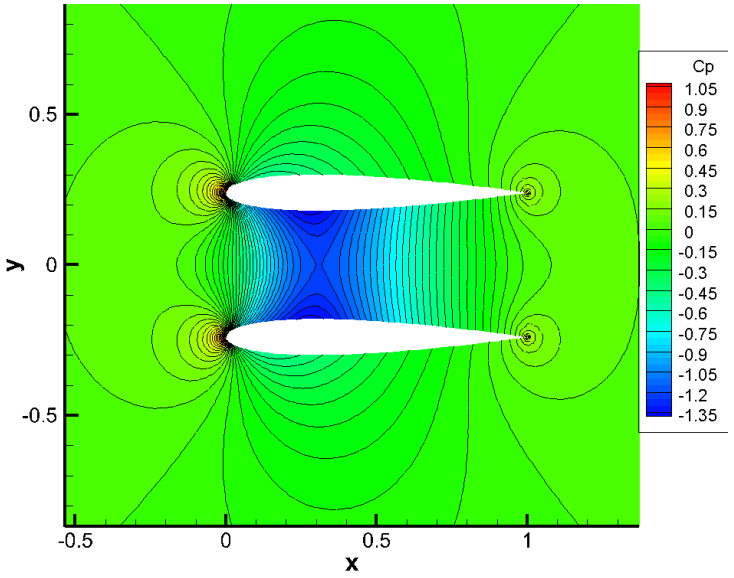


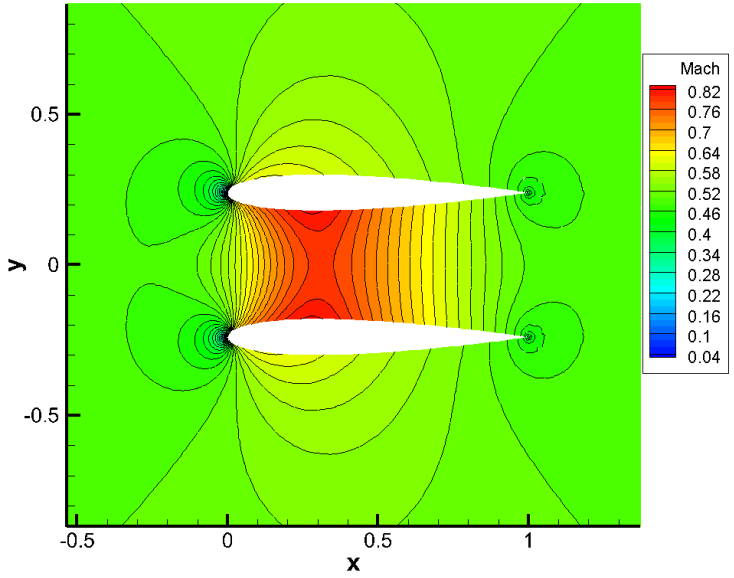


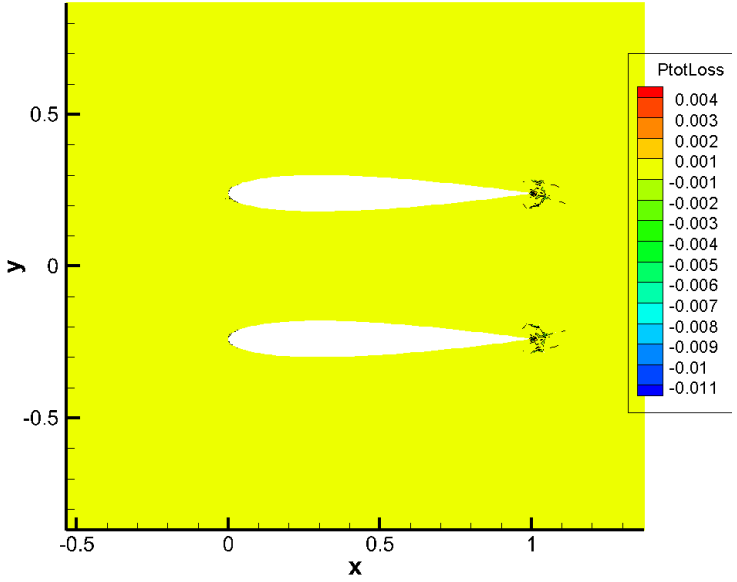
Preliminary results (p = 4, triangles, inviscid)

Visualization via linear sub-elements, not high order



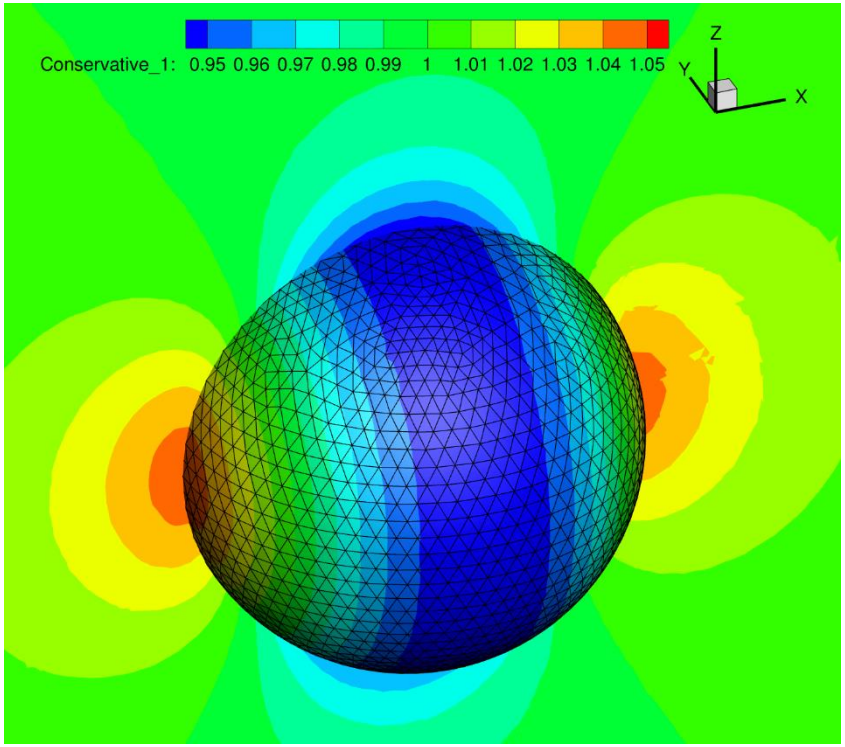




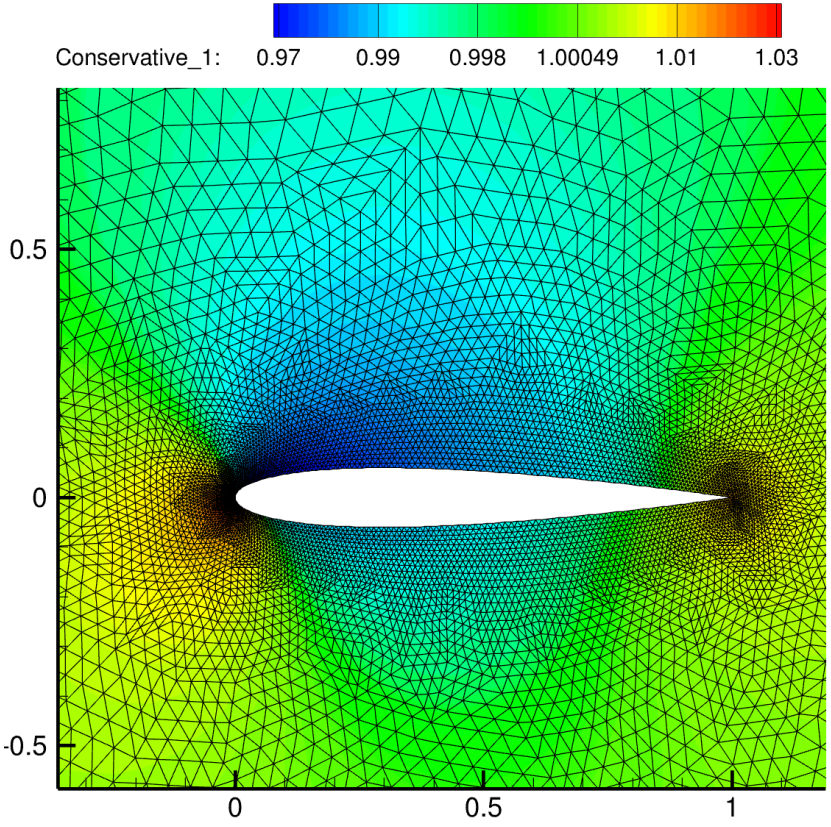


Preliminary results, inviscid

Visualization via linear sub-elements, not high order



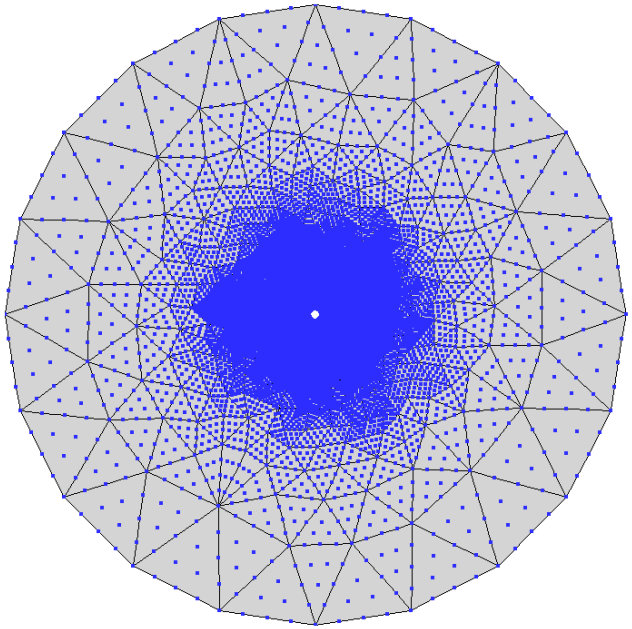
p = 3, tets

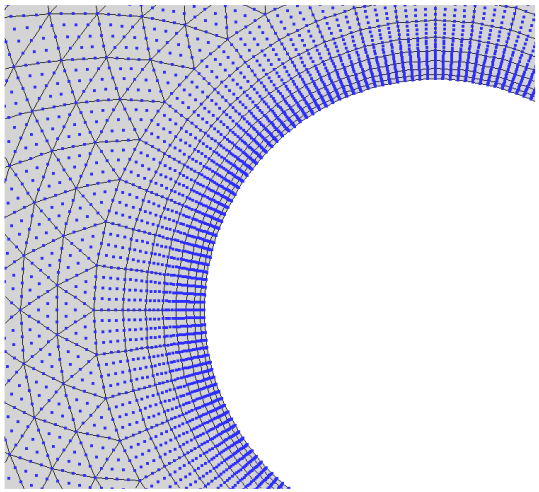


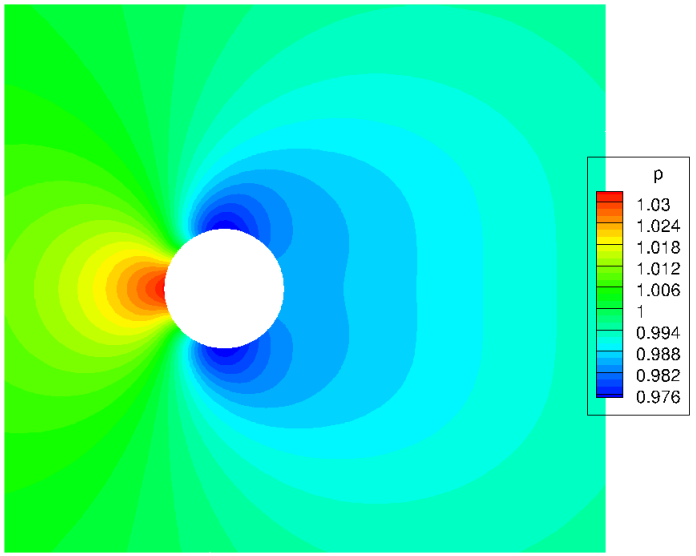
p = 4, triangles

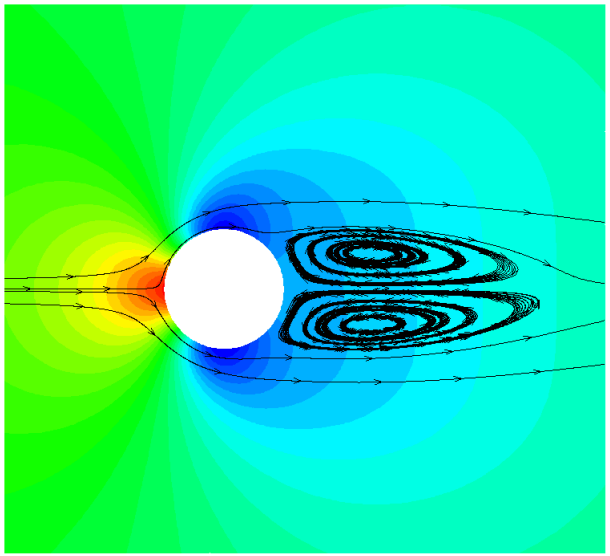
Preliminary results (p = 4, quads/triangles, viscous)

Visualization via linear sub-elements, not high order









Next things to be done

- Finalize debugging viscous terms (almost ready)
- Thorough V&V of the implementation
- Binary, parallel IO
- Shock capturing
- Time accurate local time stepping
- LES models and boundary conditions, including wall models
- Performance optimization, including OpenMP parallelization
- Grid motion, sliding mesh interfaces
- Grid sequencing, multigrid
- High order grid generation