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 flightpath 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

\[ W := (\rho, \rho u, \rho E) \rightarrow \text{Conservative variables} \]

\[
\frac{d}{dt} \int_{\Omega} W \, dx = - \int_{\partial \Omega} (F_c(W, \text{grad } W) - F_v(W, \text{grad } W)) \cdot n \, ds
\]

\( F_c \): Convective flux
\( F_v \): Viscous flux

Unstructured Finite volume discretization

\rightarrow \text{Nonlinear operator equation:}

\[
\frac{dW}{dt} = -M^{-1}R(W), \quad M = \text{diag}(\text{vol}(\Omega_i)_{i=1,...,N})
\]

Interested in steady state solution: \( R(W) = 0 \)
Necessity for improvement of solution algorithms

Typical convergence behavior for high Reynolds number viscous flows

Convergence rate deteriorates significantly after initial phase

→ Stiff set of equations

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

No convergence, iteration stagnates

\[
\frac{\Delta x}{\Delta y} > 10000
\]
Outline

• Introduction and Motivation – The RANS equations

• Solution algorithms – Multigrid smoothers

• Globalization strategies

• Numerical examples
Structure of solution algorithm

Nonlinear multigrid: Solves $R(W) = 0$

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

Krylov subspace method: Solves $Ph = R$

Linear Preconditioner: Solves $P_x w = P v$

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
Structure of solution algorithm

Nonlinear multigrid: Solves $R(W) = 0$

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

Krylov subspace method: Solves $P_h = R$

Linear Preconditioner: Solves $P_{\text{prec}} w = P v$

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

\[
\frac{dW}{dt} = -M^{-1}R(W), \quad M = \text{diag}\left(\text{vol}(\Omega_i)_{i=1,...,N}\right)
\]

Apply preconditioned expl. Runge-Kutta method

\[
\begin{align*}
W^{(0)} &:= W_n \\
W^{(j)} &:= W^{(0)} - \alpha_{j+1,j}P_j^{-1}R(W^{(j-1)}), \quad j = 1,\ldots,s \\
W_{n+1} &:= W^{(s)} \\
P_j &:= (\Delta t)^{-1}M + \frac{\partial R}{\partial W}
\end{align*}
\]

to approximate \( W \) such that \( R(W) \approx 0 \)
Multigrid smoother: **Prec. Runge-Kutta method**

\[ \frac{dW}{dt} = -M^{-1}R(W), \quad M = \text{diag}(\text{vol}(\Omega_i)_{i=1,\ldots,N}) \]

Apply **preconditioned** expl. Runge-Kutta method

\[
\begin{align*}
W^{(0)} &:= W_n \\
W^{(j)} &:= W^{(0)} - \alpha_{j+1,j} \left( \frac{CFL\Delta t_i}{\text{vol}(\Omega_i)} \right) R(W^{(j-1)}), \quad j = 1,\ldots,s \\
W_{n+1} &:= W^{(s)} \\
R_j &:= (\Delta t)^{-1} M + \frac{\partial R}{\partial W} = \left( \frac{CFL\Delta t_i}{\text{vol}(\Omega_i)} \right)^{-1} = \left( \frac{\text{vol}(\Omega_i)}{CFL\Delta t_i} \right)
\end{align*}
\]

to approximate \( W \) such that \( R(W) \approx 0 \)

\[ \Rightarrow \text{Requires inversion of a scalar value for each control volume:} \]

\[ \left( \frac{\text{vol}(\Omega_i)}{CFL\Delta t_i} \right)^{-1} \]
Multigrid smoother: Prec. Runge-Kutta method

\[ \frac{dW}{dt} = -M^{-1}R(W), \quad M = \text{diag}(\text{vol}(\Omega_i)_{i=1,\ldots,N}) \]

Apply preconditioned expl. Runge-Kutta method

\[
\begin{align*}
W^{(0)} &:= W_n \\
W^{(j)} &:= W^{(0)} - \alpha_{j+1,j} P_j^{-1} R(W^{(j-1)}), \quad j = 1,\ldots,s \\
W_{n+1} &:= W^{(s)} \\
P_j &:= (\Delta t)^{-1} M + \frac{\partial R}{\partial W}
\end{align*}
\]

\[ x = P_j^{-1}R(W^{(j-1)}) \iff P_j x = R(W^{(j-1)}) \]

to approximate \( W \) such that \( R(W) \approx 0 \)

\( \rightarrow \) Task: Need to approximate efficiently solution of \( P x = R(W) \)

\( \rightarrow \) Inversion of scalar value is replaced by solving a large scale linear system.
The connection to Newton’s method

Outer Loop: Multistage Runge-Kutta method \( \rightarrow \) Choose \( s = 1 \), i.e. only one stage

\[
W^{(0)} := W_n
\]

\[
W^{(j)} := W^{(0)} - \alpha_{j+1,j} P_j^{-1} R(W^{(j-1)}), \quad j = 1, \ldots, s
\]

\[
W_{n+1} := W^{(s)}
\]

\[
P_j := (\Delta t)^{-1} M + \frac{\partial R}{\partial W} = \frac{\partial R}{\partial W}, \quad CFL \to \infty
\]

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

\[\rightarrow\] The solution method is some kind of \textbf{generalization of Newton’s method}
\[ P x = R(W) \]

- 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 $dR$, $P = \Delta t + dR$
    - 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

\[
R_i^{2nd}(W_i, W_{j\in N(i)}, W_{k,k\in N(j)}) \approx R_i^{1st}(W_i, W_{j,j\in N(i)})
\]

\[
\Rightarrow \frac{\partial R^{2nd}}{\partial W} \approx \frac{\partial R^{1st}}{\partial W}
\]

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

→ 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

\[
\text{Prec} = \begin{pmatrix}
& \text{Anisotropic part} & \\
\text{Isotropic part} & \\
\end{pmatrix}
\]

Point ordering according to lines yields (block) tridiagonal systems
Iterative solution methods for $\text{Prec } w = z$

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:**

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

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 $dR$, $P = \Delta t + dR$
    - 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 $\leftrightarrow$ 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

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

Smoothing step

- execute preconditioned expl. Runge-Kutta algorithm
- evaluate residual \( R \)
- evaluate derivative \( dR, P = \Delta t + dR \)
- solve linear system \( (P, R) \)
  - apply preconditioned Krylov subspace method
    - construct a (further) preconditioner for the linear system \( Px = R \)
    - solve \( \text{Prec} w = b \) to precondition \( Px = R \)
- update flow variables \( W \)
Derivation of low cost smoothers

\[
W^{(0)} := W_n
\]

\[
W^{(j)} := W^{(0)} - \alpha_{j+1, j} P_j^{-1} R(W^{(j-1)}), \quad j = 1, \ldots, s
\]

\[
W_{n+1} := W^{(s)}
\]

Simplifications:
1. Number of Krylov steps = 0

Smoothing step

- execute preconditioned expl. Runge-Kutta algorithm
  - evaluate residual \( R \)
  - evaluate derivative \( dR, P = \Delta t + dR \)
  - solve linear system \((P, R)\)
    - apply preconditioned Krylov subspace method
      - construct a (further) preconditioner for the linear system \( Px = R \)
      - solve \( \text{Prec} w = b \) to precondition \( Px = R \)
  - update flow variables \( W \)
Derivation of low cost smoothers

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

Simplifications:
1. Number of Krylov steps = 0

Smoothing step

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

- 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 $W$

Smoothing step

Simplifications:
1. Number of Krylov steps = 0

Preconditioning based on 1st order approximate Jacobian
(Swanson, Rossow, Yoon + Jameson (LU-SGS))
Derivation of low cost smoothers

- execute preconditioned expl. Runge-Kutta algorithm
- evaluate residual $R$
- evaluate derivative $dR\text{\textsuperscript{1st}}$
  
  $$\text{Prec}_{j} = \Delta t + dR\text{\textsuperscript{1st}} \approx \Delta t + o(dR\text{\textsuperscript{1st}})$$

- solve $\text{Prec} w = b$
- update flow variables $W$

Simplifications:
1. Number of Krylov steps = 0
2. Simplify $dR\text{\textsuperscript{1st}}$ entries by spectral radius

Solve with 1 symmetric Gauss-Seidel sweep

Yoon + Jameson (LU-SGS)
Derivation of low cost smoothers

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

Smoothing step

- execute preconditioned expl. Runge-Kutta algorithm
- evaluate residual \( R \)
- evaluate derivative \( dR^{1st} \), \( \text{Prec} = \Delta t + dR^{1st} \)
  - solve \( \text{Prec} w = b \)
  - 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

\[
x^{(0)} = 0
\]

\[
x^{(m+1)}_{L_i} = \text{tridiag} \left( D_{L_i} \right)^{-1} \left[ b_{L_i} - \sum_{j \notin L_i, \ldots, L_{i-1}, j \notin L_i} \text{Prec}_{L_i,j} x^{(m)}_{j} \right]
\]
Derivation of low cost smoothers

\[ W^{(0)} := W_n \]
\[ W^{(j)} := W^{(0)} - \alpha_{j+1,j}(\text{Prec}_j^{-1} R(W^{(j-1)})) \quad j = 1, \ldots, s \]
\[ W_{n+1} := W^{(s)} \]

Smoothing step

- execute preconditioned expl. Runge-Kutta algorithm
  - evaluate residual R
  - evaluate derivative \( dR^{1st} \), \( \text{Prec} = \Delta t + dR^{1st} \)
    - solve \( \text{Prec} w = b \)
  - 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

\[ x^{(0)} = 0 \]
\[ x_{L_i}^{(1)} = \text{tridiag}(D_{L_i})^{-1}b_{L_i} \]
Derivation of low cost smoothers

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

Smoothing step

\[ W^{(0)} := W_{n} \]
\[ W_{L_k}^{(j)} := W_{L_k}^{(0)} - \alpha_{j+1,j} \text{tridiag}(D_{L_k}) \cdot R(W^{(j-1)}) \]
\[ W_{n+1} := W^{(s)} \]

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

$\alpha = 0$

Line implicit Runge-Kutta (Mavriplis)

\[ x^{(0)} = 0 \]
\[ x_{L_i}^{(1)} = \text{tridiag}(D_{L_i})^{-1}b_{L_i} \]
Derivation of low cost smoothers

- execute preconditioned expl. Runge-Kutta algorithm
- evaluate residual $R$
- evaluate derivative $dR^{1st}$, $\text{Prec} = \Delta t + dR^{1st}$
  - solve $w = (D)^{-1}b$
  - 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
4. Neglect lines, i.e. perform only Jacobi iteration

Point implicit Runge-Kutta (Giles, Moinier)
Derivation of low cost smoothers

- execute preconditioned expl. Runge-Kutta algorithm
- evaluate residual \( R \)
- evaluate derivative \( dR^{1st} \), Prec = \( \Delta t + dR^{1st} \)
  - solve \( w = (\Delta t / \text{vol})^{-1} b \)
  - 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
4. Neglect lines, i.e. perform only Jacobi iteration
5. Approximate diagonal terms of Jacobian by spectral radius

Explicit Runge-Kutta (Jameson)

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

\[ W^{n+1} = W^n - \left[ \frac{\partial R}{\partial W} \right]^{-1} R(W^n) \]

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
- …..

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:**

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

\[
W^{(0)} := W_n
\]
\[
W^{(j)} := W^{(0)} - \alpha_{j+1,j} P^{-1,app}_j R(W^{(j-1)})
\]
\[
W_{n+1} := W^{(s)}
\]

**Linearized Problem**

\[
\frac{dW}{dt} \approx -M^{-1}\left\{ R(W^*) + \frac{\partial R}{\partial W} [W^*] \Delta W \right\}
\]

\[
W^{(0)} := W_n
\]
\[
W^{(j)} := W^{(0)} - \alpha_{j+1,j} P^{-1,app}_j \frac{\partial R}{\partial W} W^{(j-1)}
\]
\[
W_{n+1} := W^{(s)} \iff W^{(n+1)} = q_s\left(P^{-1,app}_j \frac{\partial R}{\partial W}\right) W^{(j-1)}
\]

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

**Convergence**

\[
\iff \rho\left(q_s\left(P^{-1,app}_j \frac{\partial R}{\partial W}\right)\right) < 1
\]
Analysis of schemes: Impact of CFL number

Analysis for mesh with 5.2e6 points:

Investigation of number of stages for
• 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 off

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)
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 successfully implemented into the DLR TAU-Code
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
- $Ma = 0.8$
- $\alpha = 3.0^\circ$
- $Re = 14e6$
- Pure hexahedral mesh: $3.3e6$ points

$\rightarrow$ Implicit method converges,
$\rightarrow$ 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 $\alpha = 13\degree$, $28\degree$, $32\degree$, $34\degree$, $37\degree$

- 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, \( \text{Ma} = 0.2, \text{Re} = 4.3 \times 10^6 \)

Unstructured mesh results for \( \text{AOA} = 37^\circ \)

- 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 \( \text{AoA} = 37^\circ \)
Outline

• Introduction and Motivation – The RANS equations

• Solution algorithms – Multigrid smoothers

• Best practice considerations

• Numerical examples
  • Spalart-Allmaras (neg.)
  • Wilcox (kω)
Numerical examples: DPW5 + k\omega-model

- Wing-body configuration
- Ma = 0.85
- \( \alpha = 2.209^\circ \)
- Re = 5e6
- No. of points: 5.1e6
NASA Trap Wing + $k\omega$-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
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?