

Higher-order **SU2**: DG-FEM and WENO-FV solvers with LES/ILES/WMLES applications

Edwin van der Weide

Department of Mechanical Engineering
University of Twente

UNIVERSITY OF TWENTE.

Juan J. Alonso, Jae hwan Choi,

Paul Urbanczyk, Eduardo Molina

Department of Aeronautics and Astronautics
Stanford University



Stanford
University

aerospace*design*lab

Dimitris Drikakis, Kevin Singh

Department of Mechanical and Aerospace Engineering
University of Strathclyde



Intel Corporation



Pointwise

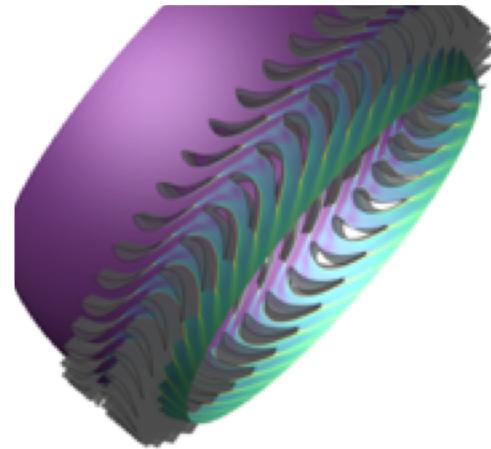
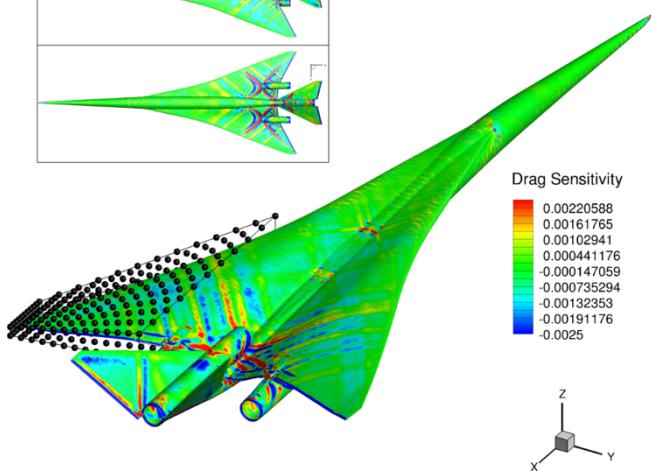
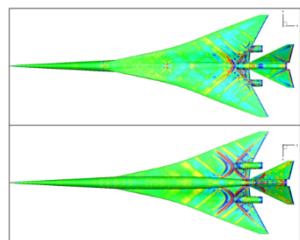
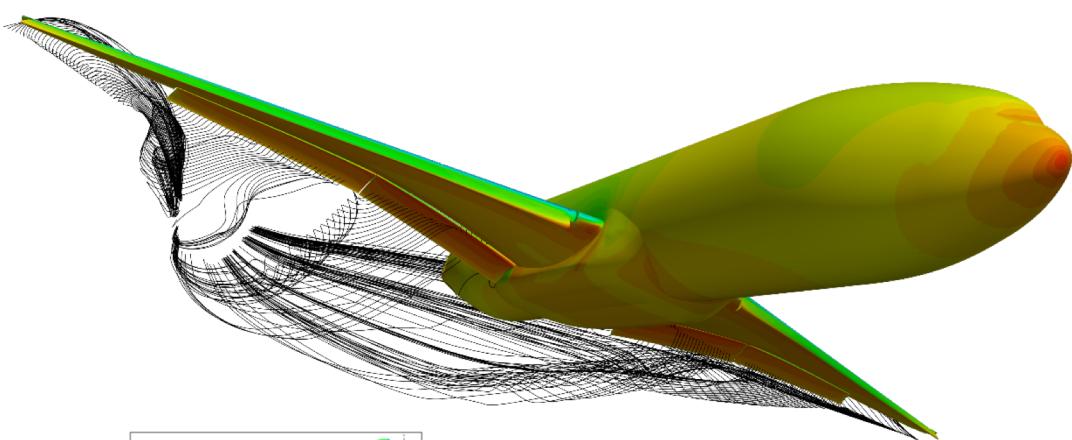


Outline

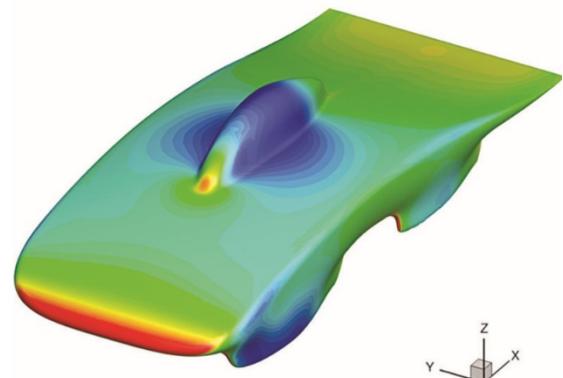
- Motivation for high-order schemes
- WENO solver
- DG solver
 - Shock capturing
 - LES models
 - Performance optimization
- Grid generation
- Access the code
- Future work

Why do we need high-order schemes?

2nd order schemes have been extremely successful, certainly for RANS. SU2 FV is used for many applications...



Pressure Coefficient: -0.6 -0.38 -0.16 0.06 0.28 0.5

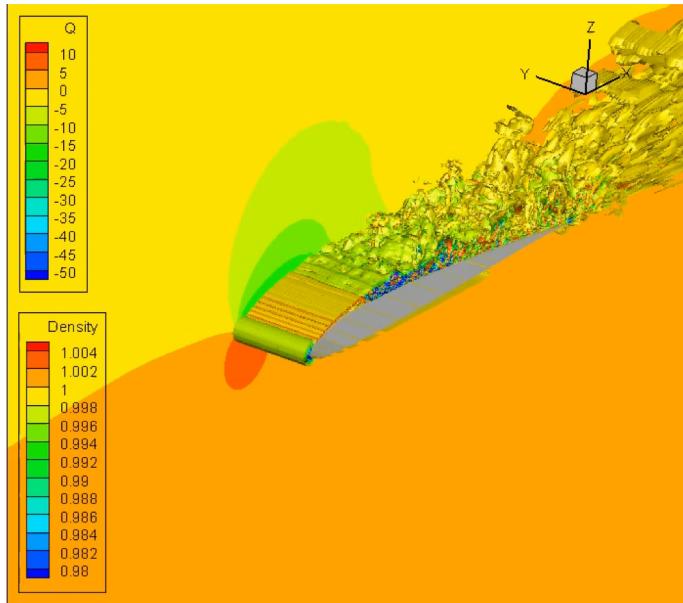


But...

For some applications 2nd order accuracy may not be sufficient

Examples

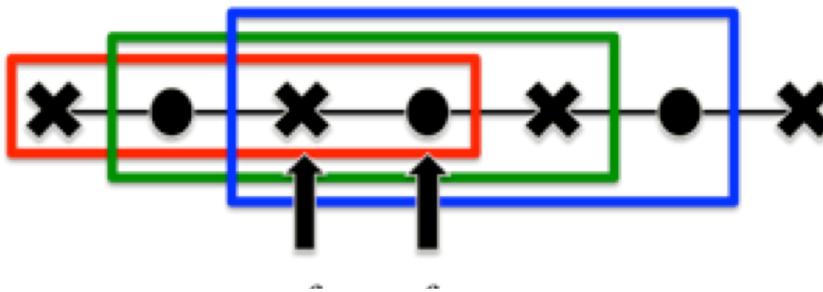
- Wake and vortex flows
- Noise prediction
- LES/DNS



High-order solver is intended for high-fidelity modeling of the turbulence, i.e. LES (wall resolved and wall modeled) and DNS

Options to increase the order of accuracy

1: Increase the stencil combined with smoothness indicators =>
WENO-FV



2: Increase the polynomial degree inside the element => DG-FEM



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

High-Order: SU2 Finite Volume

- Aim: Implementation of high-order schemes within the finite volume solver of SU2
- Objectives
 - 3rd order MUSCL limiters
 - Drikakis-Zoltak
 - Michalak & Ollivier-Gooch
 - Framework for Weighted Essentially Non-Oscillatory Schemes (WENO)
 - Use the Double Vortex Pairing Problem to assess the relative performance of these schemes

Description of Problem: Double Vortex Pairing

- Mixing layer formed by two co-flowing streams of water
- Initial velocity perturbations inflate forming two distinct vortices
- Vortices roll around each other eventually merging to form one vortical structure
- Chosen as test problem due to presence of fine structures and discontinuities



(a) 1.0 s



(b) 2.0 s



(c) 3.0 s



(d) 4.0 s

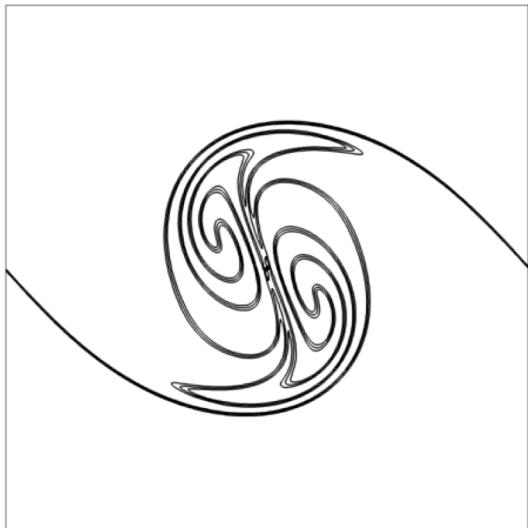


(e) 5.0 s

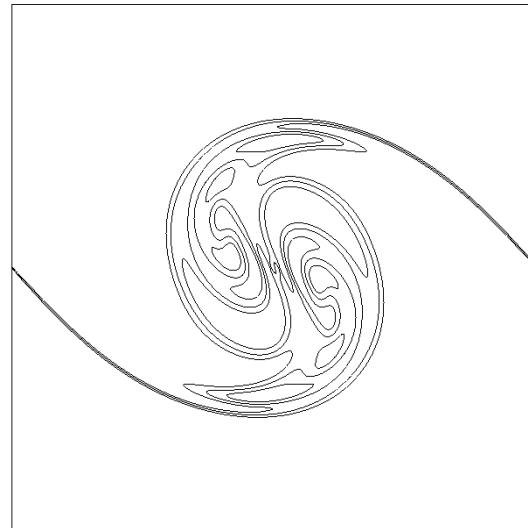


(f) 6.0 s

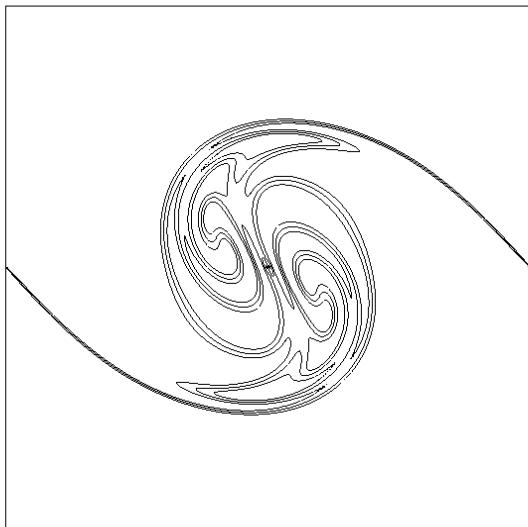
Double Vortex Pairing 256x256, M=0.2: Passive Scalar



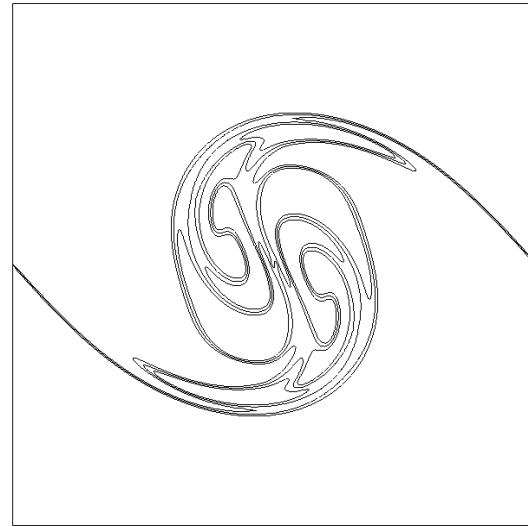
(a) WENO 11



(b) SU2: Venkatakrishnan



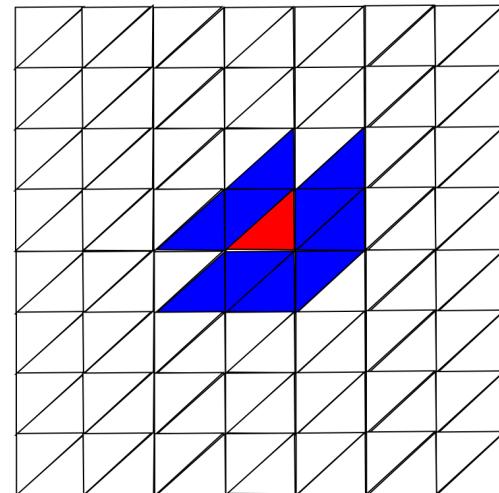
(c) SU2: Michalak-Ollivier Gooch



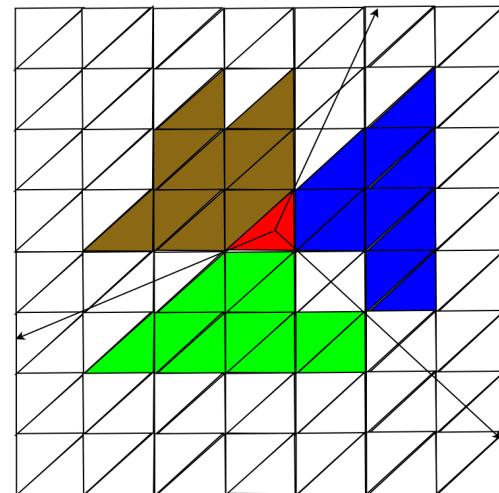
(d) SU2: Drikakis-Zoltak

WENO Outline

- Difficult to achieve orders of accuracy above 3rd order using MUSCL based approach on unstructured grids
- Aim: To create a high-order polynomial for target cell E_0 which has the same cell averaged value as the reconstructed function u
- Reconstruction uses cell averaged value from target cell E_0 as well as cell averaged values from multiple stencils consisting of neighboring cells
- Two types of stencils
 - Central
 - Directional
- Number of cells in the stencil scales with desired order of accuracy
 - $K = \frac{1}{2}(r + 1)(r + 2) - 1$
 - $M = 2K$



(a) Central Stencil



(b) Directional Stencil

WENO Outline

$$p_{weno} = \overline{u_0} + \sum_{k=1}^K \widetilde{a_k} \phi_k(\xi, \eta)$$

- Polynomials are constructed using data from each stencil
- “Essentially Non-Oscillatory” part stems from non-linear weights which are used to determine the smoothness of the solution in each stencil
- Central stencil given the largest bias since for smooth solutions the central stencil generally is the most accurate

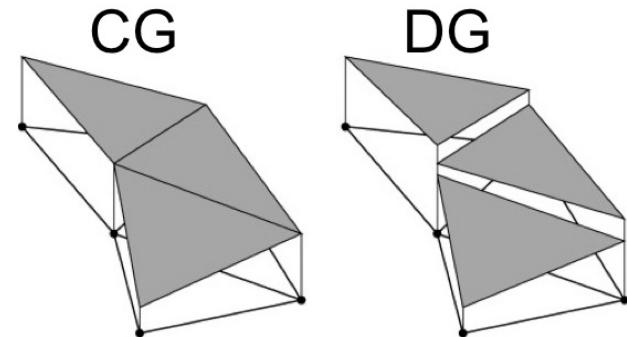
Current Status

- Geometrical Preprocessing (Commom/src/geometry_structure.cpp)
 - Check the mesh to determine which elements can have WENO reconstruction
 - Central + Directional stencil assembly for triangular elements
 - Obtain unique nodes (solution points) from stencil
 - Functions to handle assembly of mesh dependant parameters
 - LAPACK functions used to carry out matrix operations
 - Function to determine which two elements share a common edge
- Solution Reconstruction (SU2_CFD/src/solver_direct_mean.cpp)
 - MUSCL reconstruction used in region of the grid which cannot have WENO reconstruction
 - 3rd order WENO reconstruction done for remainder of domain
- Some Issues
 - Oscillations present at discontinuities
 - ~ 10 - 100 times slower than MUSCL scheme (depending on grid resolution)

Future Work

- Fully Functional in 2D
 - Quadrilateral elements
 - Hybrid grids
- Extension to higher-orders (4th , 5th etc)
- Full functionality in 3D
 - Requires additional steps in the geometrical preprocessing
 - Data structures require extension to handle the extra necessary information
- Add tunable parameters as options to be read from config file
- Parallel implementation
- Performance optimizations!

DG-FEM: the basic principles



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

System of PDE's: Weak formulation

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

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

Integrals are computed with high enough accuracy to prevent instabilities due to aliasing errors. Computationally intensive!!

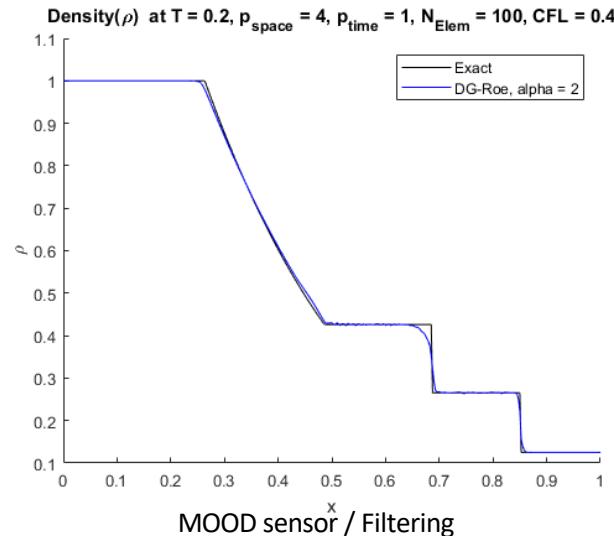
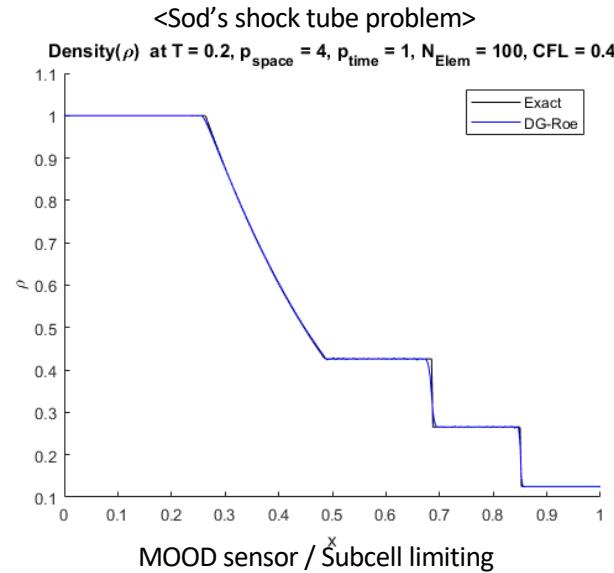
However: DG solver can be run very efficiently on modern hardware.
Also hybridized DG techniques to reduce CPU requirements.

Current Capabilities

- Both 2D and 3D, just like SU2-FV
- All standard elements (tri, quad, tet, pyra, prism, hex)
- Curved elements of arbitrary order
- Polynomial order can differ in individual elements
- Symmetric Interior Penalty method for viscous fluxes
- Explicit time integration schemes (Runge-Kutta type)
- Time-accurate local time stepping via ADER-DG
- Preliminary implementation of LES models and shock capturing

Shock capturing

- Shock capturing relies on two components:
 - Sensing the discontinuity
 - Resolving the discontinuity
- Sensing the discontinuity:
 - Persson and Peraire : Modal decay
 - Clain, Diot and Loubere : MOOD
- Resolving the discontinuity
 - Limiter
 - Artificial viscosity / filtering
 - Sub-cell limiting

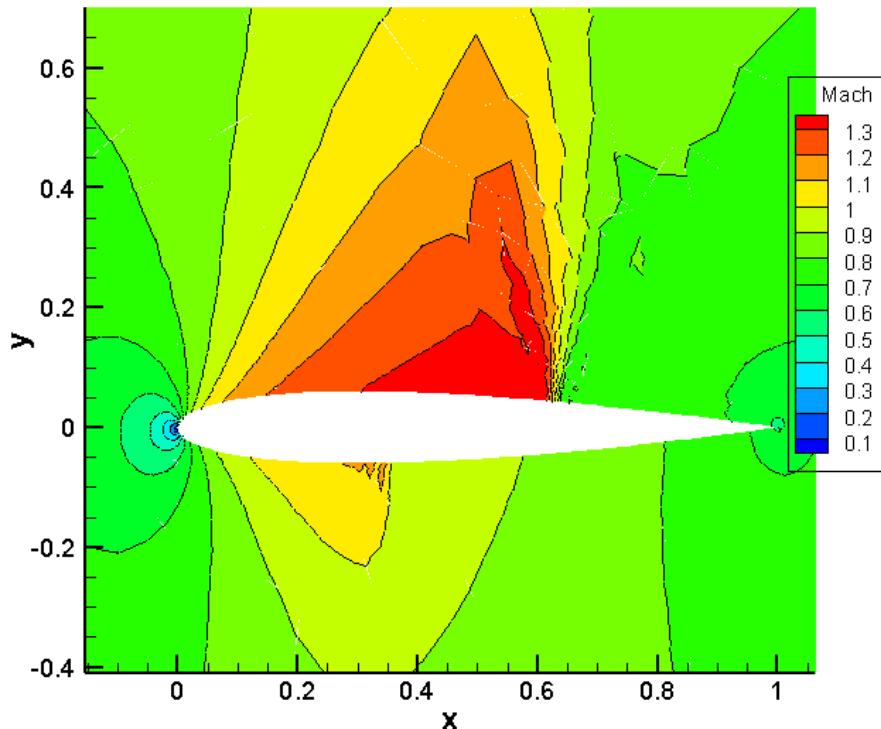


Shock capturing

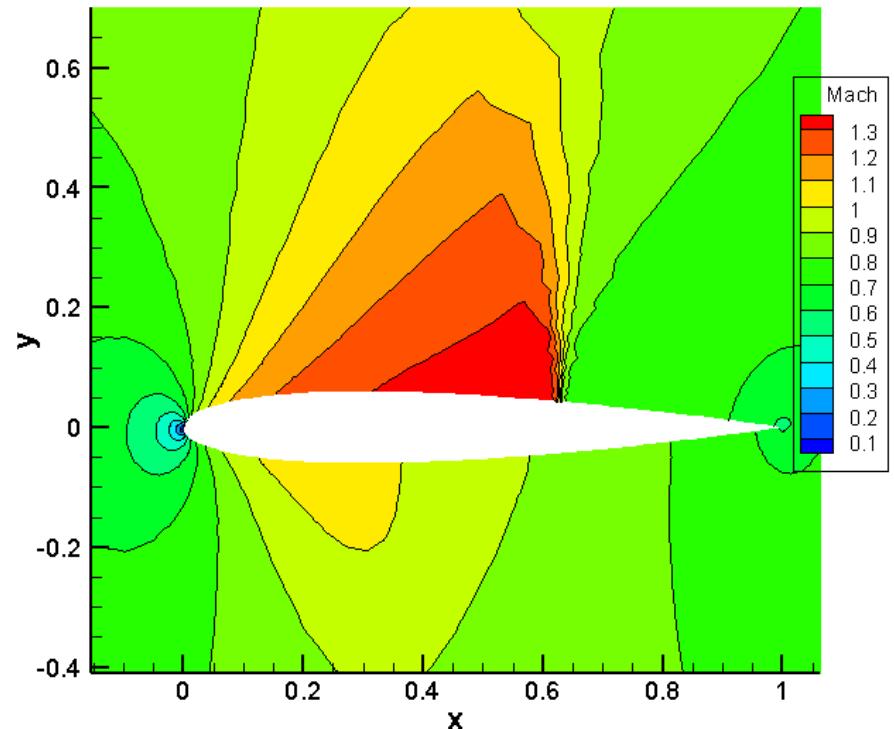
- Shock capturing method in DG-FEM
 - Discontinuity sensor : Use the ratio of the norms of the highest and the lowest modal coefficients.
(Extension of Persson's shock sensor)
 - Resolving method : Use filtering method where the filtering strength is determined by sensor value.
- Current status
 - 2D triangular elements up to $p = 3$
 - Filtering strength can be modified with one input parameter

Shock capturing

- Transonic flow over NACA0012 airfoil
 - $p = 1$ | 7,990 triangles | 119 elements on each surface | 238 DOFs on each surface



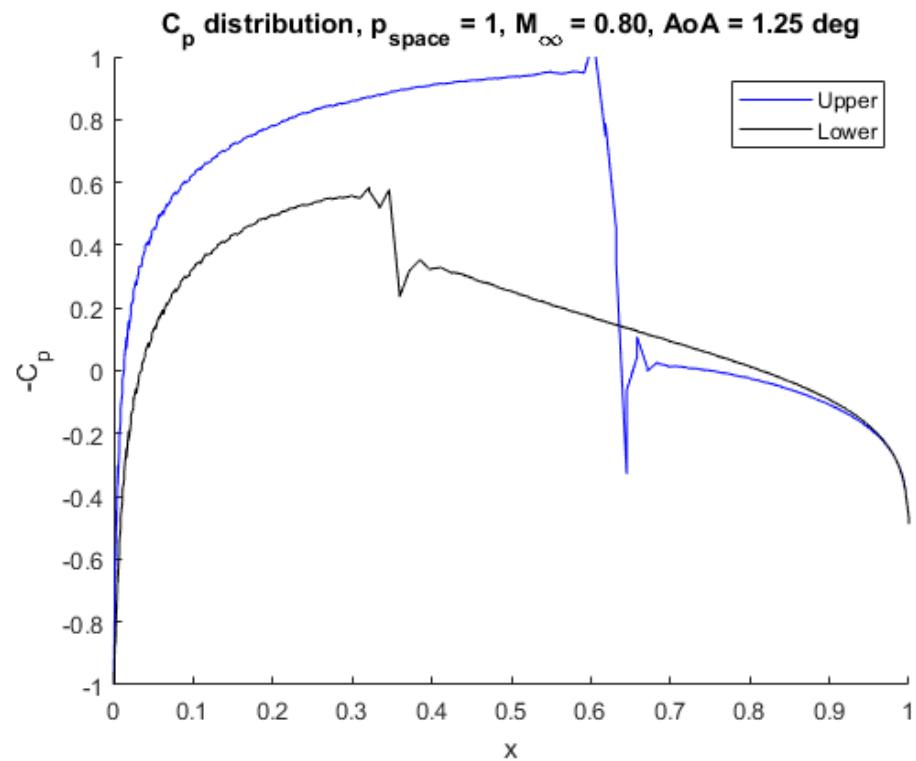
Lax-Friedrich, DG



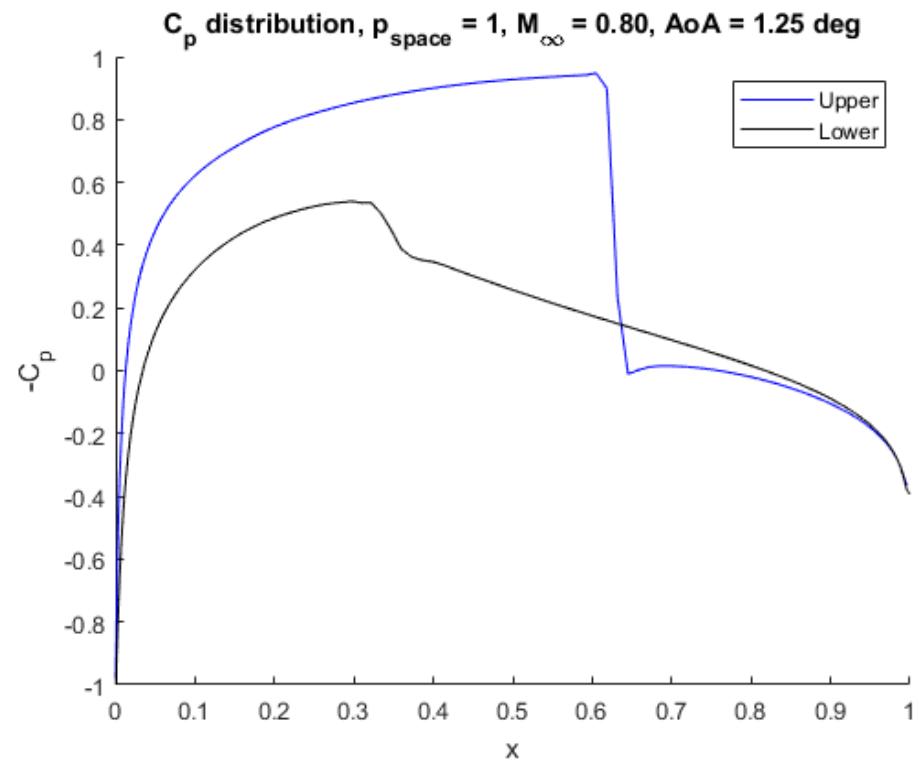
Roe, FVM, Venkatakrishnan Slope Limiter

Shock capturing

- Transonic flow over NACA0012 airfoil
 - $p = 1$ | 7,990 triangles | 119 elements on each surface | 238 DOFs on each surface



Lax-Friedrich, DG

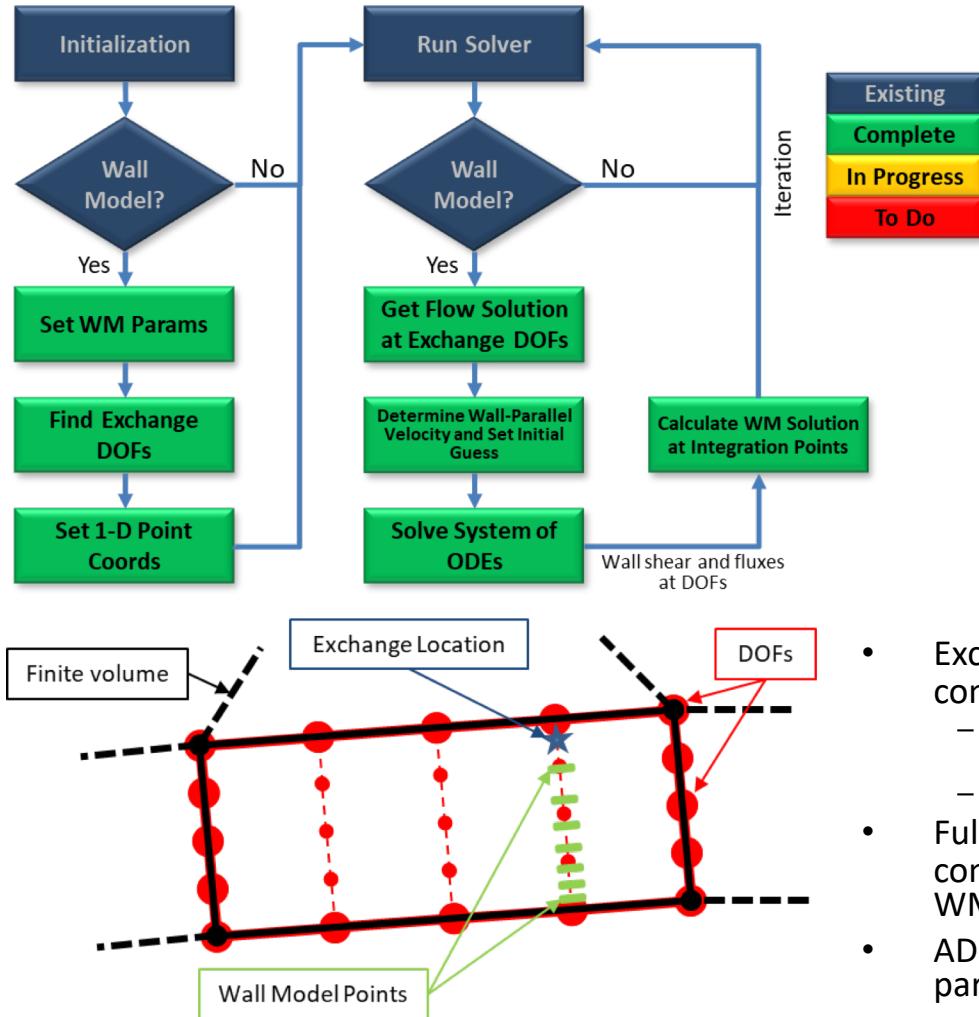


Roe, FVM, Venkatakrishnan Slope Limiter

LES Models

- SGS Models
 - Constant Smagorinsky
$$\nu_{sgs} = C_s^2 \Delta^2 |\tilde{S}|$$
 - Wall-Adapting Local Eddy Viscosity (WALE)
$$\nu_{sgs} = (C_w \Delta)^2 \frac{(S_{ij}^d S_{ij}^d)^{(3/2)}}{(\widetilde{S}_{ij} \widetilde{S}_{ij})^{(5/2)} + (S_{ij}^d S_{ij}^d)^{(5/4)}}$$
 - More sophisticated models to be implemented in future
 - Dynamic Smagorinsky, etc.
 - Wall Models
 - One-dimensional Equilibrium BL Equations
-
- Implementation of wall models within the context of DG-FEM solvers.
A. Frére, C. de Wiart, K. Hillewaert, P. Chatelain, G. Winckelmans, Phys of Fluids, Jul 2017

Development of Wall-Models



- One-Dimensional Equilibrium Model (1DEQM) of Larsson / Kawai / Bodart

- Mixing length model with wall-damping used for turbulent viscosity

$$\frac{d}{dy} \left[(\mu - \mu_t) \frac{d\tilde{u}_\parallel}{dy} \right] = 0$$

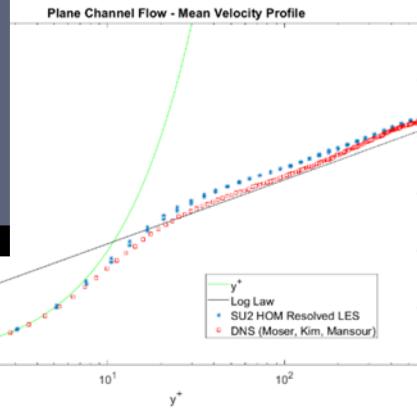
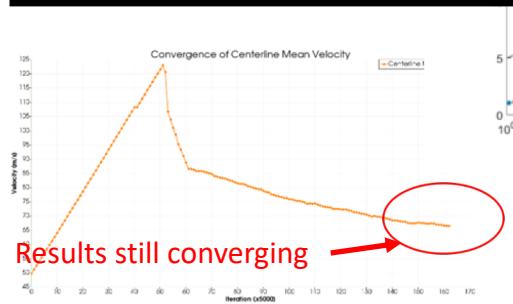
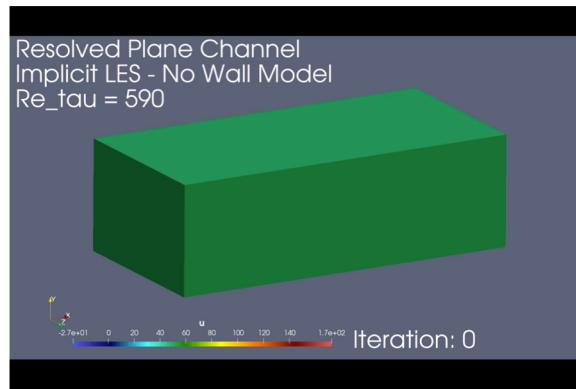
$$\frac{d}{dy} (\bar{p}) = 0$$

$$\frac{d}{dy} \left[\tilde{u}_\parallel (\mu + \mu_t) \frac{d\tilde{u}_\parallel}{dy} + \left(\frac{\mu c_p}{Pr} + \frac{\mu_t c_p}{Pr_t} \right) \frac{d\tilde{T}}{dy} \right] = 0$$

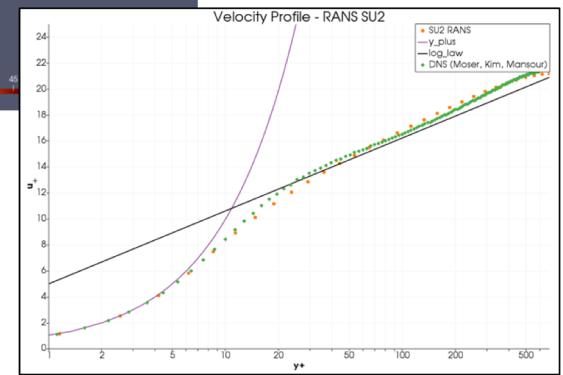
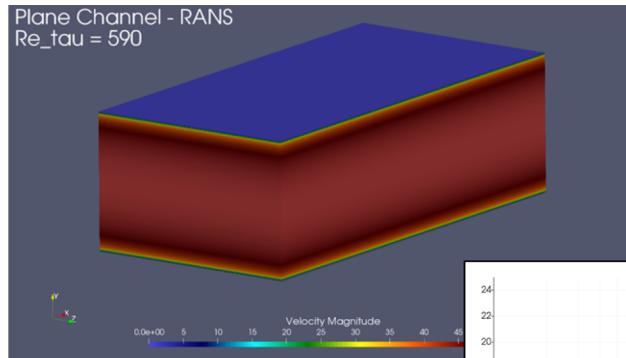
- Exchange location permitted in large-scale parallel computations:
 - On any element type on the surface (hexa, tetra, pyramid, prism)
 - Not necessarily on first element on the surface
- Full parallel search capability (ADT based), all necessary communication, and implementation of equilibrium WM have now been completed and being tested
- ADER-DG time-step requirements respected in partitioning / search

Wall-Model Validation: Plane Channel Flow

- Plane Channel simulations conducted with RANS and with *resolved* LES for comparison with WMLES
 - Flow $Re_\tau = 590$
 - Friction length $l_\tau = 1.7 \times 10^{-10} \text{ m}$
 - Friction velocity $u_\tau = 2.53 \text{ m/s}$
 - Channel half-height $\delta = 0.1 \text{ m}$
 - Domain size = $2\pi\delta \times 2\delta \times \pi\delta$



Resolved LES
Simulation
Mesh: 22x22x22
 $P = 3$



- Debugging/testing of plane channel flow with SU2 DG-FEM wall-modeled LES underway

Performance optimization (with Intel)

- First implementation was very inefficient (< 5% peak on Xeon)
- Collaboration with Intel to improve efficiency
 - Specialized matrix multiplication software (MKL, LIBXSMM)
 - Explicit unrolling of small loops (specialized 2D, 3D code)
 - Vectorization direction matrix multiplication: 128 byte aligned
 - Element-wise operation fusion for vectorization
 - Gemm calls: \approx 60% peak on Xeon
- Performance entire code: \approx 35-40% peak on Xeon (single core)
- Hybrid MPI-OpenMP to increase flexibility (about to start)

Performance analysis tools (Intel)

Intel VTune Amplifier

Welcome x r002hs x

Basic Hotspots Hotspots by CPU Utilization

Analysis Configuration Collection Log Summary Bottom-up Caller/Callee Top-down Tree Platform

INTEL VTUNE AMPLIFIER 2019

INSIGHTS

Elapsed Time: 545.030s

CPU Time: 543.560s
Total Thread Count: 1
Paused Time: 0s

Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
cblas_dgemm	libmkl_intel_lp64.so	351.361s
CFEM_DG_NSSolver::ADER_DG_AliasedPredictorR	SU2_CFD	48.980s
residual_3D		
CFEM_DG_NSSolver::Volume_Residual	SU2_CFD	19.790s
cblas_dgemv	libmkl_intel_lp64.so	19.270s
CFEM_DG_EulerSolver::ADER_DG_PredictorStep	SU2_CFD	13.500s
[Others]		90.659s

*N/A is applied to non-summable metrics.

Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.

Elapsed Time

Average Effective CPU Utilization

Idle

Poor

Ok

Ideal

Target Utilization

Simultaneously Utilized Logical CPUs

Performance analysis tools (Intel)

File View Help

Start Survey Analysis | ?

Welcome e000

Vectorization Workflow Threading Workflow

OFF Batch mode Run Roofline Collect Enable Roofline with Callstacks

1. Survey Target Collect Mark Loops for Deeper Analysis Select checkboxes in the Survey & Roofline tab to mark loops for other Advisor analyses.
-- There are no marked loops --

1.1 Find Trip Counts and FLOP Collect Trip Counts FLOP Analyze all loops --

2.1 Check Memory Access Patterns Collect No loops selected --

2.2 Check Dependencies Collect No loops selected --

INTEL ADVISOR 2019

Elapsed time: 154.02s Vectorized Not Vectorized MKL FILTER: All Modules All Sources

Summary Survey & Roofline Refinement Reports

Vectorization Advisor

Vectorization Advisor is a vectorization analysis toolset that lets you identify loops that will benefit most from vector parallelism, discover performance issues preventing from effective vectorization and characterize your memory vs. vectorization bottlenecks with Advisor Roofline model automation.

Program metrics

Elapsed Time	154.02s
Vector Instruction Set	AVX512, AVX2, AVX, SSE2, SSE
Total GFLOP Count	6054.94
Total Arithmetic Intensity	0.85148
Number of CPU Threads	1
Total GFLOPS	39.31

Loop metrics

Metric	Total	MKL det...
Total CPU time	153.58s	100.0%
Time in 49 vectorized loops	64.55s	42.0%
Time in scalar code	89.03s	58.0%
Total GFLOP Count	6054.94	100.0%
Total GFLOPS	39.31	

Vectorization Gain/Efficiency Vectorized Loops Gain/Efficiency 6.28x ~87% Program Approximate Gain 1.19x

Per program recommendations

Your application might be underperforming Consider recompiling your application with zmm registers enabled. [Show more](#)

Top time-consuming loops

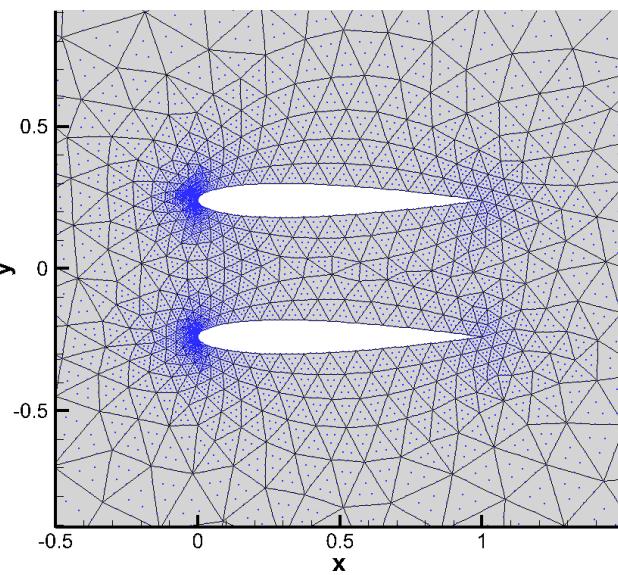
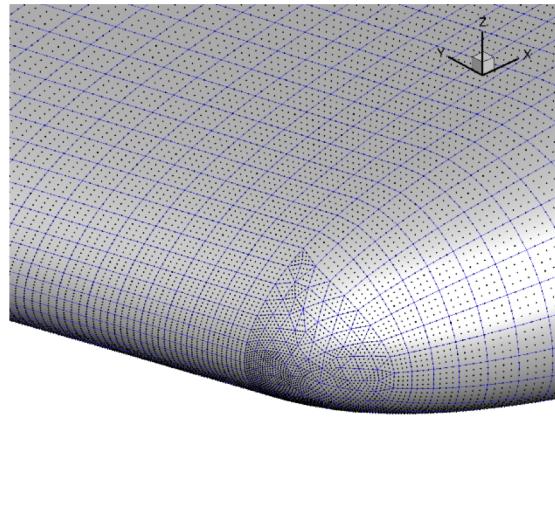
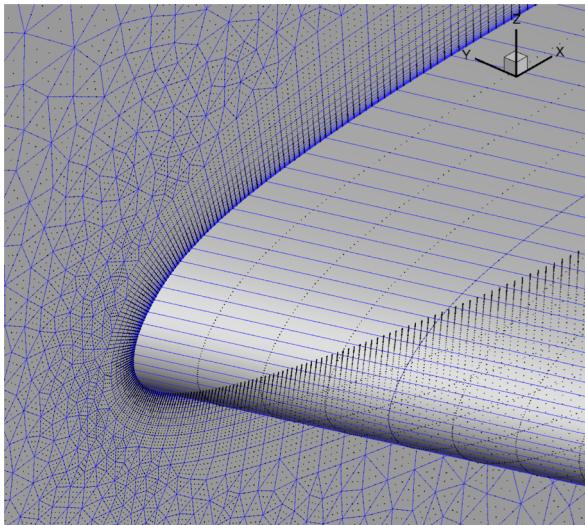
Loop	Self Time	Total Time	Trip Counts
[loop in MKL BLAS at ?]	53.649s	53.649s	8
[loop in CFEM_DG_NSSolver::ADER_DG_AliasedPredictorResidual_3D at solver_direct_mean_fem.cpp:10987]	8.700s	13.520s	64
[loop in CFEM_DG_NSSolver::ADER_DG_AliasedPredictorResidual_3D at solver_direct_mean_fem.cpp:11146]	7.460s	7.460s	125
[loop in CFEM_DG_NSSolver::Volume_Residual at solver_direct_mean_fem.cpp:12567]	7.120s	11.210s	125
[loop in CFEM_DG_EulerSolver::ComputeInviscidFluxesFace at solver_direct_mean_fem.cpp:8891]	5.200s	5.200s	25

Recommendations

Loop	Self Time	Recommendations
[loop in CFEM_DG_NSSolver::ResidualFaces at solver_direct_mean_fem.cpp:13143]	0.170s	Disable unrolling
[loop in CFEM_DG_NSSolver::BC_HeatFlux_Wall at solver_direct_mean_fem.cpp:14903]	0.070s	Disable unrolling
[loop in CFEM_DG_NSSolver::ResidualViscousBoundaryFace at solver_direct_mean_fem.cpp:15560]	0.020s	Disable unrolling
[loop in PhysicalGeometry::DetermineFEMConstantJacobiansAndLenScale at geometry_structure_fem_part.cpp:3381]	0.010s	Disable unrolling
[loop in MKL BLAS at ?]	0s	Vectorize serialized function(s) inside loop

Re-finalize Sur...

Higher-order Grids with Curved Elements (with Pointwise, work of Steve Karman)



In-house capability to generate higher-order grids for simple cases.

Pointwise V18.2 will have degree elevation and mesh curving capabilities. Available end of September 2018.

Access the code

- *feature_hom* branch on GitHub is the main branch for the DG solver. It is about to be merged with develop.
- Several other development branches exist

SU2 Suite <https://su2code.github.io>

The screenshot shows a GitHub repository page for the SU2 Suite. At the top, there are statistics: 3,713 commits, 109 branches, 33 releases, 37 contributors, and a license of LGPL-2.1. Below this is a navigation bar with buttons for 'Branch: master' (selected), 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. A dropdown menu titled 'Switch branches/tags' is open, showing a list of branches and tags. The 'feature_hom' branch is highlighted with a blue border. The main area displays a list of recent commits, all of which are 6 months old. The commits are listed under the 'feature_hom' branch, with file changes like 'File header change' and folder changes like 'SU2_PY', 'SU2_SOL', and 'TestCases'.

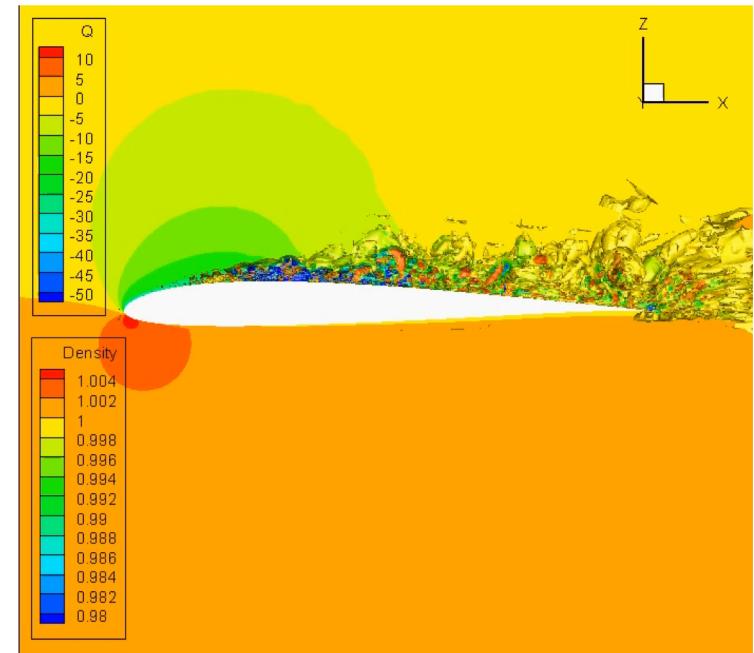
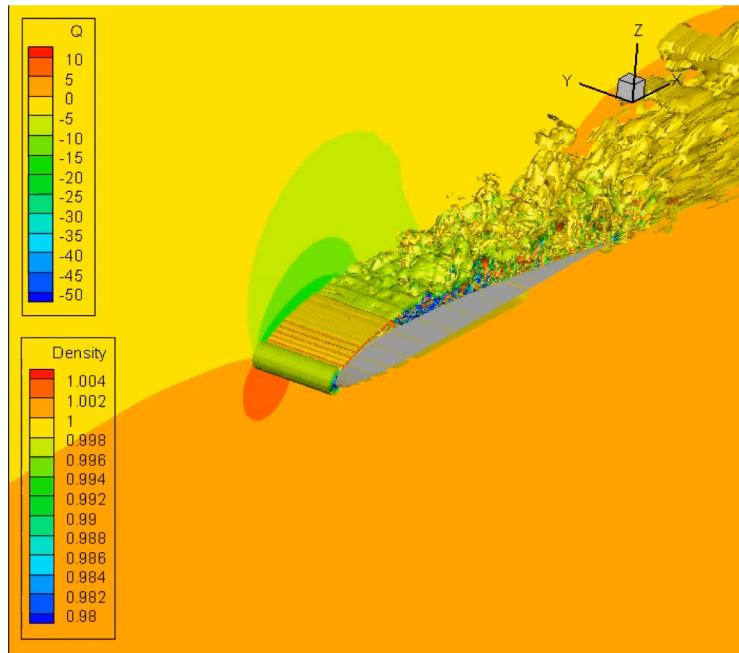
Commit	Message	Date
feature_hom	File header change	6 months ago
feature_hom_Jacobian	File header change	6 months ago
feature_hom_gemm_test	File header change	6 months ago
feature_hom_output	File header change	6 months ago
feature_hom_singleNodeOpt	File header change	6 months ago
feature_hom_wallModel	File header change	6 months ago
feature_hom	File header change	6 months ago
SU2_PY	File header change	6 months ago
SU2_SOL	File header change	6 months ago
TestCases	File header change	6 months ago

Future Work DG-FEM

- Finish Shock Capturing and LES wall models
- LES statistics (common to DDES and URANS)
- Improve boundary conditions (non-reflective)
- Improve time step estimates by eigenvalue analysis of the Jacobian matrix (computed with CodiPack)
- Parallel performance optimization, including OpenMP
- Entropy variables
- Hybridized techniques, e.g. Embedded Discontinuous Galerkin (EDG), to reduce computational requirements
- Implicit algorithms
- Discrete adjoint version
- Verification and Validation (Manufactured solutions)

Results, viscous

Implicit LES, SD 7003 (Reynolds = 60,000)



$p = 4$, hexahedra