Implicit Methods and Globalization Strategies for the Robust Approximation of Solutions to the Reynolds Averaged Navier-Stokes equations

Stefan Langer

September 15th, 2016

Outline

- Introduction and Motivation – The RANS equations
- Solution algorithms – Multigrid smoothers
- Globalization strategies
- Numerical examples

Motivation

Goals of flighpath 2050:

1. In 2050 technologies and procedures available allow a 75% reduction in CO2 emissions per passenger kilometre to support the ATAG (Air Transport Action Group) target and a 90% reduction in NOx emissions.
2. The perceived noise emission of flying aircraft is reduced by 65%.

These are relative to the capabilities of typical new aircraft in 2000.

3. Overall, the European air transport system has less than one accident per ten million commercial aircraft flights.

→ The future aircraft is ecologically sensitive, low noise, and safe.

A key element, to design aircrafts ready for the future, is the accurate and efficient simulation of fluid flow coupled with other disciplines such as aerolastics and aeroacoustics.

Requirements of a CFD code

- Reliable tool in a process chain
- Interaction with other components (e.g. structure, mesh deformation, …)
- Accuracy, e.g. prediction of force coefficients up to a certain accuracy
- Evaluation and assessment of turbulence models
- …

Basic demand:

- Machine accurate solutions (on a given grid, that is a given resolution)
- Mesh converged solutions (in general hard to obtain, in particular in 3D)
- The code needs to run on regular basis without user interaction

How to prepare a CFD code for the future

- Identify the main building blocks
- Disaggregate the code into its building blocks
- Write routines which check the building blocks with respect to correctness
- Design interfaces such that the building blocks can be easily exchanged

Modular software design

Requirement: Identification of the main building blocks of a CFD code, to create algorithms prepared for demands in fully automatic process chains.

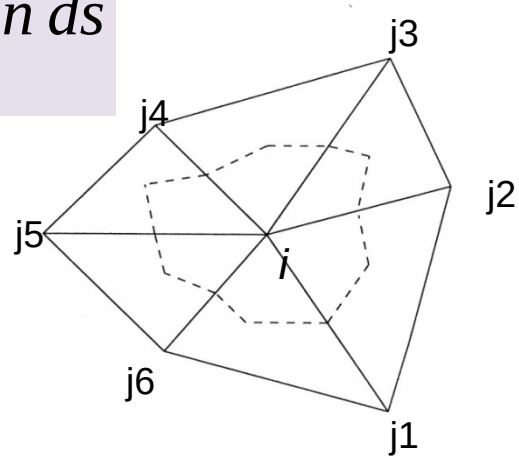
Compressible Navier-Stokes equations

$$\mathrm{W} := (\rho, \rho u, \rho E)$$ → Conservative variables

$$\frac{d}{dt}\int_{\Omega} \mathrm{W}\,\mathrm{dx} = -\int_{\partial\Omega} \left(\mathrm{F}_c(\mathrm{W}, \mathrm{grad\ W}) - \mathrm{F}_v(\mathrm{W}, \mathrm{grad\ W})\right)\cdot n\, ds$$

F_c : Convective flux

F_v : Viscous flux



Unstructured Finite volume discretization

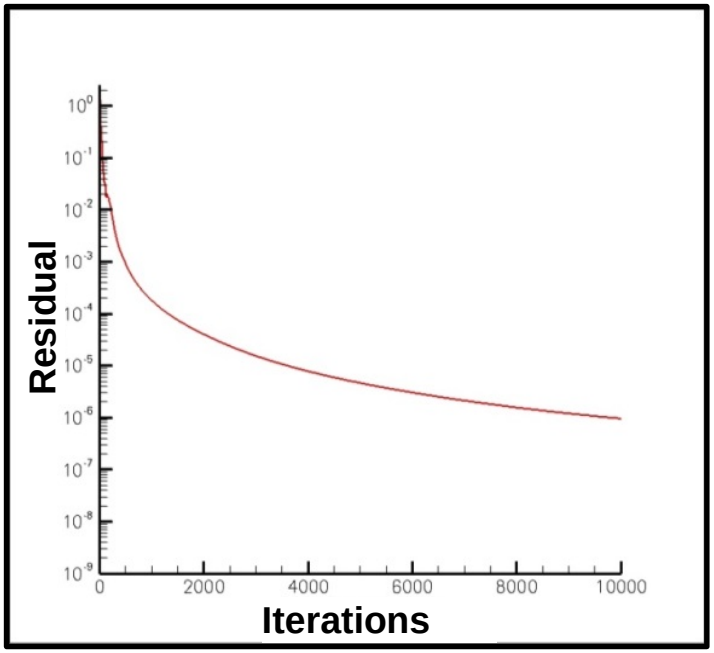
→ **Nonlinear operator equation:**

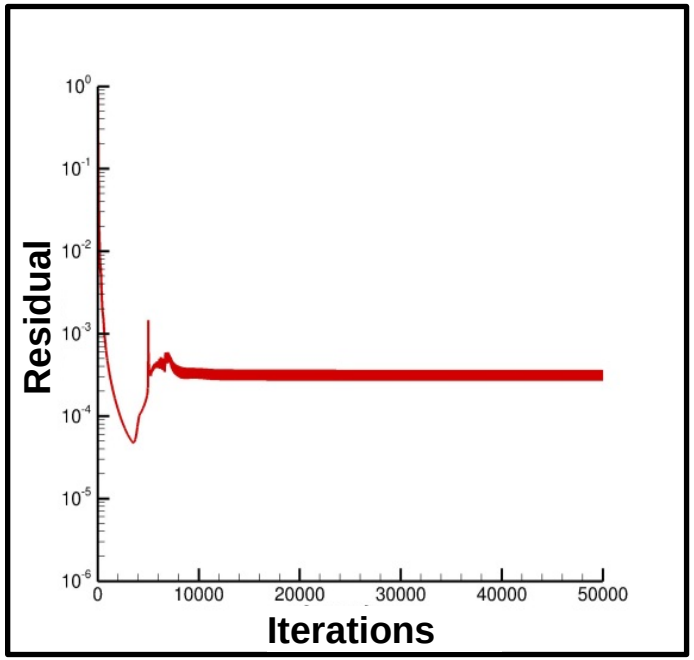
$$\frac{d\mathrm{W}}{dt} = -\mathrm{M}^{-1}\mathrm{R}(\mathrm{W}), \quad \mathrm{M} = \mathrm{diag}\left(\mathrm{vol}(\Omega_i)_{i=1,\ldots,N}\right)$$

Interested in steady state solution: $\mathrm{R}(\mathrm{W}) = 0$

Necessity for improvement of solution algorithms

Typical convergence behavior for high Reynolds number viscous flows

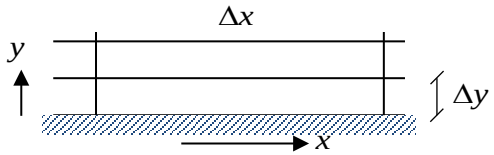




☞ **Convergence rate deteriorates significantly after inital phase**

☞ **No convergence, iteration stagnates**

→ Stiff set of equations



$$\frac{\Delta x}{\Delta y} > 10000$$

- **anisotropic cells to represent gradients in the boundary layer**
- **turbulent flow equations with source terms**

Outline

- Introduction and Motivation – The RANS equations
- Solution algorithms – Multigrid smoothers
- Globalization strategies
- Numerical examples

Structure of solution algorithm

Nonlinear multigrid: Solves $\mathbf{R}(\mathbf{W}) = 0$

Runge-Kutta smoother: Computes $\mathbf{W}^{(n+1)} = \mathbf{W}^{(n)} + \mathbf{P}^{-1}\mathbf{R}\left(\mathbf{W}^{(n)}\right)$

Krylov subspace method: Solves $\mathbf{Ph} = \mathbf{R}$

Linear Preconditioner: Solves $\mathbf{Prec}\ \mathbf{w} = \mathbf{Pv}$

Linear Multigrid
Requires: Sequence of meshes or algebraic multigrid, Smoother, Interpolation and Projection

Requires: Efficient iterative linear solver

Requires: Efficient preconditioner

Requires: Derivative, Linear solver

Requires: Sequence of meshes, Smoother, Interpolation and Projection operator

Structure of solution algorithm

Nonlinear multigrid: Solves $\mathbf{R}(\mathbf{W}) = 0$

Runge-Kutta smoother: Computes $\mathbf{W}^{(n+1)} = \mathbf{W}^{(n)} + \mathbf{P}^{-1}\mathbf{R}\left(\mathbf{W}^{(n)}\right)$

Krylov subspace method: Solves $\mathbf{P}\mathbf{h} = \mathbf{R}$

Linear Preconditioner: Solves $\mathbf{Prec}\ \mathbf{w} = \mathbf{P}\mathbf{v}$

Linear Multigrid
Requires: Sequence of meshes or algebraic multigrid, Smoother, Interpolation and Projection

Requires: Efficient iterative linear solver

Requires: Efficient preconditioner

Requires: Derivative, Linear solver

Requires: Sequence of meshes, Smoother, Interpolation and Projection operator

Multigrid smoother: Prec. Runge-Kutta method

$$\frac{d\mathrm{W}}{dt} = -\mathrm{M}^{-1}\mathrm{R}(\mathrm{W}), \quad \mathrm{M} = \mathrm{diag}\left(\mathrm{vol}(\Omega_i)_{i=1,\ldots,N}\right)$$

Apply preconditioned expl. Runge-Kutta method

$$\mathrm{W}^{(0)} := \mathrm{W}_n$$

$$\mathrm{W}^{(j)} := \mathrm{W}^{(0)} - \alpha_{j+1,j}\boxed{\mathrm{P}_j^{-1}}\mathrm{R}\left(\mathrm{W}^{(j-1)}\right), \quad j = 1,\ldots,s$$

$$\mathrm{W}_{n+1} := \mathrm{W}^{(s)}$$

$$\mathrm{P}_j := (\Delta t)^{-1}\,\mathrm{M} + \frac{\partial \mathrm{R}}{\partial \mathrm{W}}$$

to approximate W such that $\mathrm{R}(\mathrm{W}) \approx 0$

Multigrid smoother: Prec. Runge-Kutta method

$$\frac{d\mathrm{W}}{dt} = -\mathrm{M}^{-1}\mathrm{R}(\mathrm{W}), \quad \mathrm{M} = \mathrm{diag}\left(\mathrm{vol}(\Omega_i)_{i=1,\ldots,N}\right)$$

Apply preconditioned expl. Runge-Kutta method

$$\mathrm{W}^{(0)} := \mathrm{W}_n$$

$$\mathrm{W}^{(j)} := \mathrm{W}^{(0)} - \alpha_{j+1,j}\boxed{\left(\frac{CFL\Delta t_i}{\mathrm{vol}(\Omega_i)}\right)}\mathrm{R}\left(\mathrm{W}^{(j-1)}\right), \quad j = 1,\ldots,s$$

$$\mathrm{W}_{n+1} := \mathrm{W}^{(s)}$$

$$\mathrm{P}_j := (\Delta t)^{-1}\,\mathrm{M} + \cancel{\frac{\partial \mathrm{R}}{\partial \mathrm{W}}} = \left(\frac{CFL\Delta t_i}{\mathrm{vol}(\Omega_i)}\right)^{-1} = \left(\frac{\mathrm{vol}(\Omega_i)}{CFL\Delta t_i}\right)$$

Explicit Runge Kutta scheme with local time stepping

to approximate W such that $\mathrm{R}(\mathrm{W}) \approx 0$

→ **Requires inversion of a scalar value for each control volume:** $\left(\frac{\mathrm{vol}(\Omega_i)}{CFL\Delta t_i}\right)^{-1}$

Multigrid smoother: Prec. Runge-Kutta method

$$\frac{d\mathrm{W}}{dt} = -\mathrm{M}^{-1}\mathrm{R}(\mathrm{W}), \quad \mathrm{M} = \mathrm{diag}\left(\mathrm{vol}(\Omega_i)_{i=1,\ldots,N}\right)$$

Apply preconditioned expl. Runge-Kutta method

$$\mathrm{W}^{(0)} := \mathrm{W}_n$$

$$\mathrm{W}^{(j)} := \mathrm{W}^{(0)} - \alpha_{j+1,j}\boxed{\mathrm{P}_j^{-1}}\mathrm{R}\left(\mathrm{W}^{(j-1)}\right), \quad j = 1,\ldots,s$$

$$\mathrm{W}_{n+1} := \mathrm{W}^{(s)}$$

$$\mathrm{P}_j := (\Delta t)^{-1}\,\mathrm{M} + \frac{\partial \mathrm{R}}{\partial \mathrm{W}}$$

$$x = \mathrm{P}_j^{-1}\mathrm{R}\left(\mathrm{W}^{(j-1)}\right) \Leftrightarrow \mathrm{P}_j\, x = \mathrm{R}\left(\mathrm{W}^{(j-1)}\right)$$

to approximate W such that $\mathrm{R}(\mathrm{W}) \approx 0$

→ **Task: Need to approximate efficiently solution of** $\mathrm{P}x = \mathrm{R}(\mathrm{W})$

→ **Inversion of scalar value is replaced by solving a large scale linear system.**

The conncetion to Newton‘s method

Outer Loop: Multistage Runge-Kutta method → Choose s = 1, i.e. only one stage

$$\mathrm{W}^{(0)} := \mathrm{W}_n$$

$$\mathrm{W}^{(j)} := \mathrm{W}^{(0)} - \alpha_{j+1,j}\mathrm{P}_j^{-1}\mathrm{R}\left(\mathrm{W}^{(j-1)}\right), \quad j = 1,\ldots,s$$

$$\mathrm{W}_{n+1} := \mathrm{W}^{(s)}$$

$$\mathrm{P}_j := (\Delta t)^{-1}\mathrm{M} + \frac{\partial \mathrm{R}}{\partial \mathrm{W}} = \frac{\partial \mathrm{R}}{\partial \mathrm{W}}, \; CFL \to \infty$$

$$(\Delta t)^{-1} = \left(\frac{\mathrm{vol}(\Omega_i)}{CFL\Delta t_i}\right) \to 0, \quad CFL \to \infty$$

$$\mathrm{W}_{n+1} := \mathrm{W}_n - \left(\frac{\partial \mathrm{R}}{\partial \mathrm{W}}\right)^{-1}\mathrm{R}(\mathrm{W}_n)$$

→ The solution method is some kind of **generalization of Newton‘s method**

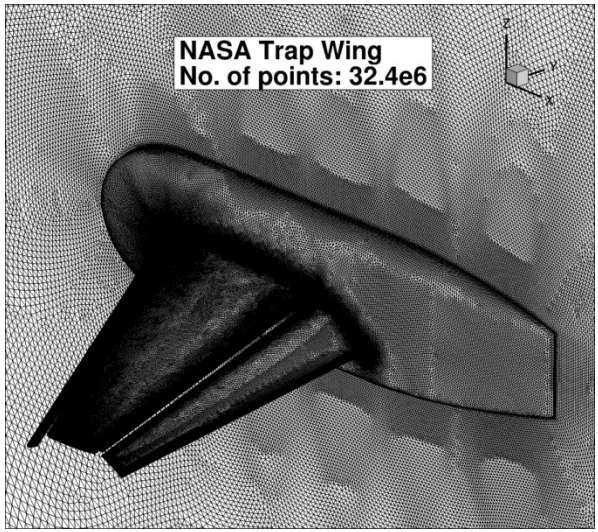
$$Px = R(W)$$

- Represents a large scale (in general more than 10^8 unknowns), ill-conditioned linear system

- It is not of interest to solve these linear systems, it is of interest to get a ***reasonable update*** for the outer nonlinear loop

- Krylov subspace methods are a natural choice for a matrix-free implementation

- A well suited **preconditioner** is required



Krylov subspace methods are in general only effective in combination with a well suited preconditioner!

Code design

- execute multigrid cycle (until convergence)
  - execute preconditioned expl. Runge-Kutta algorithm
    - evaluate residual R
    - evaluate derivative dR, P = Δt + dR
    - solve linear system (P, R)
      - apply preconditioned Krylov subspace method
        - construct a (further) preconditioner for the linear system Px = R
    - update flow variables W

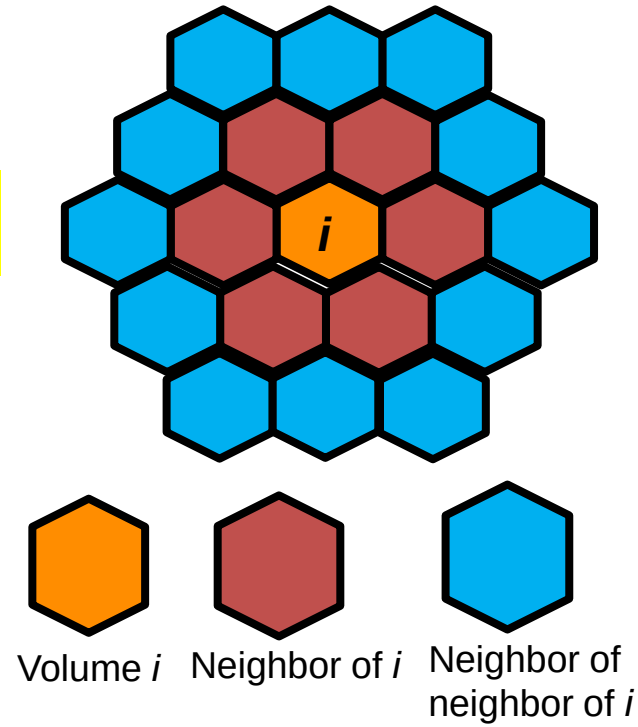
Construction of Preconditioner (for lin. System)

Idea: Base preconditioner upon next neighbor stencil

$$\mathrm{R}_i^{2nd}\left(\mathrm{W}_i, \mathrm{W}_{j\in N(i)}, \mathrm{W}_{k,k\in N(j)}\right) \approx \mathrm{R}_i^{1st}\left(\mathrm{W}_i, \mathrm{W}_{j,j\in N(i)}\right)$$

$$\Rightarrow \frac{\partial \mathrm{R}^{2nd}}{\partial \mathrm{W}} \approx \frac{\partial \mathrm{R}^{1st}}{\partial \mathrm{W}}$$

$$\Rightarrow \mathrm{Prec} = (\Delta t)^{-1}\mathrm{M} + \frac{\partial \mathrm{R}^{1st}}{\partial \mathrm{W}}$$



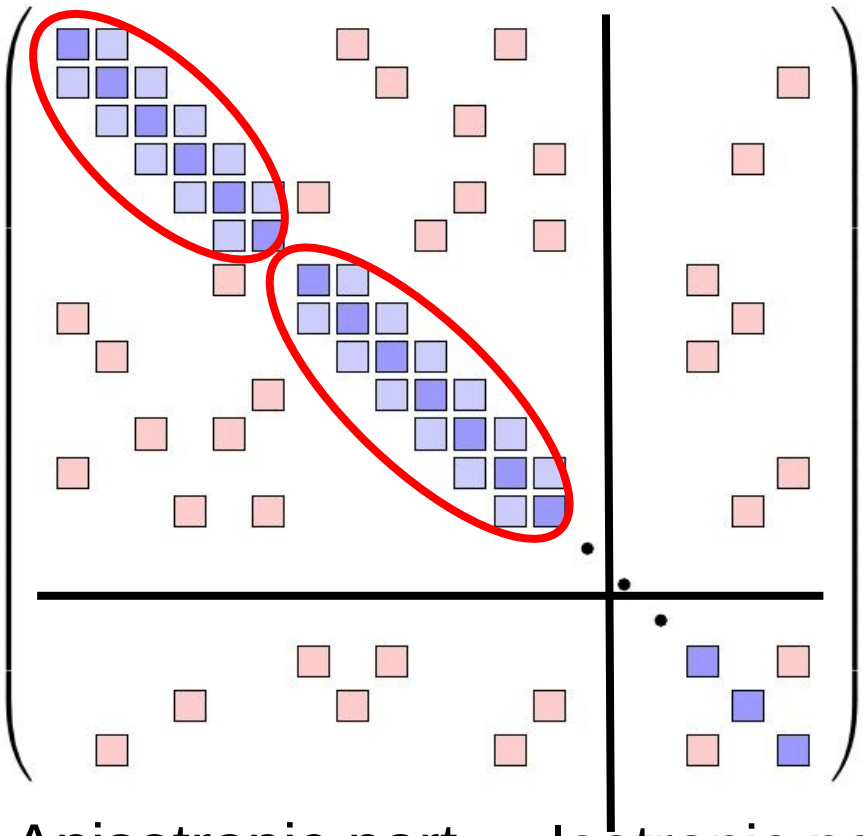
→ Required: Solution method for **Prec w = b**

Challenge: Find approximate solution of linear system

$$\text{Prec}\, w = b$$

where Prec is a block sparse matrix of dimension number of mesh points

$$\text{Prec} =$$

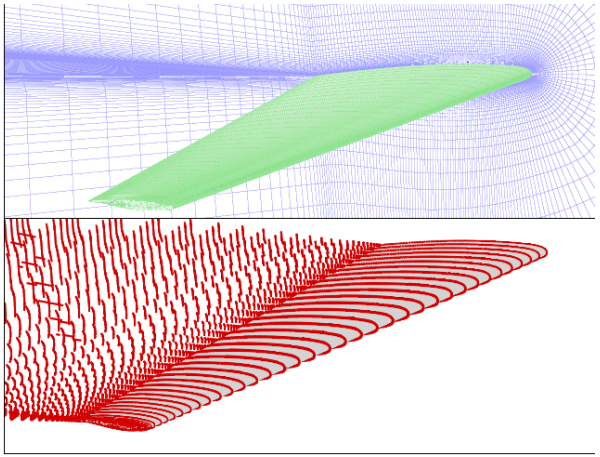


Anisotropic part Isotropic part

matrix blocks

- element diagonal
- line neighbor
- off-line off-diagonal

Point ordering according to lines yields (block) tridiagonal systems



DLR

Iterative solution methods for Prec w = z

Mathematical textbook methods for solution of linear systems, e.g

- (Block-) Jacobi method
- (Block-) Gauss-Seidel method
- Symmetric (Block-) Gauss-Seidel method

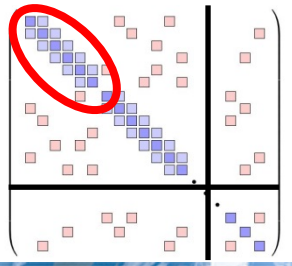
Methods have been extended:

→ Exploit ***directions of strongest coupling*** in iterative solution process

(Symmetric) Line (Jacobi) Gauss-Seidel method:

$$x_{L_i}^{(m+1)} = \mathrm{tridiag}\left(D_{L_i}\right)^{-1}\left(b_{L_i} - \sum_{j\in L_1,\ldots,L_{i-1},\, j\notin L_i} \mathrm{Prec}_{L_i j}\, x_j^{(m+1)} - \sum_{j\notin L_1,\ldots,L_{i-1},\, j\notin L_i} \mathrm{Prec}_{L_i j}\, x_j^{(m)}\right)$$

Algebraic representation and implementation of geometric data (Lines)



Code design

- execute multigrid cycle (until convergence)
  - execute preconditioned expl. Runge-Kutta algorithm
    - evaluate residual R
    - evaluate derivative dR, P = Δt + dR
    - solve linear system (P, R)
      - apply preconditioned Krylov subspace method
        - construct a (further) preconditioner for the linear system Px = R
        - solve Prec w = b to precondition Px = R (by Line symmetric Gauss-Seidel method)
    - update flow variables W

A historical view on solution methods in CFD

Two competitive views

Multigrid + Low cost smoother ←→ Newton's method (expensive smoother)

Low cost smoothers:

1. Expl. Runge-Kutta + local time stepping (Jameson)
2. Point implicit Runge-Kutta (Pierce, Giles, Moinier)
3. Line implicit Runge-Kutta (Mavriplis)
4. 1.st order approximate Jacobian (Swanson, Rossow, Yoon + Jameson (LU-SGS))

Preconditioned explicit Runge-Kutta smoother

All well known specific smoothers developed throughout the CFD literature are specifications of the general method shown here

The suggested methods just differ with respect to the ***approximation of the exact Jacobian*** and the **iterative solver**

Derivation of low cost smoothers

$$\mathrm{W}^{(0)} := \mathrm{W}_n$$
$$\mathrm{W}^{(j)} := \mathrm{W}^{(0)} - \alpha_{j+1,j} \mathrm{P}_j^{-1} \mathrm{R}\left(\mathrm{W}^{(j-1)}\right), \quad j = 1, \ldots, s$$
$$\mathrm{W}_{n+1} := \mathrm{W}^{(s)}$$

Smoothing step

- execute preconditioned expl. Runge-Kutta algorithm
  - evaluate residual R
  - evaluate derivative dR, P = Δt + dR
  - solve linear system (P, R)
    - apply preconditioned Krylov subspace method
      - construct a (further) preconditioner for the linear system Px = R
      - solve Prec w = b to precondition Px = R
  - update flow variables W

Derivation of low cost smoothers

$$W^{(0)} := W_n$$
$$W^{(j)} := W^{(0)} - \alpha_{j+1,j} P_j^{-1} R\left(W^{(j-1)}\right), \quad j = 1,\ldots,s$$
$$W_{n+1} := W^{(s)}$$

Simplifications:

1. Number of Krylov steps = 0

Smoothing step

- execute preconditioned expl. Runge-Kutta algorithm
  - evaluate residual R
  - evaluate derivative dR, P = Δt + dR
  - ~~solve linear system (P, R)~~
    - ~~apply preconditioned Krylov subspace method~~
      - construct a (further) preconditioner for the linear system Px = R
      - solve Prec w = b to precondition Px = R
  - update flow variables W

Derivation of low cost smoothers

$$W^{(0)} := W_n$$
$$W^{(j)} := W^{(0)} - \alpha_{j+1,j} P_j^{-1} R\left(W^{(j-1)}\right), \quad j = 1,\ldots,s$$
$$W_{n+1} := W^{(s)}$$

Simplifications:

1. Number of Krylov steps = 0

Smoothing step

- execute preconditioned expl. Runge-Kutta algorithm
  - evaluate residual R
  - evaluate derivative dR^{1st}, Prec = $\Delta t + dR^{1st}$
    - ~~construct a (further) preconditioner for the linear system Px = R~~
    - solve Prec w = b ~~to precondition Px = R~~
  - update flow variables W

Derivation of low cost smoothers

$$\mathrm{W}^{(0)} := \mathrm{W}_n$$
$$\mathrm{W}^{(j)} := \mathrm{W}^{(0)} - \alpha_{j+1,j}\,\mathrm{Prec}_j^{-1}\mathrm{R}\left(\mathrm{W}^{(j-1)}\right), \quad j = 1,\ldots,s$$
$$\mathrm{W}_{n+1} := \mathrm{W}^{(s)}$$

Simplifications:

1. Number of Krylov steps = 0

Smoothing step

- execute preconditioned expl. Runge-Kutta algorithm
  - evaluate residual R
  - evaluate derivative dR^{1st}, Prec = $\Delta t + dR^{1st}$
    - solve Prec w = b
  - update flow variables W

Preconditiong based on 1.st order approximate Jacobian
(Swanson, Rossow, Yoon + Jameson (LU-SGS))

Derivation of low cost smoothers

$$W^{(0)} := W_n$$
$$W^{(j)} := W^{(0)} - \alpha_{j+1,j}\,\mathrm{Prec}_j^{-1} R\left(W^{(j-1)}\right), \quad j = 1,\ldots,s$$
$$W_{n+1} := W^{(s)}$$

Simplifications:

1. Number of Krylov steps = 0
2. Simplify dR^{1st} entries by spectral radius

Smoothing step

- execute preconditioned expl. Runge-Kutta algorithm
  - evaluate residual R
  - evaluate derivative dR^{1st}, $\mathrm{Prec} = \Delta t + dR^{1st} \approx \Delta t + \varrho(dR^{1st})$
    - solve Prec w = b
  - update flow variables W

Solve with 1 symmetric Gauss-Seidel sweep

Yoon + Jameson (LU-SGS)

Derivation of low cost smoothers

$$\mathrm{W}^{(0)} := \mathrm{W}_n$$
$$\mathrm{W}^{(j)} := \mathrm{W}^{(0)} - \alpha_{j+1,j}\,\mathrm{Prec}_j^{-1}\mathrm{R}\left(\mathrm{W}^{(j-1)}\right), \quad j = 1,\ldots,s$$
$$\mathrm{W}_{n+1} := \mathrm{W}^{(s)}$$

Simplifications:

1. Number of Krylov steps = 0
2. Simplify dR^{1st} entries by spectral radius
3. Iterative solver: Line Jacobi truncated after one step

Smoothing step

- execute preconditioned expl. Runge-Kutta algorithm
  - evaluate residual R
  - evaluate derivative dR^{1st}, Prec = $\Delta t + dR^{1st}$
    - solve Prec w = b
  - update flow variables W

$$x^{(0)} = 0$$

$$x_{L_i}^{(m+1)} = \mathrm{tridiag}\left(D_{L_i}\right)^{-1}\left(b_{L_i} - \sum_{j \notin L_1,\ldots,L_{i-1},\, j \notin L_i} \mathrm{Prec}_{L_i j}\, x_j^{(m)}\right)$$

Derivation of low cost smoothers

$$\mathrm{W}^{(0)} := \mathrm{W}_n$$
$$\mathrm{W}^{(j)} := \mathrm{W}^{(0)} - \alpha_{j+1,j}\,\mathrm{Prec}_j^{-1}\mathrm{R}\left(\mathrm{W}^{(j-1)}\right), \quad j = 1,\ldots,s$$
$$\mathrm{W}_{n+1} := \mathrm{W}^{(s)}$$

Simplifications:

1. Number of Krylov steps = 0
2. Simplify dR^{1st} entries by spectral radius
3. Iterative solver: Line Jacobi truncated after one step

Smoothing step

- execute precondtioned expl. Runge-Kutta algorithm
  - evaluate residual R
  - evaluate derivative dR^{1st}, Prec = $\Delta t + dR^{1st}$
    - solve Prec w = b
  - update flow variables W

$$x^{(0)} = 0$$

$$x_{L_i}^{(1)} = \mathrm{tridiag}\left(D_{L_i}\right)^{-1} b_{L_i}$$

Derivation of low cost smoothers

$$W^{(0)} := W_n$$

$$W_{L_k}^{(j)} := W_{L_k}^{(0)} - \alpha_{j+1,j}\,\mathrm{tridiag}\left(D_{L_k}\right)^{-1} R\left(W^{(j-1)}\right)$$

$$W_{n+1} := W^{(s)}$$

Smoothing step

- execute precondtioned expl. Runge-Kutta algorithm
  - evaluate residual R
  - evaluate derivative dR^{1st}, Prec = $\Delta t + dR^{1st}$
    - solve $w = \mathrm{tridiag}(D)^{-1} b$
  - update flow variables W

Simplifications:

1. Number of Krylov steps = 0
2. Simplify dR^{1st} entries by spectral radius
3. Iterative solver: Line Jacobi truncated after one step

$$x^{(0)} = 0$$

$$x_{L_i}^{(1)} = \mathrm{tridiag}\left(D_{L_i}\right)^{-1} b_{L_i}$$

Line implicit Runge-Kutta (Mavriplis)

Derivation of low cost smoothers

$$W^{(0)} := W_n$$

$$W_k^{(j)} := W_k^{(0)} - \alpha_{j+1,j} (D_k)^{-1} R\left(W^{(j-1)}\right)$$

$$W_{n+1} := W^{(s)}$$

Smoothing step

Simplifications:

1. Number of Krylov steps = 0
2. Simplify dR^{1st} entries by spectral radius
3. Iterative solver: Line Jacobi truncated after one step
4. Neglect lines, i.e. perform only Jacobi iteration

- execute precondtioned expl. Runge-Kutta algorithm
  - evaluate residual R
  - evaluate derivative dR^{1st}, Prec = $\Delta t + dR^{1st}$
    - solve $w = (D)^{-1} b$
  - update flow variables W

$$x^{(0)} = 0$$

$$x_i^{(1)} = (D_i)^{-1} b_i$$

Point implicit Runge-Kutta (Giles, Moinier)

Derivation of low cost smoothers

$$W^{(0)} := W_n$$

$$W_k^{(j)} := W_k^{(0)} - \alpha_{j+1,j} \frac{\Delta t_k}{\mathrm{vol}(\Omega_k)} R\left(W^{(j-1)}\right)$$

$$W_{n+1} := W^{(s)}$$

Smoothing step

Simplifications:

1. Number of Krylov steps = 0
2. Simplify dR^{1st} entries by spectral radius
3. Iterative solver: Line Jacobi truncated after one step
4. Neglect lines, i.e. perform only Jacobi iteration
5. Approximate diagonal terms of Jacobian by spectral radius

- execute precondtioned expl. Runge-Kutta algorithm
  - evaluate residual R
  - evaluate derivative dR^{1st}, Prec = $\Delta t + dR^{1st}$
    - solve $w = (\Delta t / \mathrm{vol})^{-1} b$
  - update flow variables W

Explicit Runge-Kutta (Jameson)

→ Full hierarchy of solution methods

$$x_i^{(1)} = \left(\rho(D_i)\right)^{-1} b_i = \frac{\Delta t_i}{\mathrm{vol}(\Omega)_i} b_i$$

Main building blocks of a CFD code

- Data structure for (block) sparse matrices → $\frac{\partial R}{\partial W}$
- Data structure for (block) vectors → $R(W)$
- Algorithms acting on these data structures

→ Link to a suited, efficient **LINEAR ALGEBRA PACKAGE**

Outline

- Introduction and Motivation – The RANS equations
- Solution algorithms – Multigrid smoothers
- Globalization strategies
- Numerical examples

What is a globalization strategy?

Consider Newton‘s method $$W^{n+1} = W^n - \left[\frac{\partial R}{\partial W}\right]^{-1} R\left(W^n\right)$$

A globalization strategy is the try to construct an algorithm which

1. **preserves the nice properties of Newton‘s method**
2. **circumvents its shortcomings**

Why do we need it?

Newton‘s method converges only

1. **under certain smoothness assumptions**
2. **if the initial guess is in a neighborhood of the root**
3. **…**

Analysis of schemes: Globalization strategies

Parameter settings allow for several possible smoothing techniques:

- Number of Runge-Kutta stages
- Number of Gauss-Seidel sweeps
- Number of Krylov subspace steps
- Approximation of Jacobian
- …..

Newton‘s method

↕

Explicit Runge-Kutta

How to choose a robust and efficient method?

→ Development of an analysis tool to give some guideline

Evaluation of smoother: Consider linearized problem

Nonlinear Problem:

$$\frac{dW}{dt} = -M^{-1}R(W)$$

$$W^{(0)} := W_n$$
$$W^{(j)} := W^{(0)} - \alpha_{j+1,j} P_j^{-1,\mathrm{app}} R\left(W^{(j-1)}\right)$$
$$W_{n+1} := W^{(s)}$$

Linearized Problem

$$\frac{dW}{dt} \approx -M^{-1}\left(\underbrace{R(W^*)}_{=0;\ \text{Steady state}} + \frac{\partial R}{\partial W}[W^*]\Delta W \right)$$

$$W^{(0)} := W_n$$

$$W^{(j)} := W^{(0)} - \alpha_{j+1,j} P_j^{-1,\mathrm{app}} \frac{\partial R}{\partial W} W^{(j-1)}$$

$$W_{n+1} := W^{(s)} \Leftrightarrow W^{(n+1)} = q_s\left(P_j^{-1,\mathrm{app}} \frac{\partial R}{\partial W} \right) W^{(j-1)}$$

$$q_s(z) = 1 + \sum_{j=1}^{s} \beta_j z^j$$

Convergence $\Leftrightarrow \rho\left(q_s\left(P_j^{-1,\mathrm{app}} \frac{\partial R}{\partial W} \right) \right) < 1$

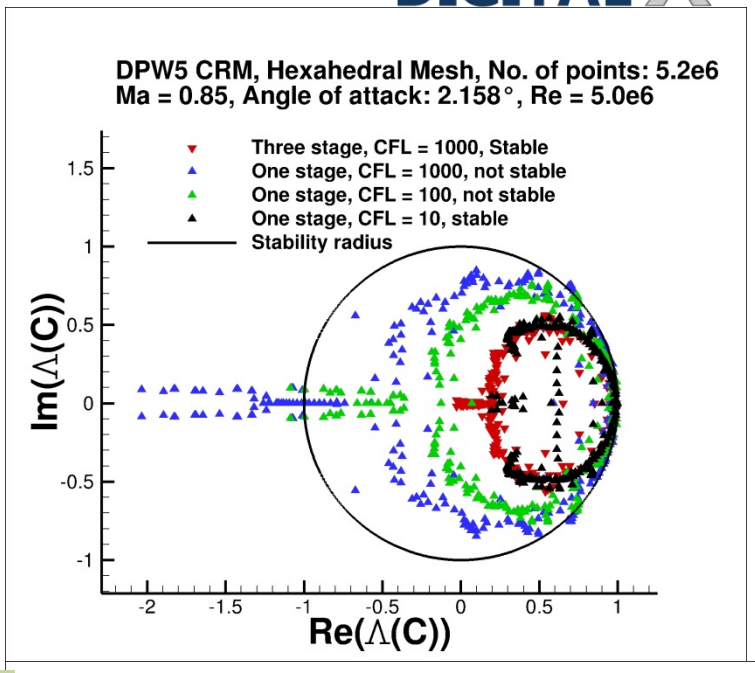
Analysis of schemes: Impact of CFL number

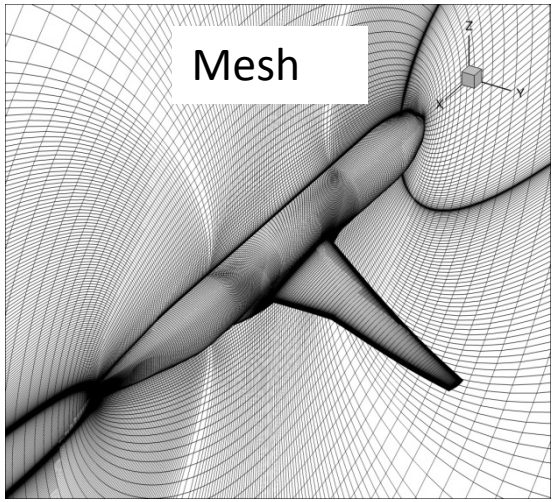
Analysis for mesh with 5.2e6 points:

Investigation of <u>number of stages for</u>

- **symmetric Line Gauss-Seidel**
- **different CFL numbers**

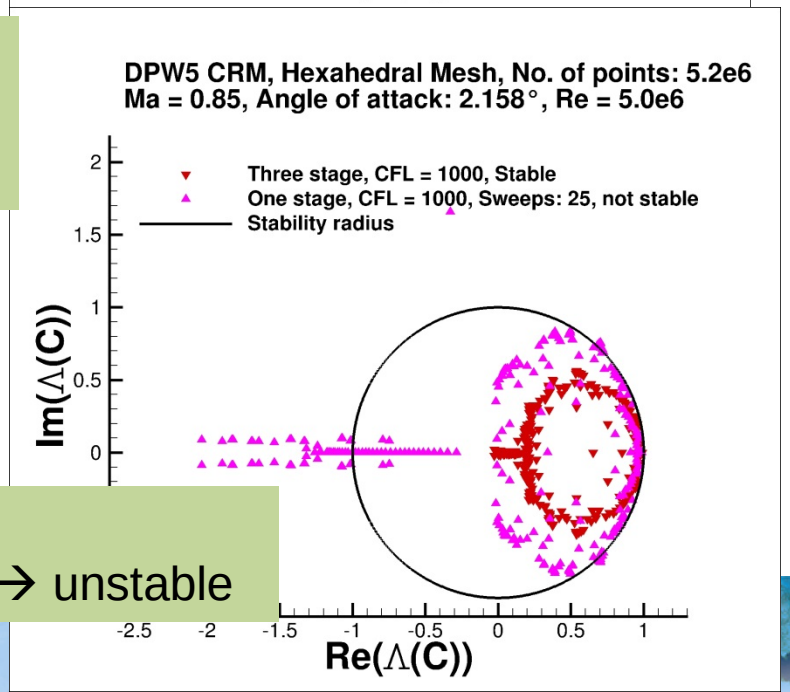
One stage: CFL = 1000 → unstable
One stage: CFL = 100 → unstable
One stage: CFL = 10 → stable
Three stage: CFL = 1000 → stable





Significant reduction of CFL necessary for one stage schemes

Additional effort does not pay of



Sweeps: 25
One stage: CFL = 1000 → unstable

Outline

- Introduction and Motivation – The RANS equations
- Solution algorithms – Multigrid smoothers
- Globalization strategies
- Numerical examples
  - Spalart-Allmaras (neg.)
  - Wilcox (kω)

Spalart-Allmaras 1992 ←→ Spalart Allmaras 2012

1992 model:

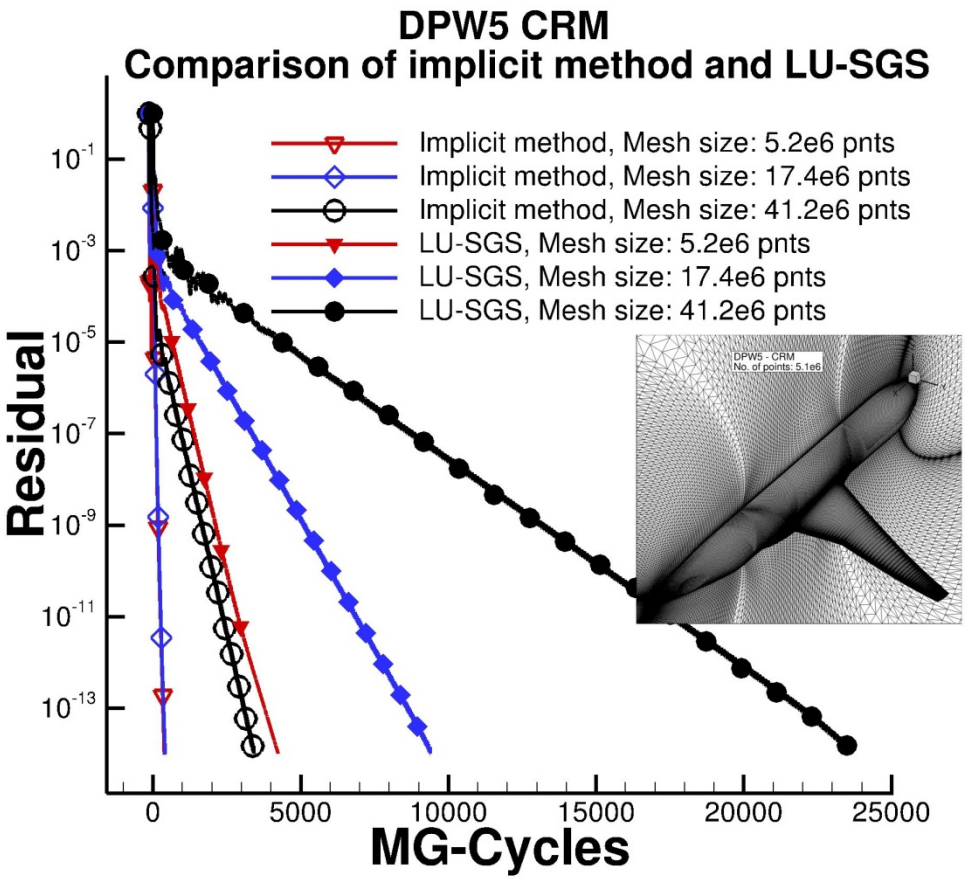
- transported variable can become negative such that iteration diverges
- choice of farfield values not clarified
- ….

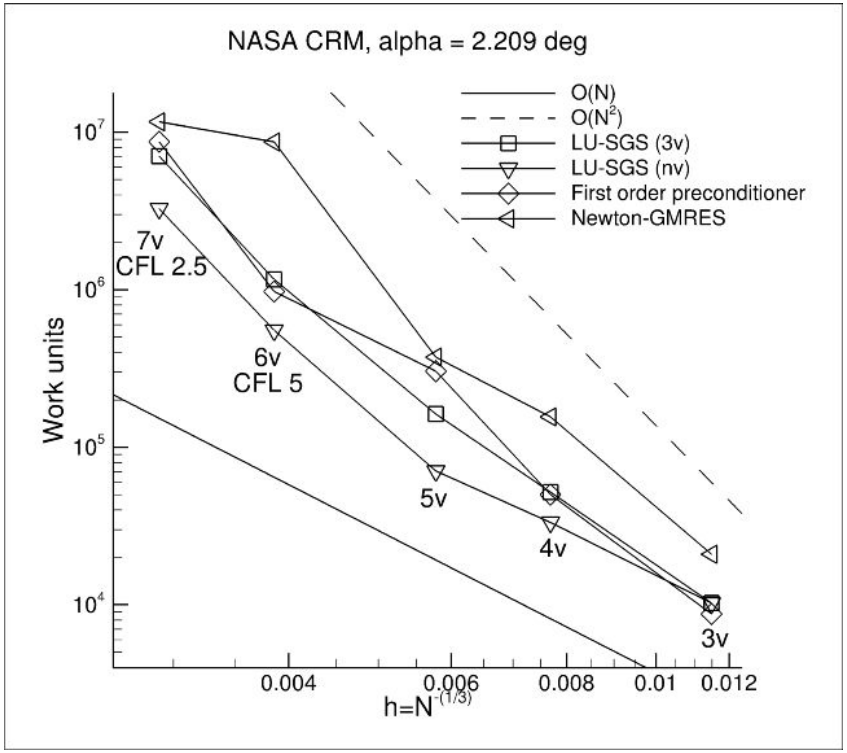
2012 modification of original model:

- Allows for small negative values of transported variable
- clarification of choice of farfield values
- description and recommendations of implementation of several terms and details

→ 2012 version has been succesfully implemented into the DLR TAU-Code

CRM of 5th AIAA drag prediction workshop

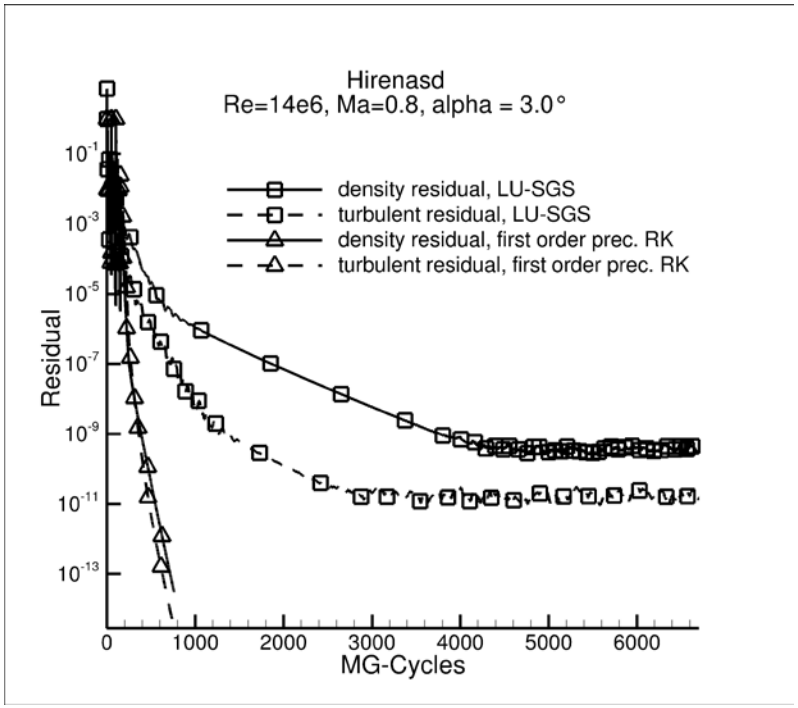


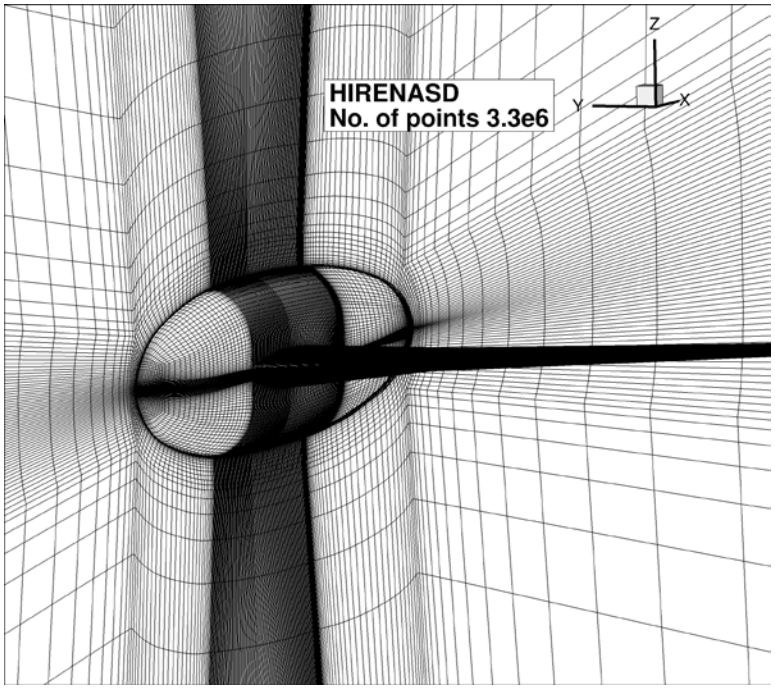


Implicit methods have comparable complexity to standard LU-SGS method by improved robustness

Numerical example: HIRENASD

- High Reynolds number Aero-Structural Dynamics wind tunnel configuration
- Ma = 0.8
- α = 3.0°
- Re = 14e6
- Pure hexahedral mesh: 3.3e6 points



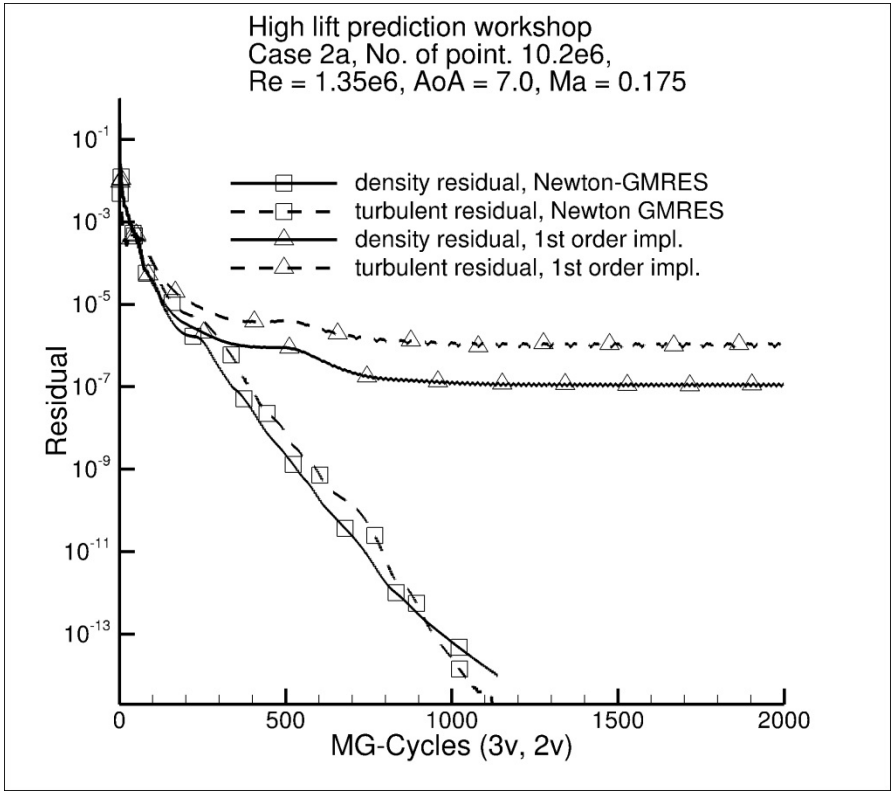


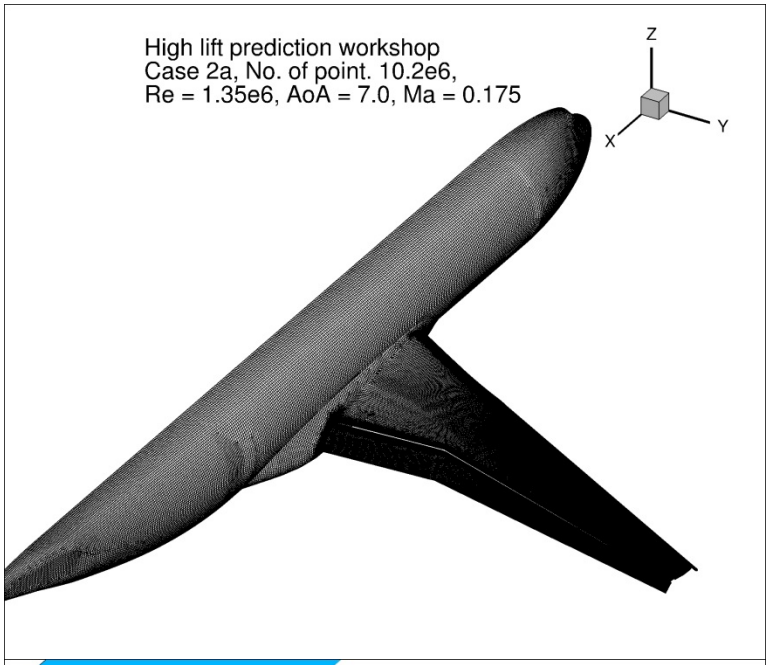
→ Implicit method converges,
→ LU-SGS method stalls

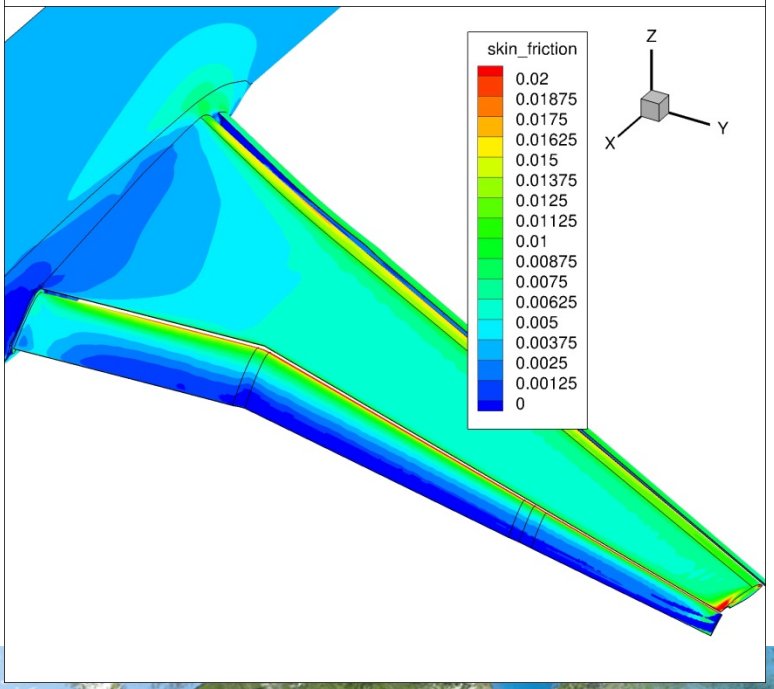
Configuration from second high-lift prediction workshop: Case 2a

Necessity of Newton-kind algorithms

Turbulence model: SA-Neg
Ma = 0.175, Re = 1.35e6, AoA = 7.0°

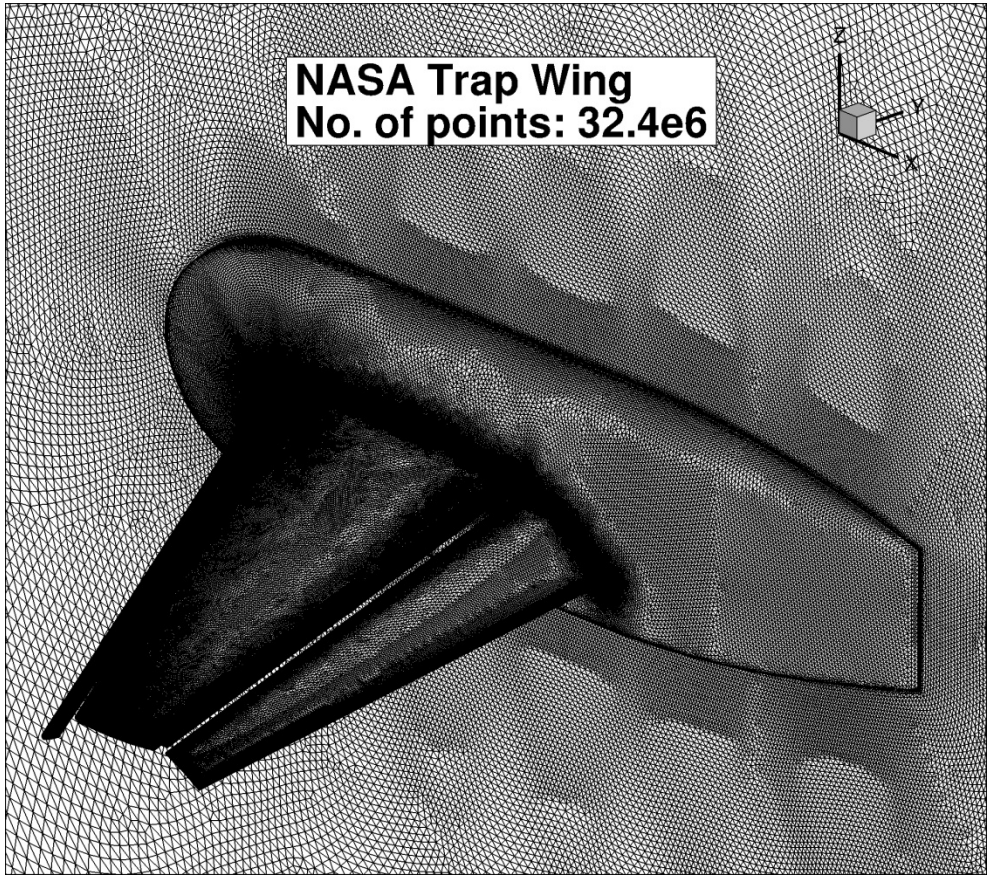


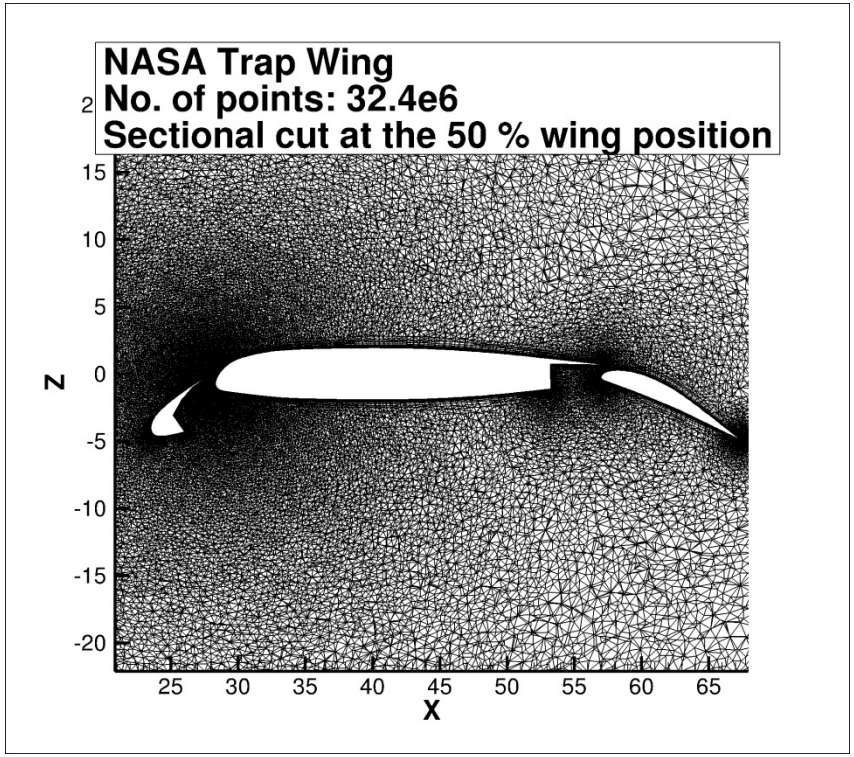




NASA Trap Wing, Ma = 0.2, Re = 4.3e6

Unstructured mesh results for
AOA = 13°, 28°, 32°, 34°, 37°





- Coarse Mesh: 3.7e6 NDOF
- Medium Mesh: 11.0e6 NDOF
- Fine Mesh: 32.4e6 NDOF

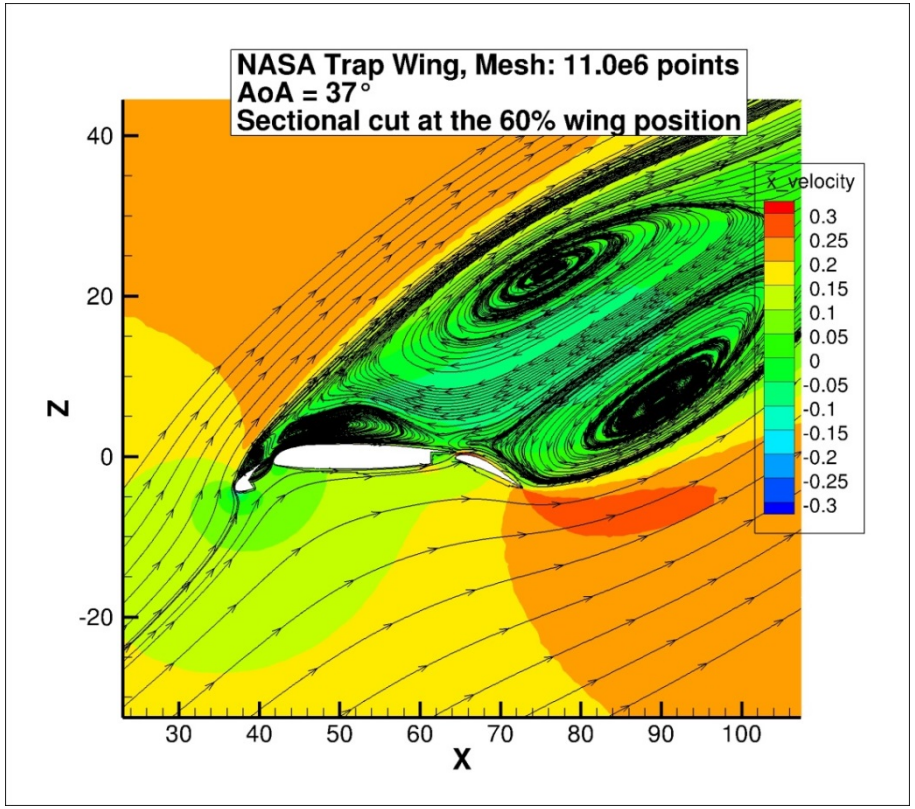
VGRID Meshes used at High Lift Prediction Workshop 1

NASA Trap Wing, Ma = 0.2, Re = 4.3e6

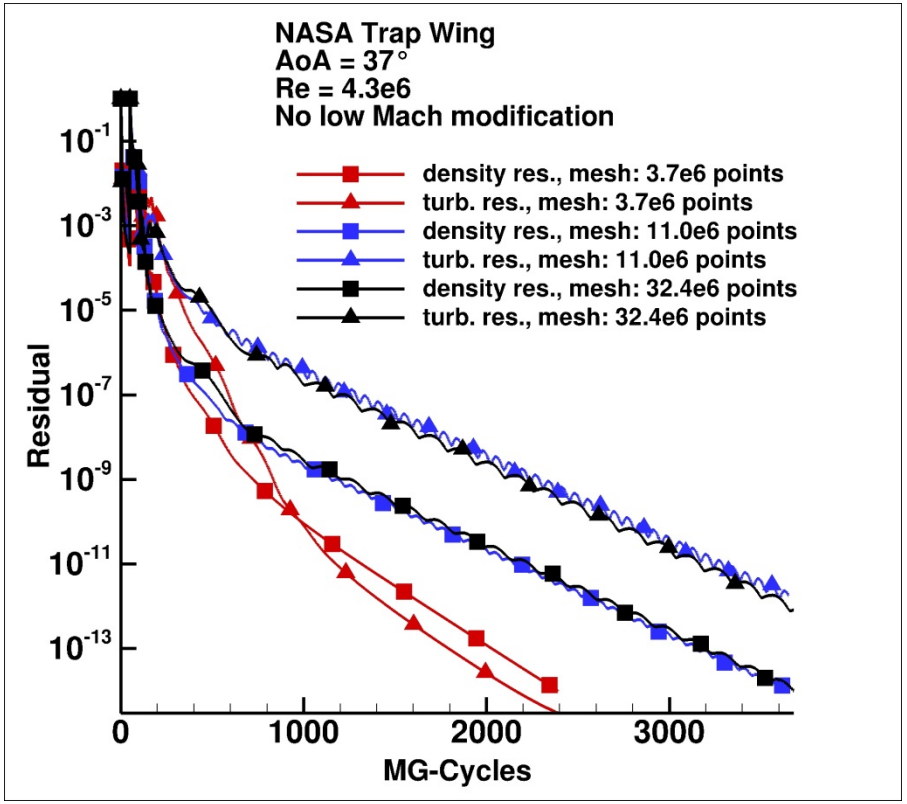
Unstructured mesh results for **AOA = 37°**

- **Residual has been reduced to machine accuracy using Newton kind methods**

→ **Steady state could not be found with simplified algorithms**



Flow field at the 60% wing section



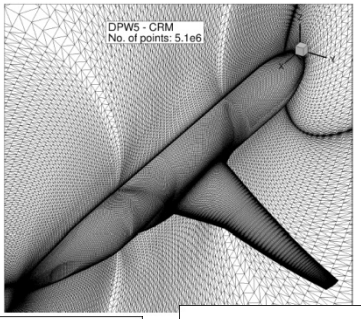
Convergence history for AoA = 37°

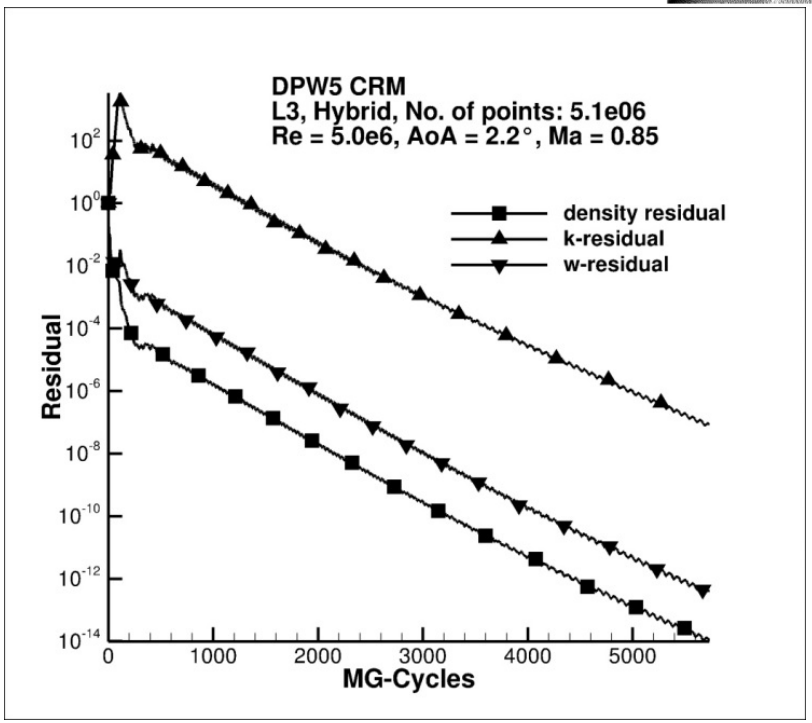
Outline

- Introduction and Motivation – The RANS equations
- Solution algorithms – Multigrid smoothers
- Best practice considerations
- Numerical examples
  - Spalart-Allmaras (neg.)
  - Wilcox (kω)

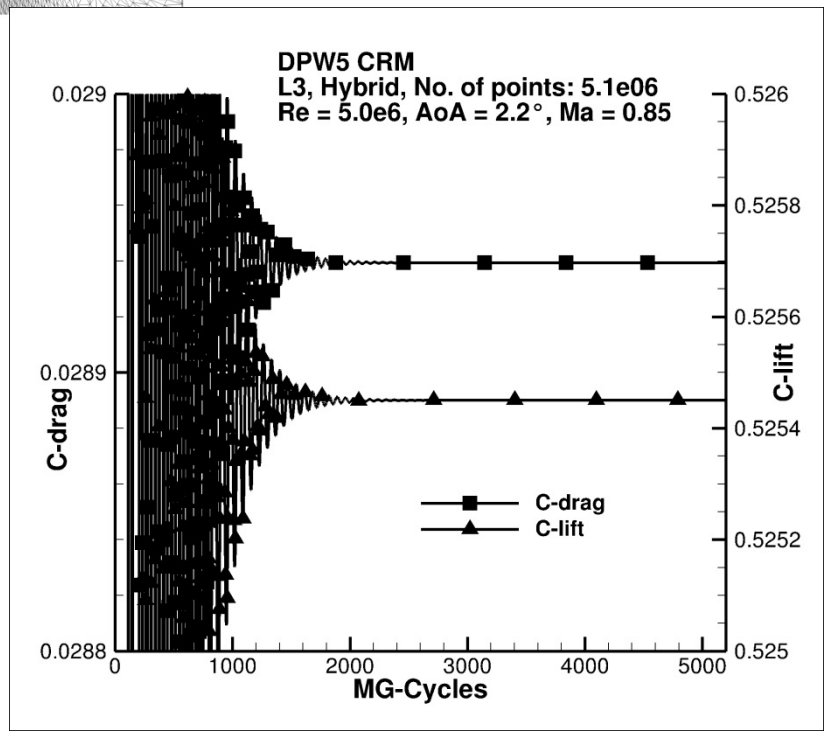
Numerical examples: DPW5 + kω-model

- Wing-body configuration
- Ma = 0.85
- α = 2.209°
- Re = 5e6
- No. of points: 5.1e6



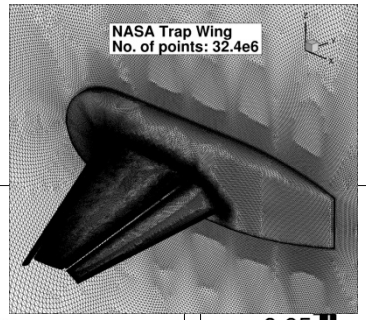


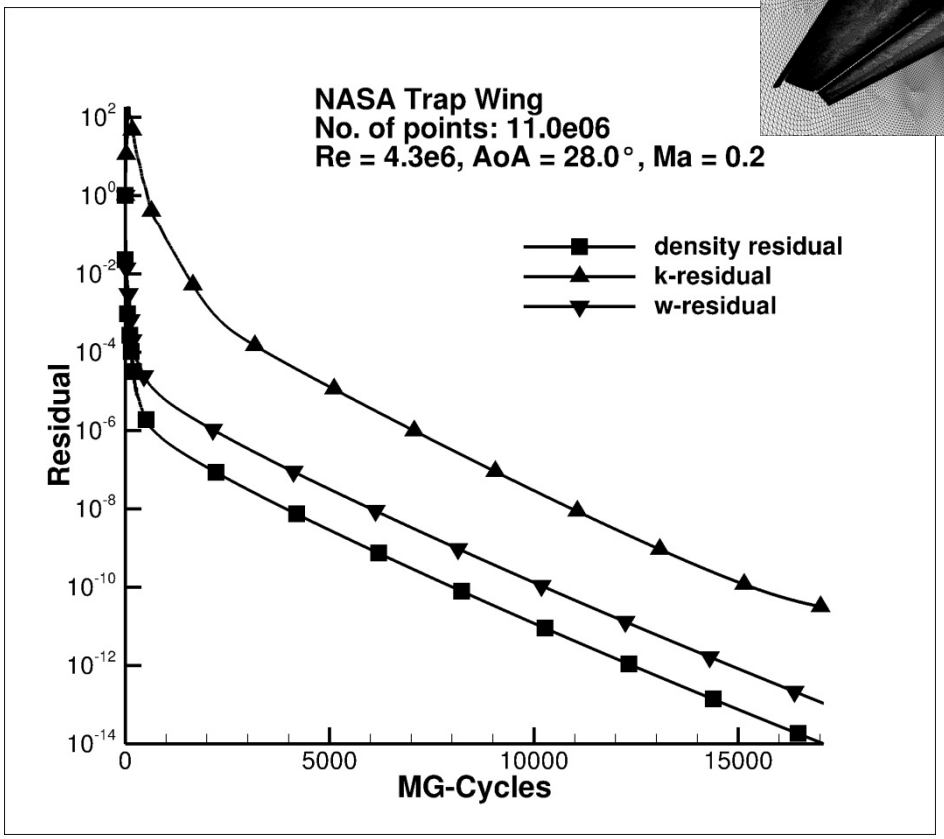
Convergence history of residuals

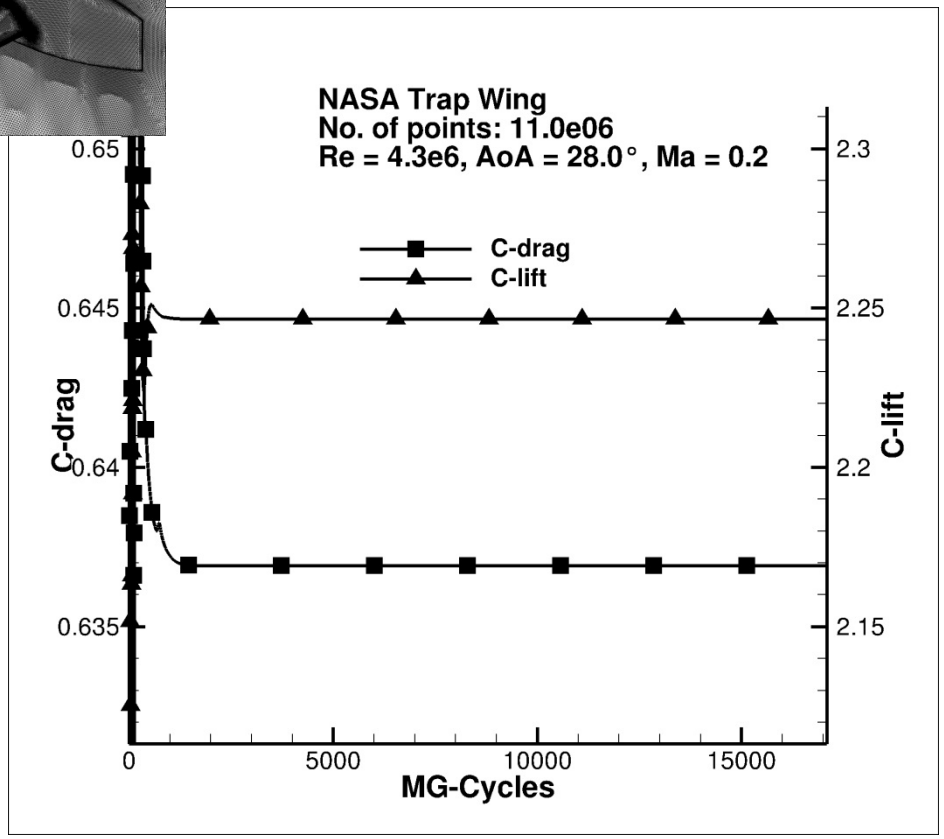


Convergence history of lift and drag

NASA Trap Wing + kω-model
Ma = 0.2, Re = 4.3e6, AoA = 28.0°



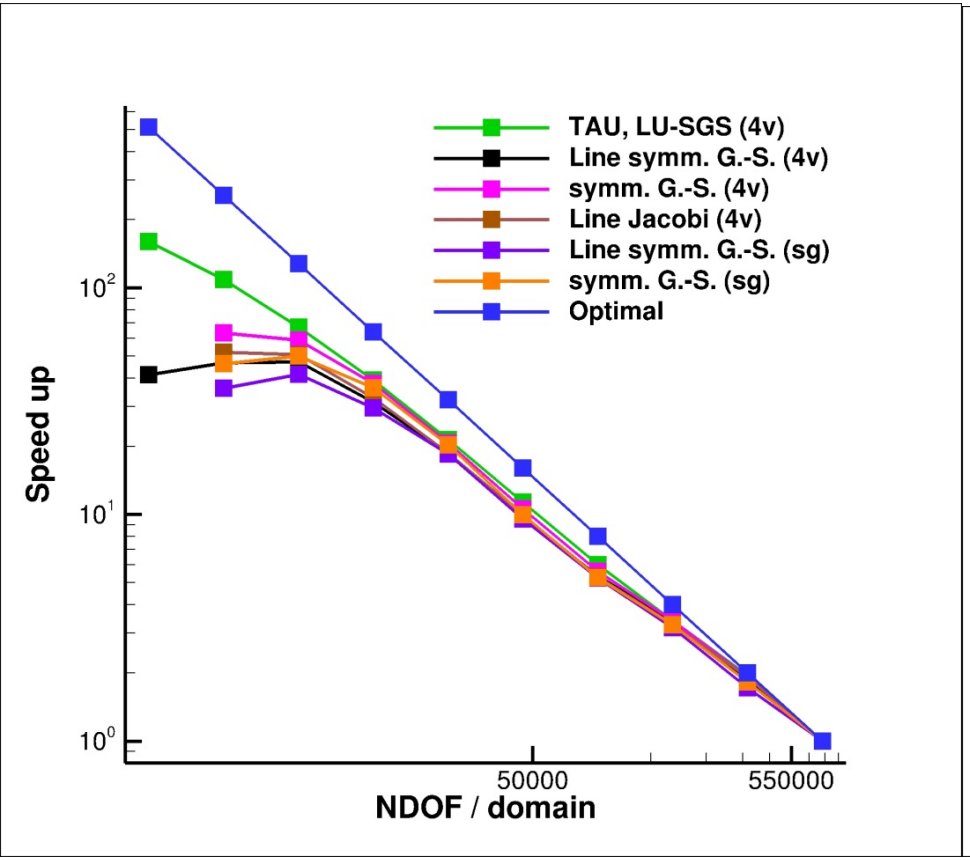




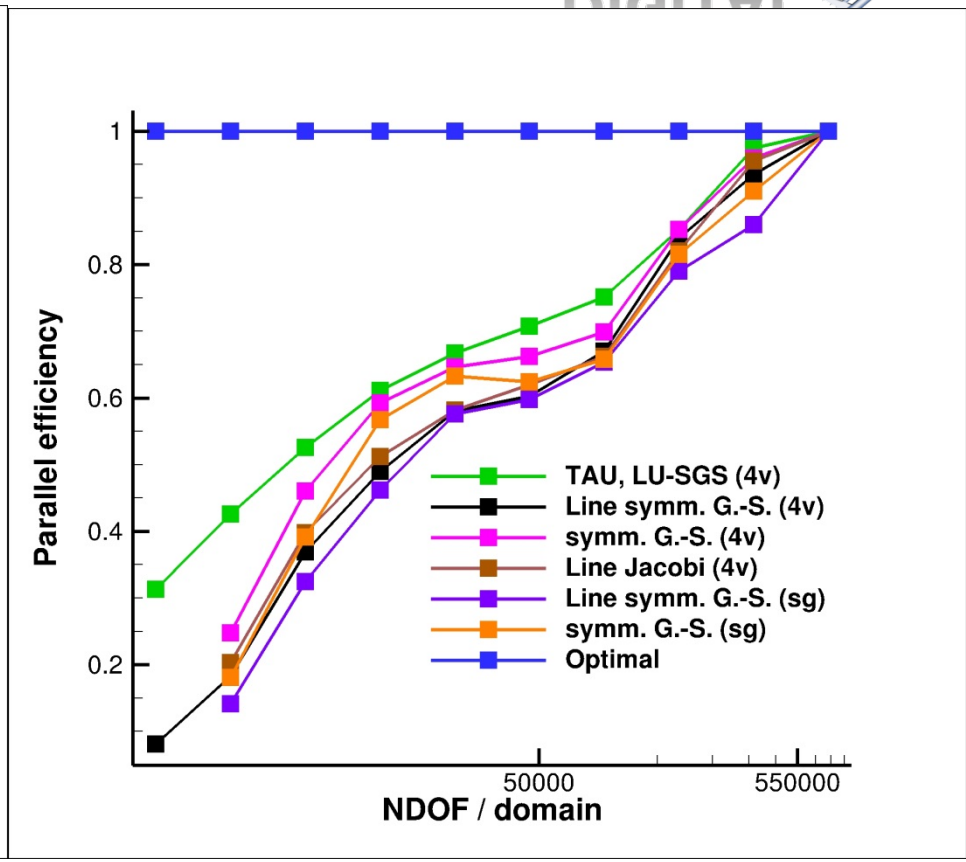
Convergence history of residuals

Convergence history of lift and drag

Speed up and parallel efficiency: Strong scaling



Actual speed up



Actual parallel efficiency,
System effectiveness

→Severe issue with respect to exploitation of modern hardware clusters

Summary

Implicit methods offer the potential to

- improve significantly the observed convergence rates
- find fully (machine accurate) converged solutions of complex flows
- significantly increase robustness (e.g. they work for a broad range of CFL numbers)
- implement the hierarchy of smoothers in one framework
- outsource and decouple the main work into a suited linear algebra package

Implicit methods require

- significantly more time per iteration than explicit methods
- a fully differentiated code which needs to be kept up to date
- significant more fast memory
- are not straightforward to ensure good parallel scalability
- to outsource and decouple the main work into a suited **linear algebra package**
- a new framework → Flucs code

Thank you!
Questions?