Algebraic multigrid and defect correction for solution of adjoint equations in compressible aerodynamics

Malte Förster, Anna Naumovich

FlowHead Conference on Industrial Design Optimization
for Fluid Flow

Munich, 28-29 March 2012

Outline

- Discrete adjoint equations in DLR TAU code
- Evaluation of linear residual
- Available solution approaches
- Algebraic multigrid
- Inner and outer defect correction
- Results

DLR TAU code. Discretization of flow equations.

TAU is a fully parallelized 2nd-order finite volume flow solver for unstructured hybrid grids solving the Reynolds-Averaged Navier-Stokes equations or the Euler equations.

Consider a FV discretization of flow equations:

$$R(W, X, D) = 0$$

W	- Flow variables
X	- Mesh coordinates
D	- Design variable(s)

Inviscid flux: 2.-order Central Jameson-Schmidt-Turkel Scheme with either **scalar** or **matrix** dissipation.

Viscous flux: Green-Gauss or TSL approximation

Turbulence equations: SAO, SAE, k-omega

Adjoint equations

Minimization problem

$I(W, X, D) \longrightarrow min$ - cost function

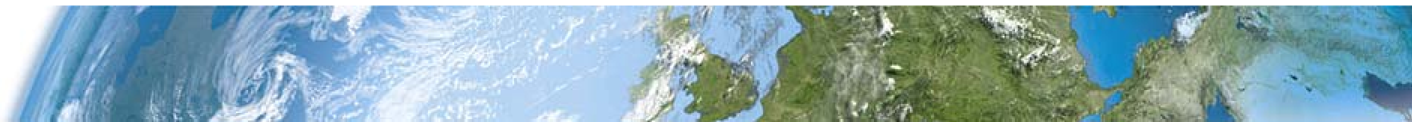
$$\left.\begin{array}{ll} R(W, X, D) = 0 & \text{- resid. of flow eqs} \\ T(X, D) = 0 & \text{- mesh deformation} \end{array}\right\} \text{constraints}$$

W - Flow variables

X - Mesh coordinates

D - Design variable(s)

Lagrangian $L = I + \Lambda^T R + \hat{\Lambda}^T T$



Adjoint equations

Gradient evaluation

$$\frac{dI}{dD} = \frac{\partial I}{\partial D} + \Lambda^T \frac{\partial R}{\partial D} + \hat{\Lambda}^T \frac{\partial T}{\partial D}$$

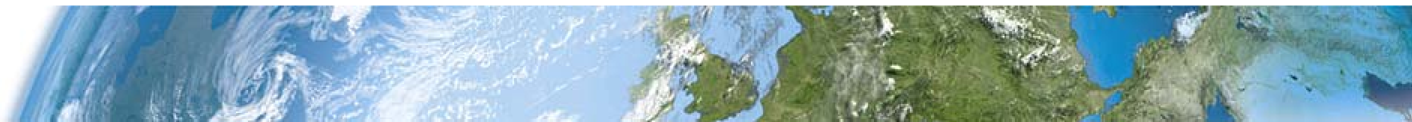
Flow adjoint equation

$$\boxed{\left(\frac{\partial R}{\partial W}\right)^T \Lambda = -\left(\frac{\partial I}{\partial W}\right)^T}$$

In TAU, the derivatives are obtained after hand differentiation of all terms

Mesh adjoint equation

$$\boxed{\left(\frac{\partial T}{\partial X}\right)^T \hat{\Lambda} = -\left(\frac{\partial I}{\partial X}\right)^T - \left(\frac{\partial R}{\partial X}\right)^T \Lambda}$$



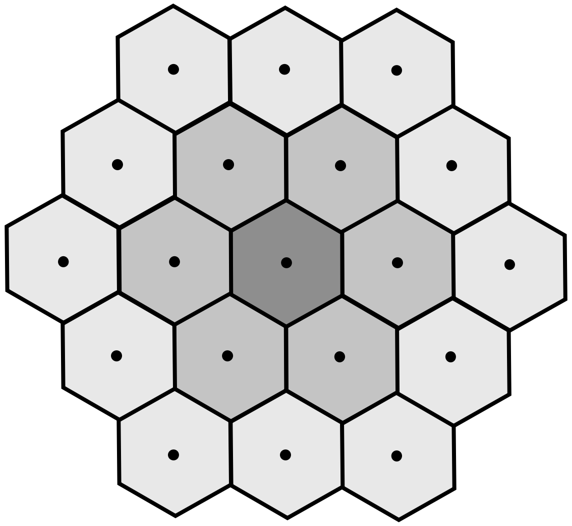
Adjoint equations

$$\boxed{\left(\frac{\partial R}{\partial W}\right)^T \Lambda = -\left(\frac{\partial I}{\partial W}\right)^T}$$

Large sparse system of linear algebraic equations

$$\downarrow$$

$$\boxed{Ax = b}$$



2D: neighbors and next-neighbors of a grid point

Sparsity of A is defined by number of neighbors and next neighbors of each grid point.

The amount of next-neighbors is higher on unstructired grids and can reach ~ 80!

Linear residual evaluation in TAU code

Linear residual is $r = b - Ax$

How to evaluate Ax ? There are two ways in TAU code:

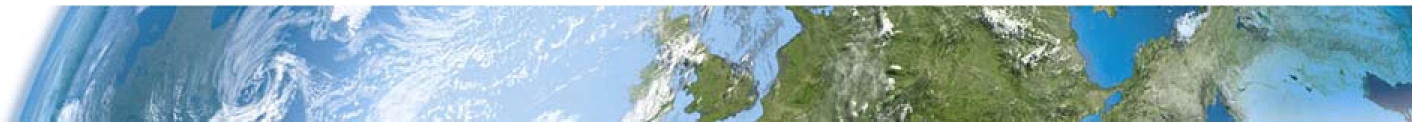
Facemat residual (Ax on-the-fly)

Pre-compute and store parts of A (~1/3 of full Jac.). However, due to discretiazion used in TAU it is very complicated to obtain the exact linear residual in this way.
A.t.m. this approach **relies on a number of simplifying assumptions** concerning differentiation of dissipation coefficients.
Obtained linear residual is **not exact**.

PETSc residual (full Jac. storage)

Relies on usage of PETSc library

Compute and store A in a sparse matrix data structure (**expensive**).
In TAU, this approach gives exact linear residual.



Solution approaches in TAU

1.

Linear geometric multigrid

Linear version of TAU MG

RK or LU-SGS smoother

GMRes to stabilize / accelerate

Practical experience

- Rather cheap approach (memory-wise)
- However, convergence of the solver is not very good (often slow or stagnates)

2.

PETSc library solvers

ILU(n)-based preconditioners
RCM-reordering
(in parallel combined with Block-Jacobi or Additive Schwarz)

Krylov to stabilize / accelerate

Practical experience

Very efficient for 2D cases
If converges, it is hard to compete with it.
For 3D cases might require high levels of ILU, not always affordable

Requires storage of full Jacobian matrix

3. NEW

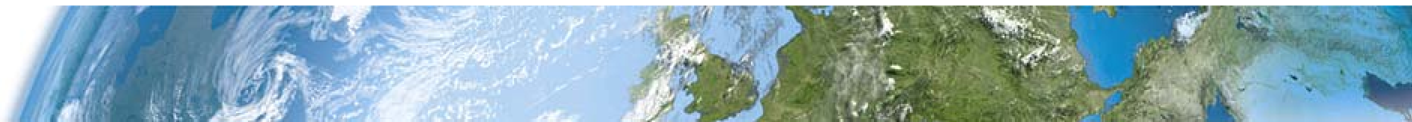
Solvers based on algebraic multigrid and defect correction

Algebraic multigrid solvers from the SAMG library (Fraunhofer SCAI) combined with inner or outer defect correction

Practical experience

A.t.m. the solver is being evaluated in the institute.

Requires storage of 1.-order Jacobian matrix



Solvers based on AMG and Defect Correction

- The work on these solvers is a joint work between DLR and Fraunhofer SCAI, done within ComFliTe Project.
- We use the **SAMG package** (product of Fraunhofer SCAI)
- **SAMG package** offers a large multi-level environment, based on AMG methodology, and provides a large set of components needed for a definition of various AMG algorithms.

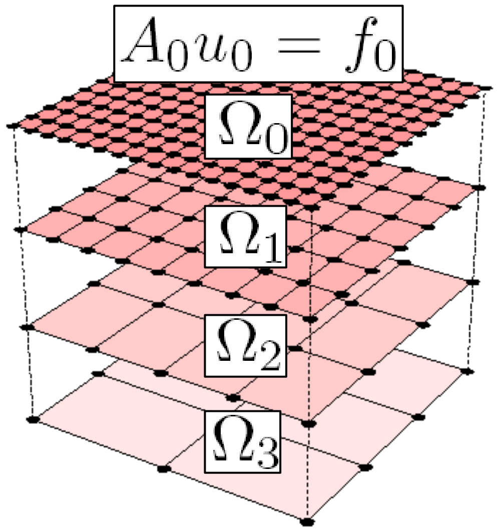
Algebraic multigrid

- The main idea of AMG is the same as of geometric multigrid: efficient interplay of smoothing and coarse grid correction.
- The only input required by AMG is the matrix and the right-hand side vector (no need for any geometric information).
- AMG is advantageous for complex domains, unstructured grids, discontinuous coefficients and anisotropies.
- Optimal method for scalar elliptic equations: for other types of problems special extensions are needed.

Algebraic vs Geometric multigtid

Geometric multigrid

Algebraic multigrid



Restriction

&

prolongation

$$\boxed{A_0 u_0 = f_0}$$

$$A_1 u_1 = f_1$$

$$A_2 u_2 = f_2$$

$$A_3 u_3 = f_3$$

Towards defect correction approach

Observations:

Direct application of any available AMG configuration to 2nd-order accurate matrices (target discretizations) was not successful

However, 1st-order accurate discretizations could be solved by AMG solvers very efficiently

Exploit it in a Defect Correction approach

Two approaches

Solution of 2nd-order discretizations

$\Downarrow$ $\Downarrow$

Outer defect correction

- Perform defect correction with 1st-order matrix on the l.h.s.
- Apply AMG at each iteration to solve (very approximately) the 1st-order system.
- Use ILU(0) as a smoother in AMG

Inner defect correction
(as a fine level smoother in SAMG)

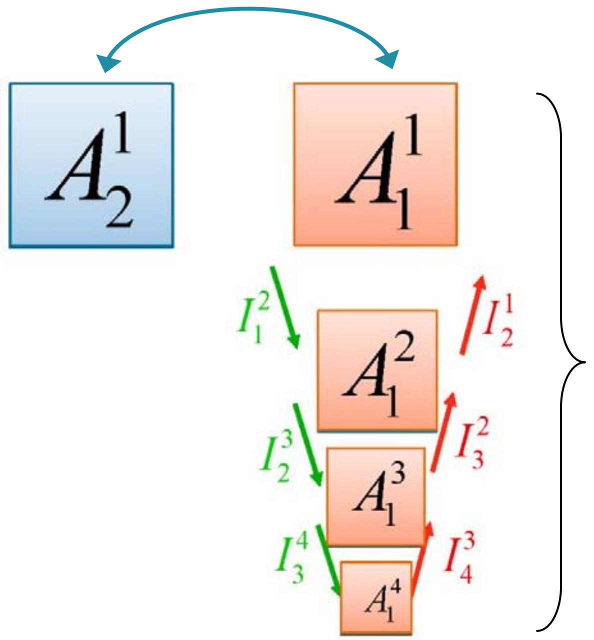
- Within AMG, apply defect correction combined with ILU(0) as a smoother on the finest level
- Apply just ILU(0) as a smoother on coarse levels (on coarse levels only 1.-order accurate)

Simplified in comparison with the initial version.

Additionally, we apply a Krylov method (GMRES) to stabilize each of the approaches

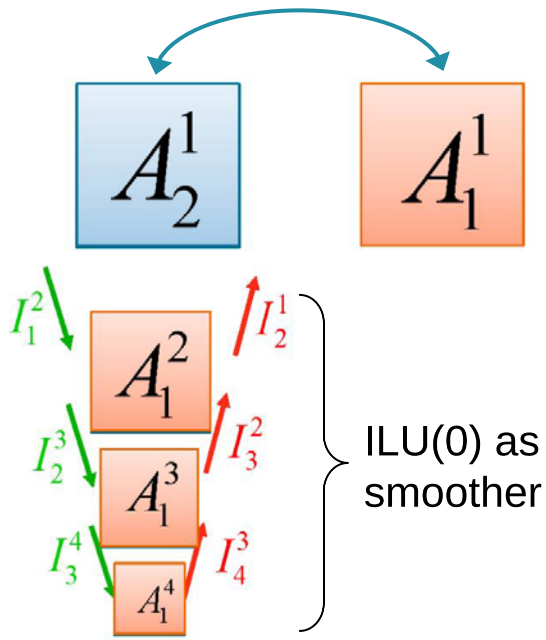
Two approaches

AMG & Outer defect correction



ILU(0) as a smoother

AMG & Inner defect correction



ILU(0) to „invert“ A_1

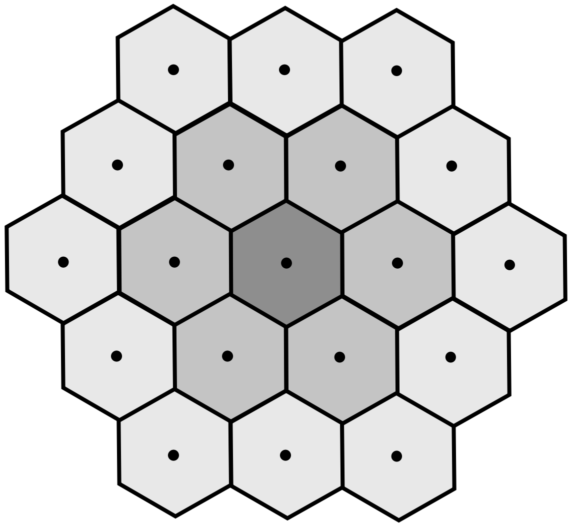
ILU(0) as a smoother

AMG combined with outer Defect Correction

Target 2nd-order problem: $Ax = b$

Auxiliary 1st-order operator: A_1

Employ AMG @ each iteration.
In most practical applications we only use 1 step of AMG per defect correction step.

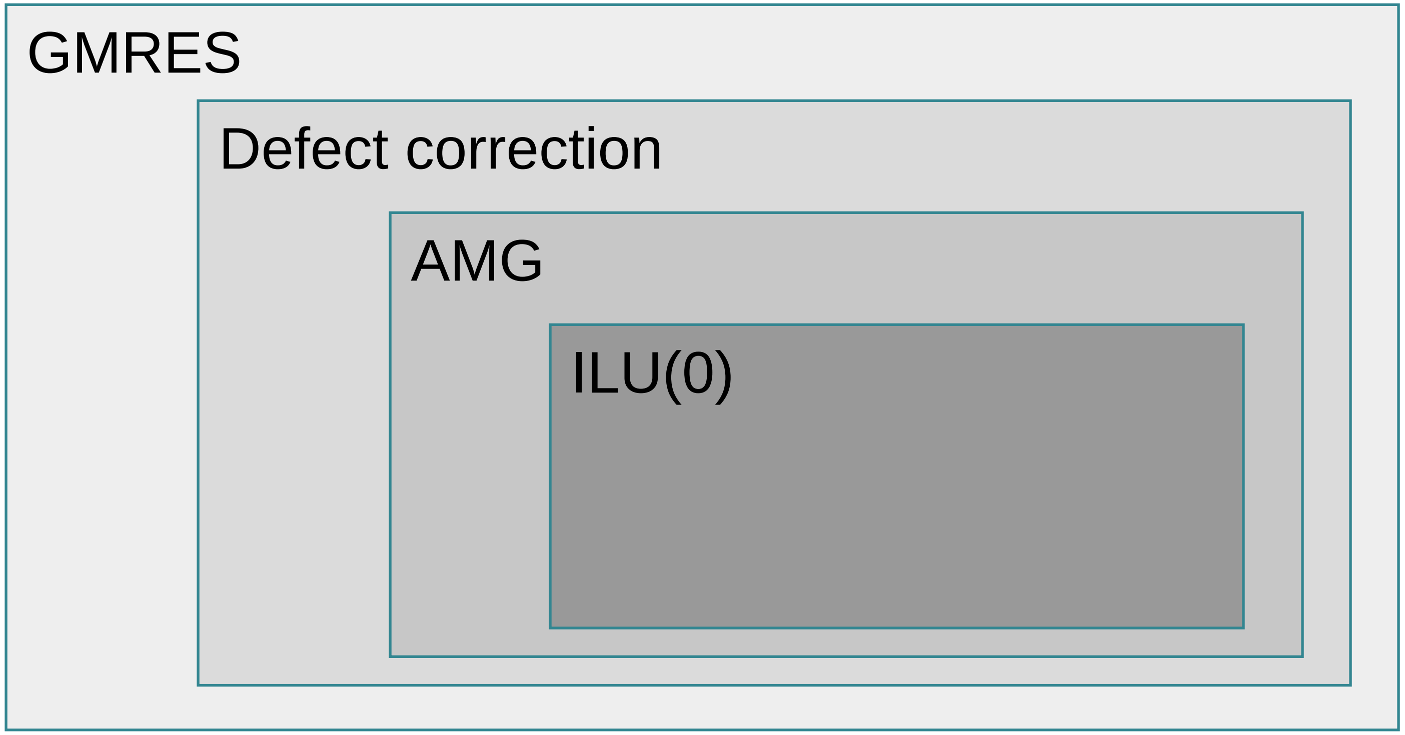


Mesh points involved in 1.- and 2.-order Jacobians

$$\boxed{A_1} x^{(n+1)} = \tilde{b}^{(n)} \quad n = 0, 1, \ldots$$

$$\tilde{b}^{(n)} = b - \left(Ax^{(n)} - A_1 x^{(n)}\right)$$

Scheme of the solution approach



Memory cost

Additional memory required for the approach

1. Storage of the 1.-order Jacobian (1/4 - 1/3 of 2.-order Jac.)
2. ILU(0) decomposition as a smoother(=1.)
3. AMG Hierarchy (~1/4 of 1.)
4. Krylov (depends on Krylov dimension)

Still less than ILU(0) for the 2.-order accurate Jacobian

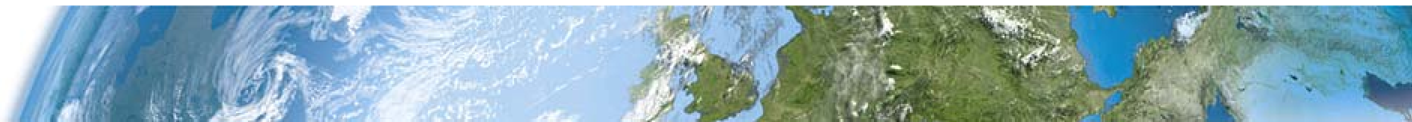
Therefore, the approach is cheaper than the cheapest ILU(0)-based PETSc solver

But significantly more expensive than TAU geometric multigrid

As a reference, for a 1 mln. points semi-structured NS mesh, 1 turb. eq.

Storage of **2.-order Jacobian** in CSR format for **~ 10 GByte**

Storage of **1.-order Jacobian** in CSR format **~ 2.5 GByte**



Aggregation-based AMG

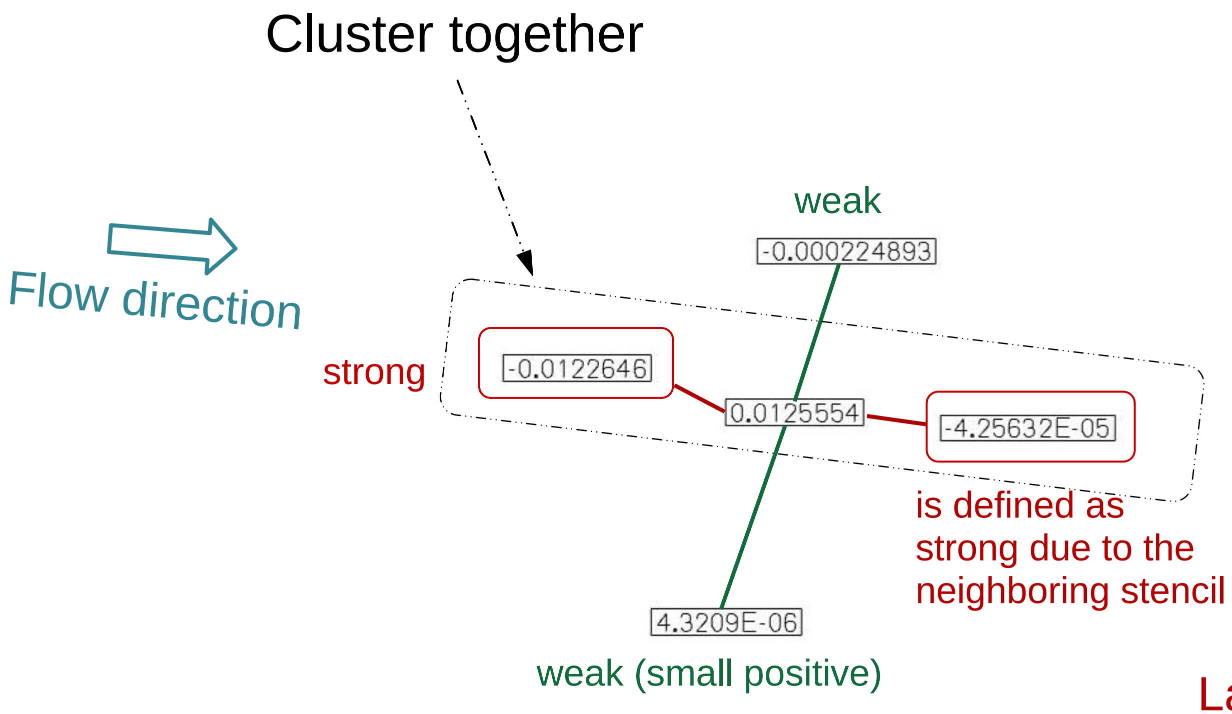
- Aggregation-based AMG is employed
  - fine “points” are grouped into “clusters”
  - piecewise-constant interpolation inside each cluster

- It is a favorable version of AMG for convection-dominated problems

- It is a “cheap” version of AMG: fast setup phase & sparse interpolation, restriction and coarse level operators

AMG coarsening (“clustering“)

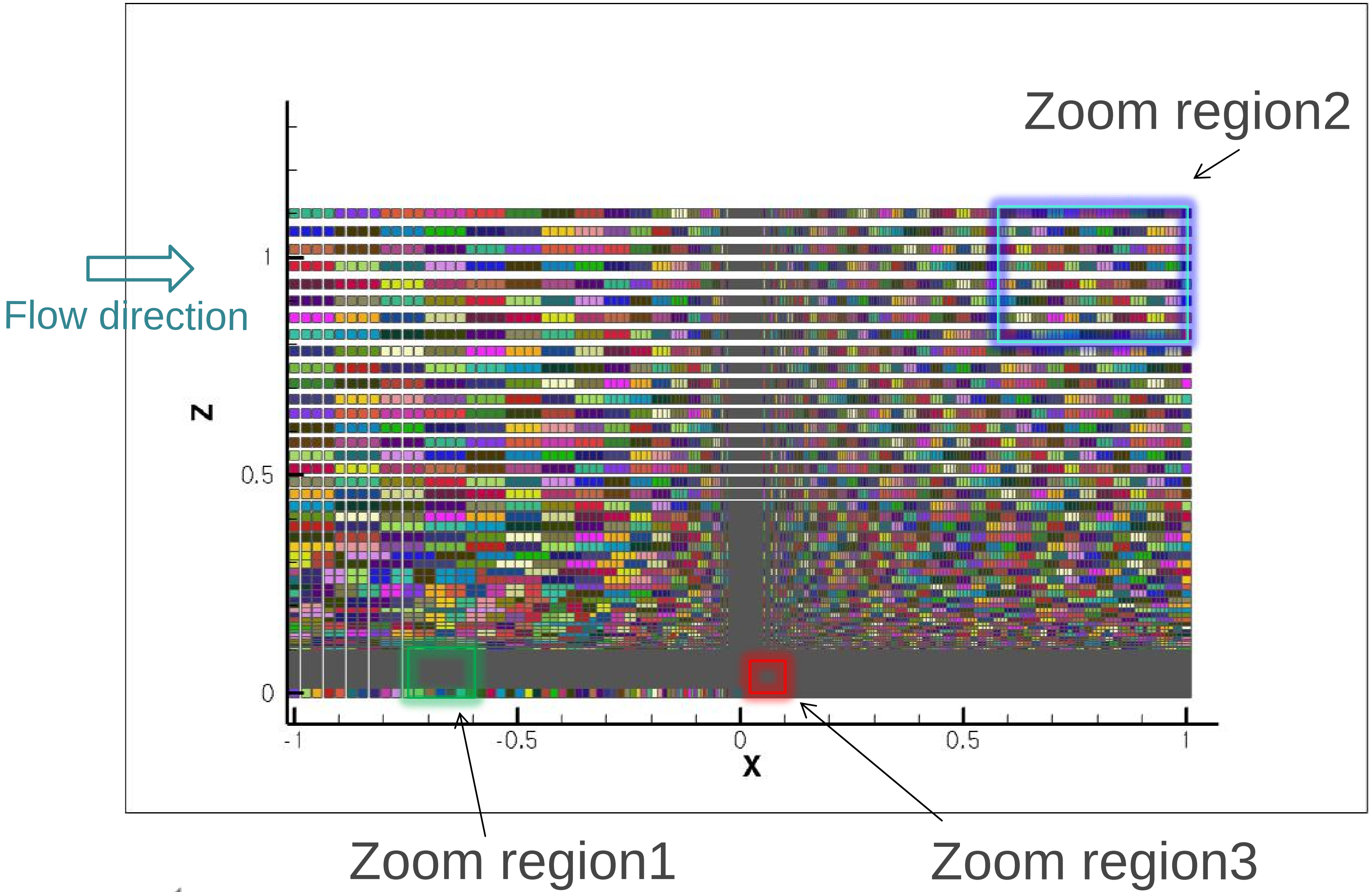
Indicate directions of strong couplings by evaluating the entries of 1.- order accurate Jacobian matrix: large negative couplings are defined as strong



Large negative couplings are defined as strong

density-density couplings in one grid point

AMG coarsening: Flat plate

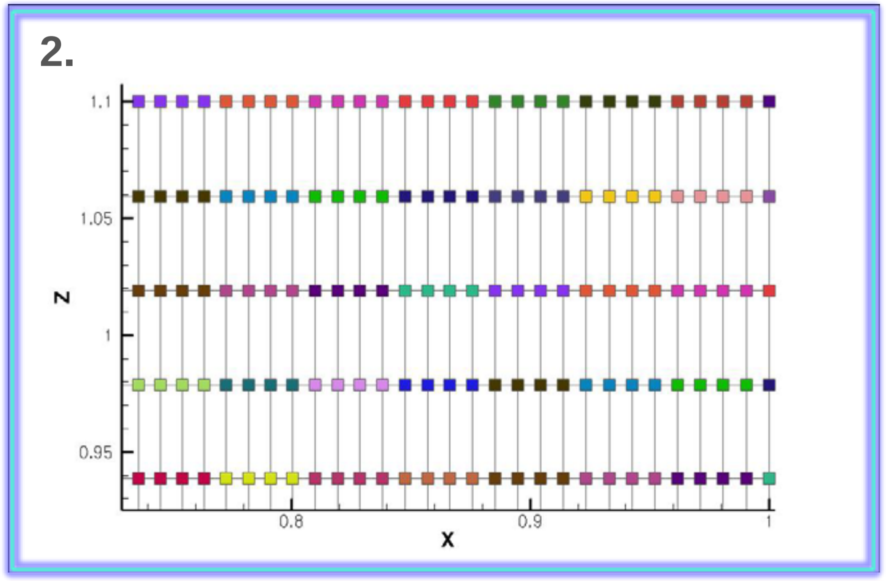


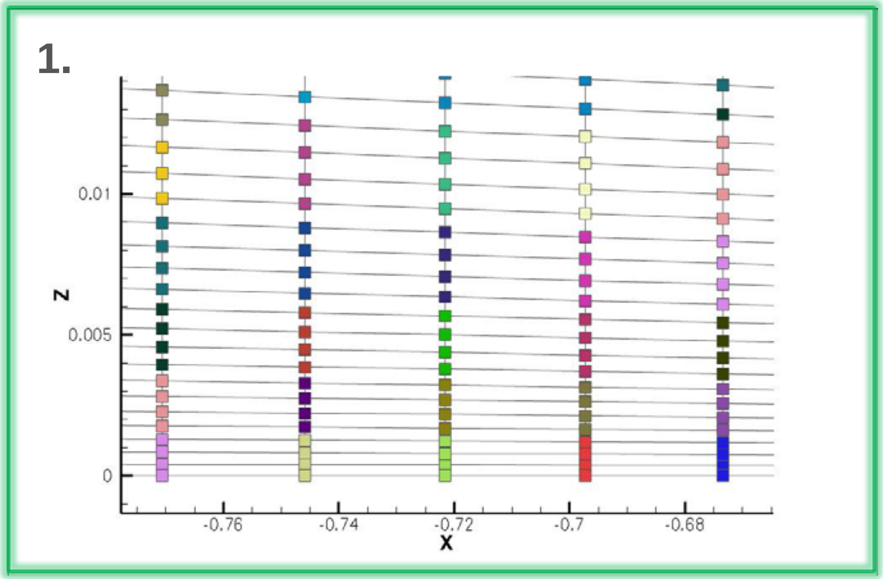
AMG coarsening and interpolation

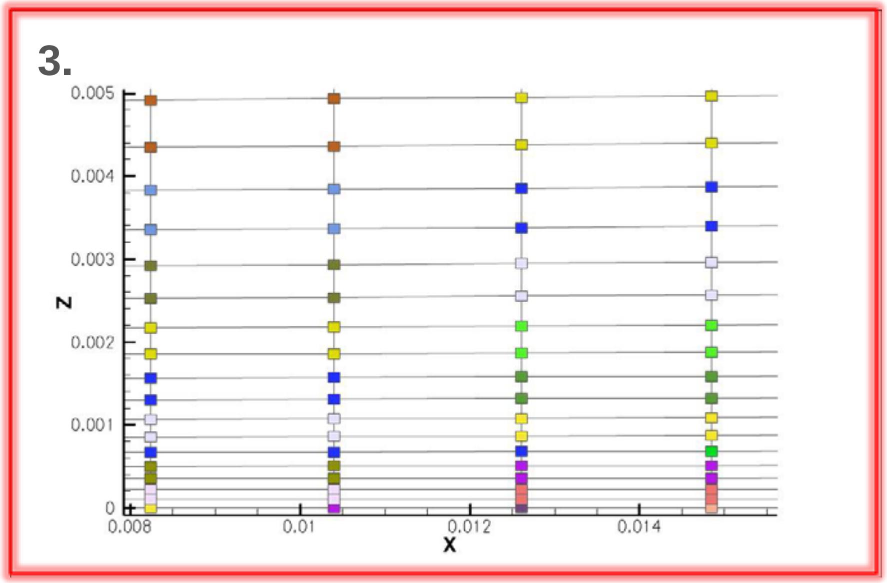
1. Clusters are built normal to the wall
(due to highly anisotropic cells)

2. Clusters are built in flow direction

3. Clusters are isotropic: effects of flow
and grid anisotr. compensate each other

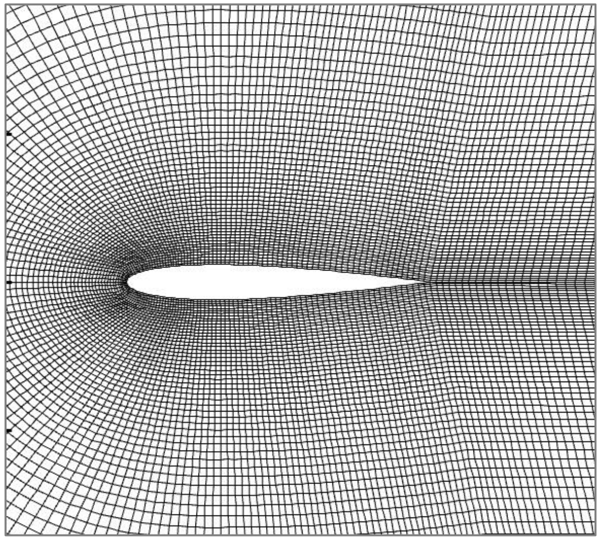






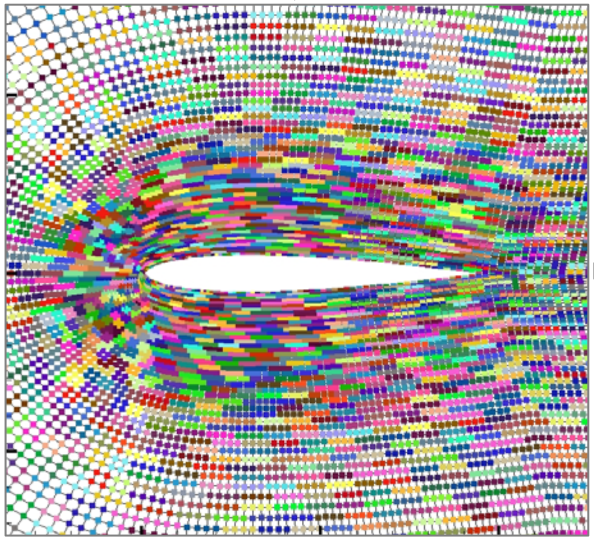
Remark: cells are more anisotropic than they are
shown in this zoom region (rescaled for visualization)

AMG Coarsening: NACA0012, laminar NS

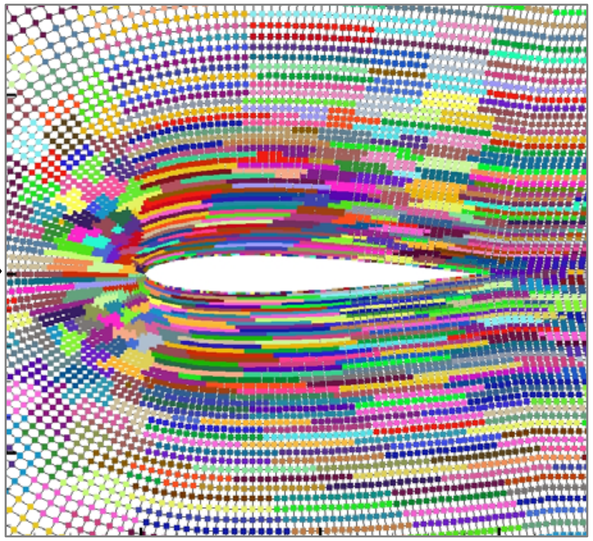


Fine level mesh

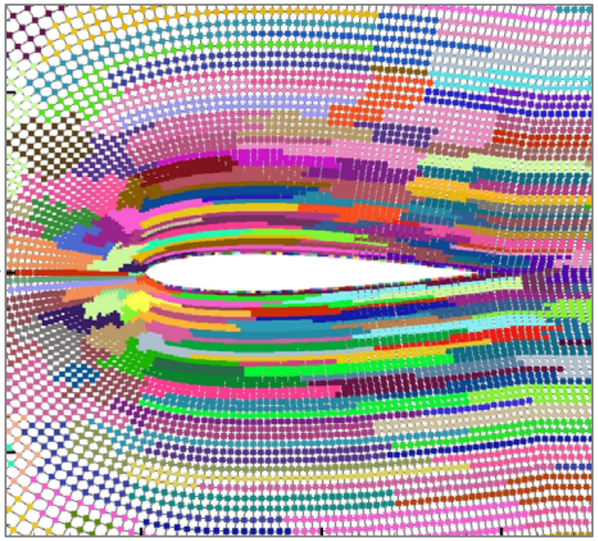
1. coarse level



2. coarse level



3. coarse level



AMG smoother

Initially, we used

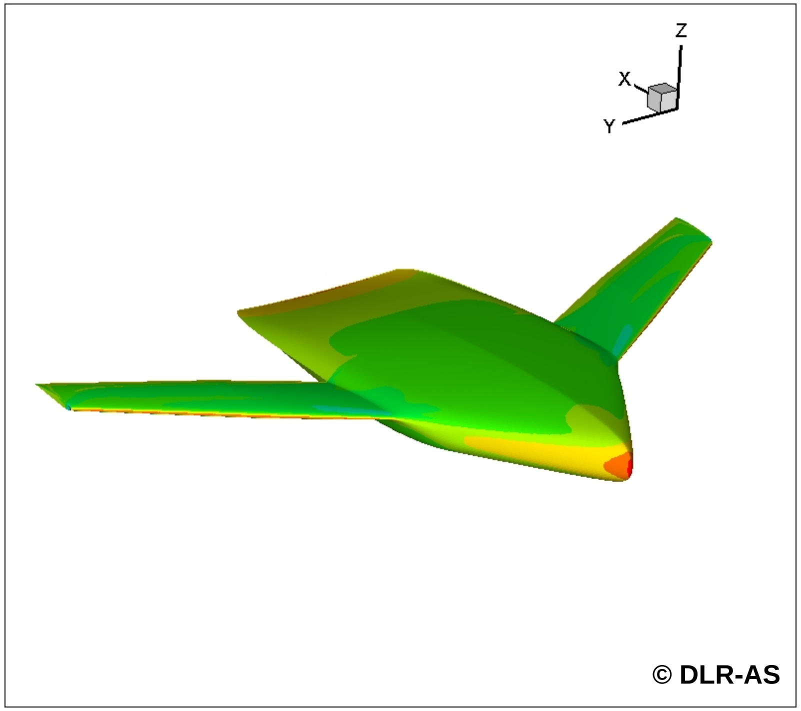
- Plain ILU(0) as a smoother (sequential)
- Local ILU(0) combined with Block-Jacobi (parallel)

A lot of benefit can be gained from using

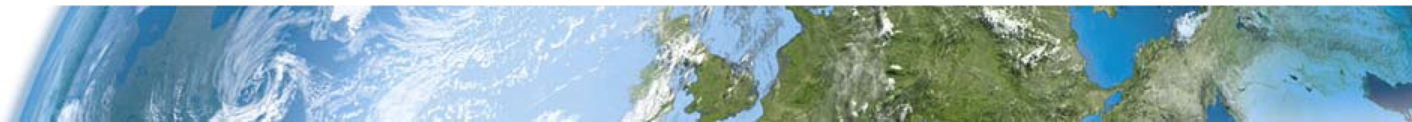
- Not the natural, but **reverse Cuthil-McKee ordering** for ILU(0)
- For parallel cases, instead of block-Jacobi, **Additive Schwarz** method (accounts for overlaps). The cost is not so high since ILU is done for 1.-order Jacobian.

Combining ILU(0) with RCM ordering and ASM results in a much more powerful smoother and in most cases speeds up convergence of the whole approach significantly

Numerical experiment: VELA, Euler flow

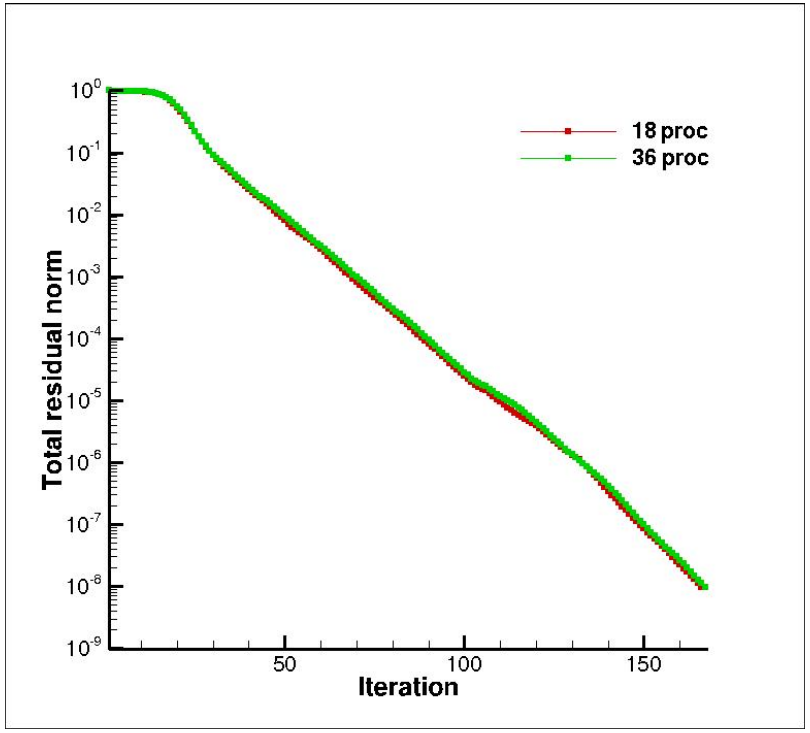


- 1.061.433 pts, unstructured
- alpha=1,8
- Mach=0,85



Numerical experiment: VELA, Euler flow

Convergence of the approach with
18 and 36 MPI processes



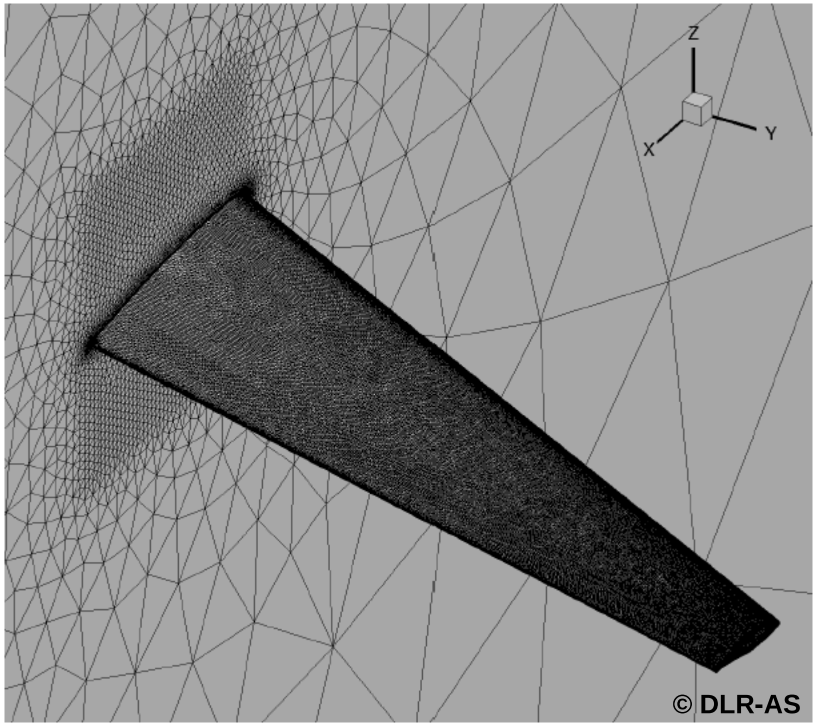
Run time on the CASE cluster:

9 min (18 proc) / 4 min (36 proc)

For this test case, solution with PETSc solvers was also successfull (GMRES +ASM+ILU(0)/ RCM-ordering)

Run time ~3 min (36 proc)

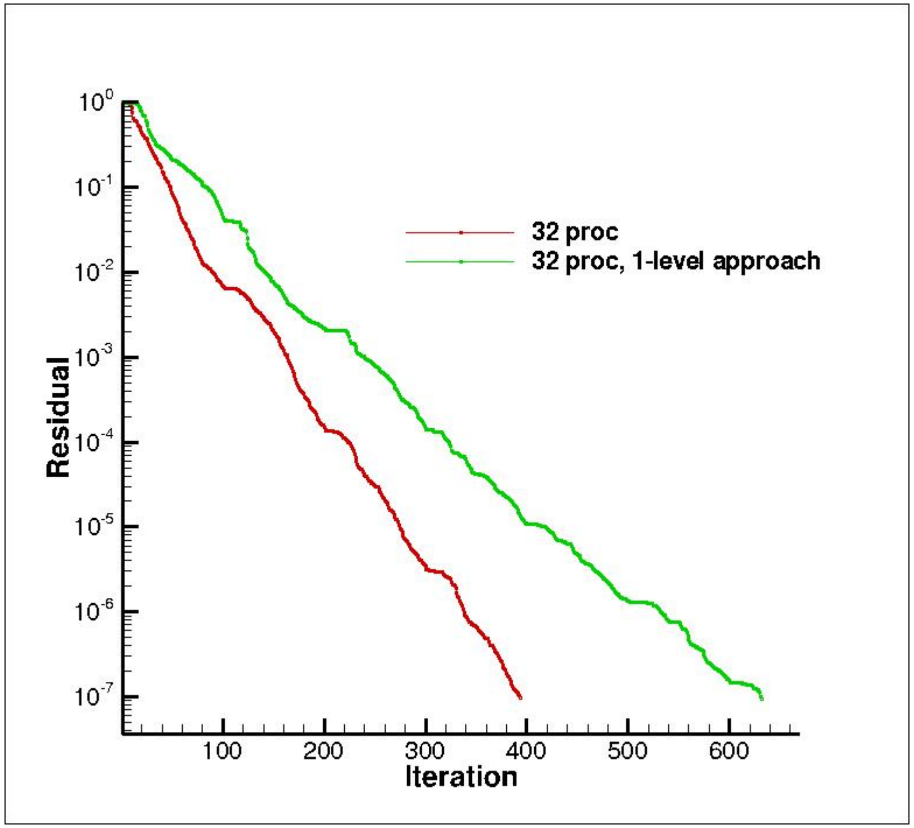
Numerical experiment: LANN wing



- 5.163.387 pts, semi-structured grid
- alpha=0.59
- Mach= 0.822, Re =5.43e6
- SAE turb. Model (1.-order upwind)

Numerical experiment: LANN wing

Convergence of the approach with
32 MPI processes



Run time on the CASE cluster:

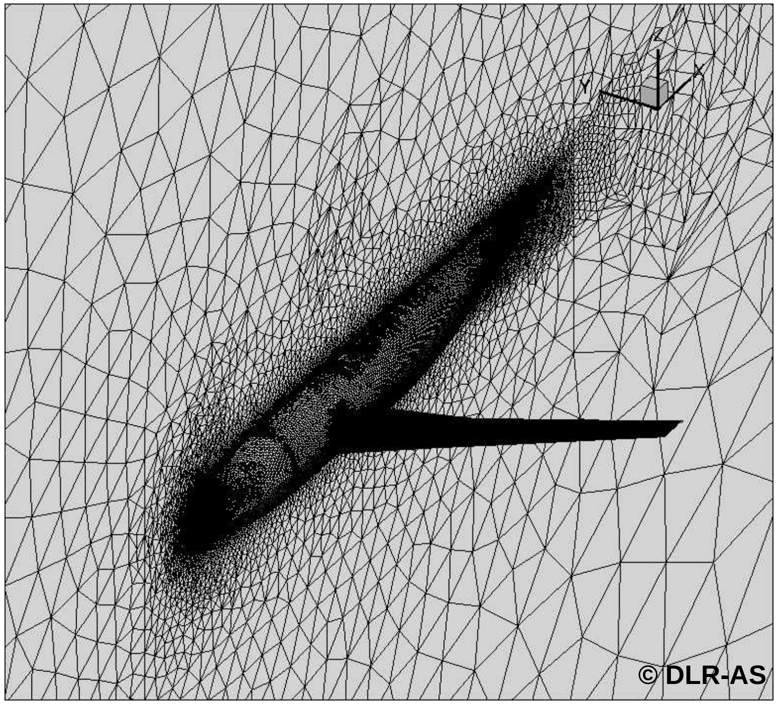
33 min (32 proc., DC with AMG)

28 min (32 proc., DC with 1-level solver)

For this test case, PETSc solvers as well as geometric multigrid failed to converge

DLR

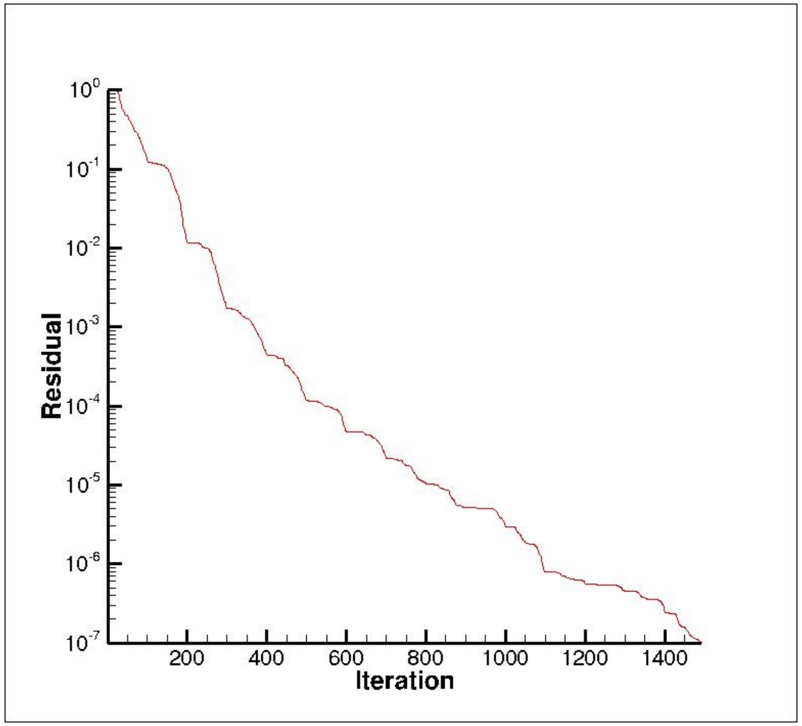
Numerical experiment: DLR F6



- 5.836.028 pts, semi-structured grid
- alpha=0.1
- Mach= 0.75, Re =3e6
- SAE turb. Model (1.-order upwind)

Numerical experiment: DLR F6

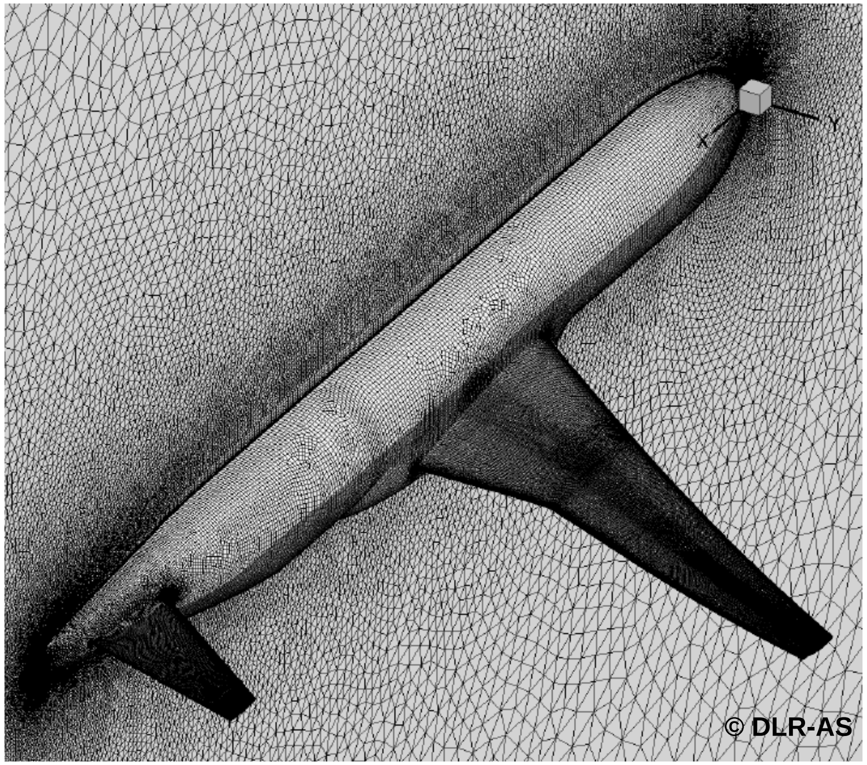
Convergence of the approach with
32 MPI processes



Run time on the CASE cluster:

1h 28 min (32 proc)

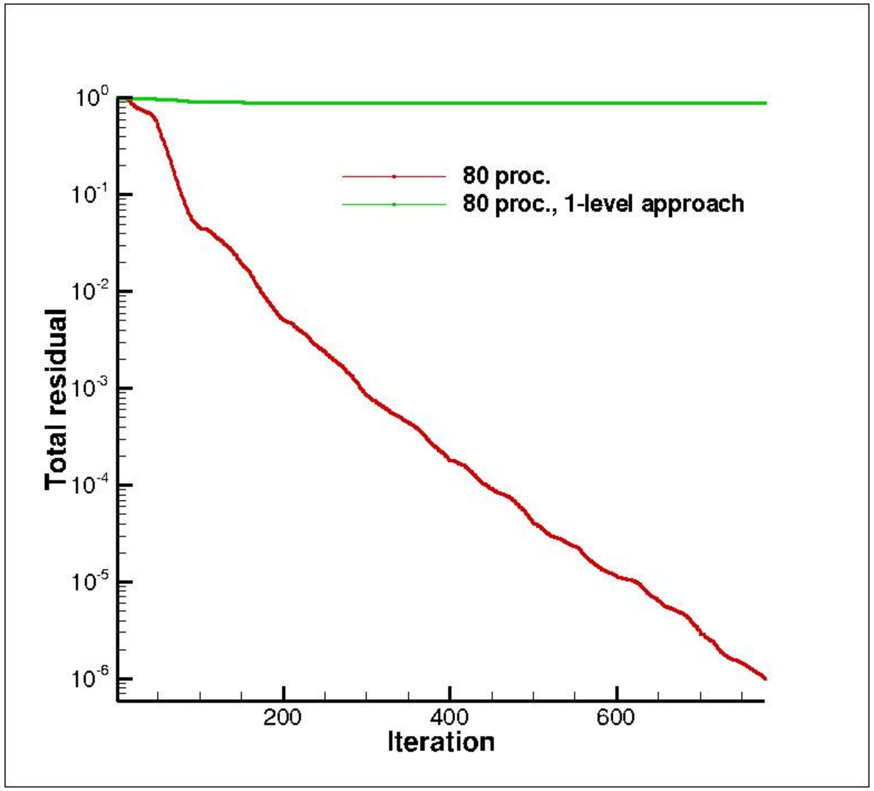
Numerical experiment: DPW4



- 11.696.804 pts, semi-structured
- alpha=2.29948
- Mach=0.85, Re =5.0e6
- SAO turb. Model (1.-order upwind)

Numerical experiment: DPW4

Convergence of the approach with
80 MPI processes



Run time on the CASE cluster:

1 h (80 proc.)

For this test case, PETSc solvers failed to converge

Geometric multigrid only converged with frozen turbulence (~7h, 192 proc., 10^-10)

Summary

3 different solution approaches for linear problems are available in TAU code

- Geometric multigrid
  - cheap but not very good convergence
- PETSc solvers
  - very fast if converges
  - very efficient in 2D
  - requires storage of full 2.-ord. Jacobian matrix
  - expensive, sometimes problematic for large cases
- Combination of Defect Correction and AMG
  - more expensive that gmg, cheaper than PETSc
  - requires storage of 1.-order Jacobian
  - rather good convergence
  - stable
  - more extensive testing is needed

Thank you for your attention

Fraunhofer SCAI

DLR