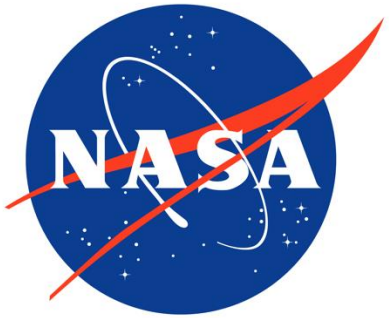
Implicit Preconditioning for Explicit Multigrid Solvers on Cut-Cell Cartesian Meshes

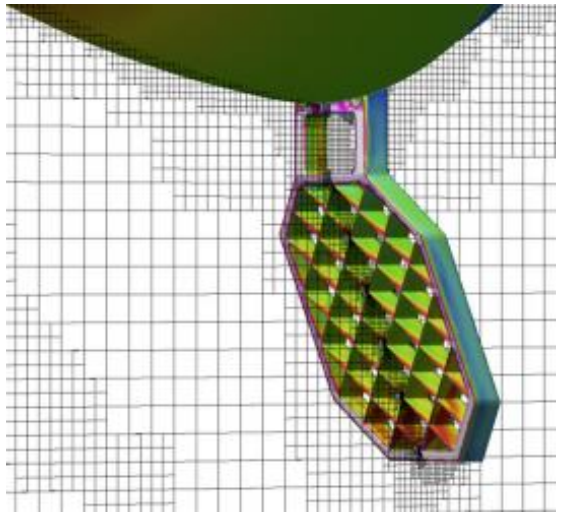


Jonathan Chiew & **Michael Aftosmis**

Computational Aerosciences Branch

Ames Research Center

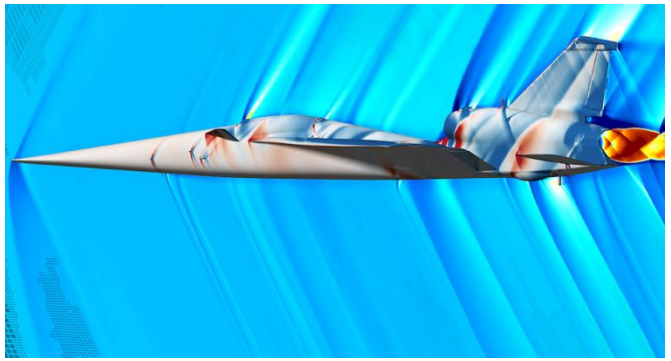
Cut-cell Cartesian meshes enable robust, automated meshing for CFD

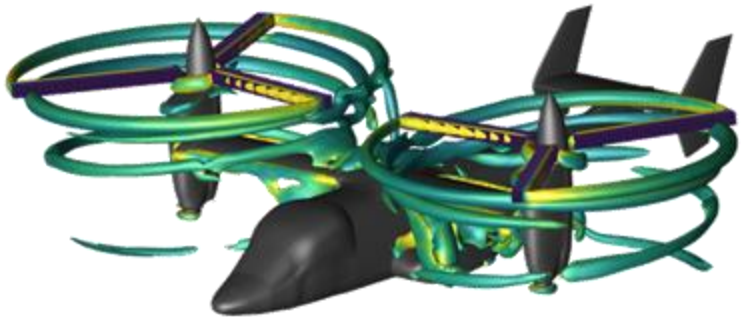


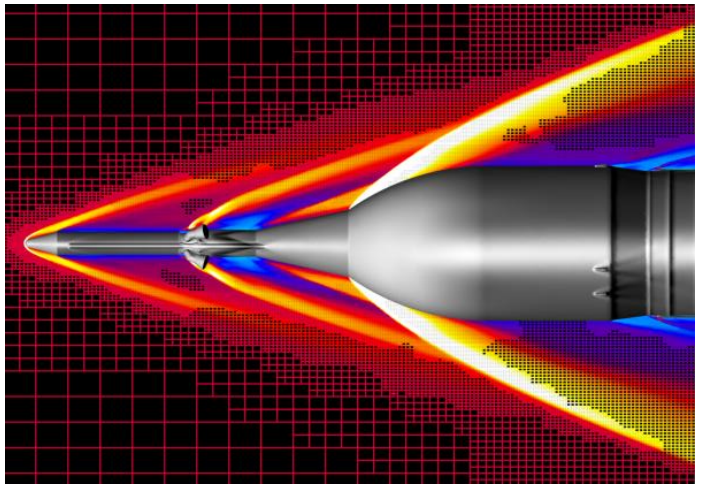
Automated mesh generation and adaptation are essential for vehicle design and optimization

Complex configurations can be meshed automatically with **cut-cell Cartesian grids**

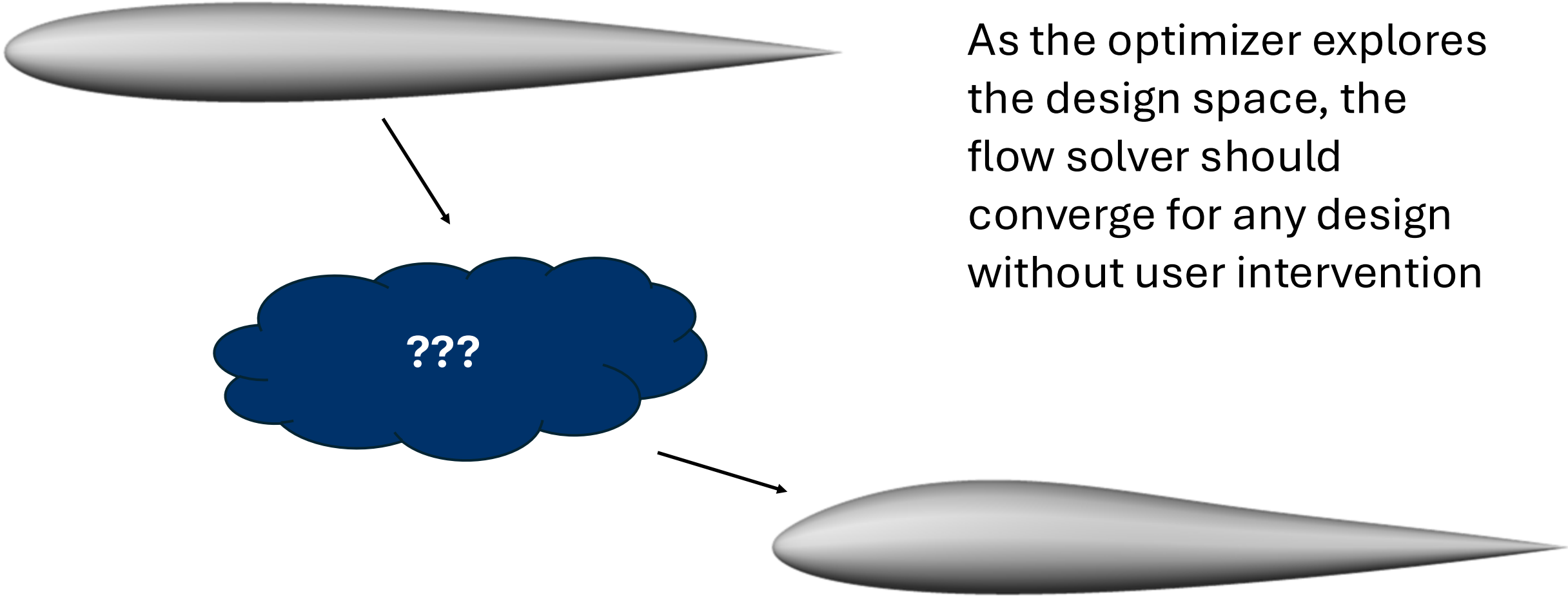
Mesh adaptation provides **error control** via the method of adjoint-weighted residuals







Robust mesh generation and solution algorithms are critical for aerodynamic shape optimization



As the optimizer explores the design space, the flow solver should converge for any design without user intervention

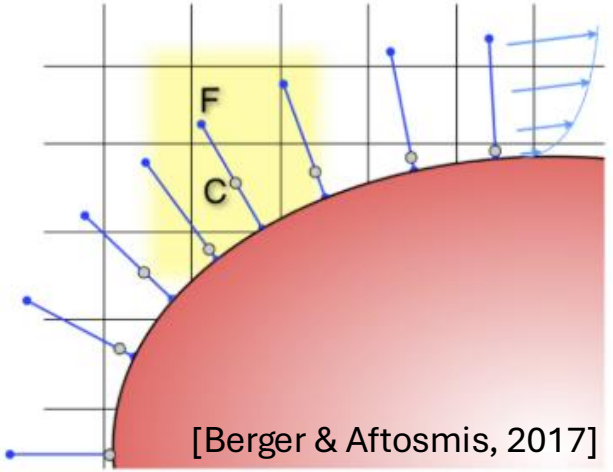
RANS extensions to Cartesian methods expands their design impact but requires stronger solvers

Developing **automated RANS capability** for design and optimization**:**

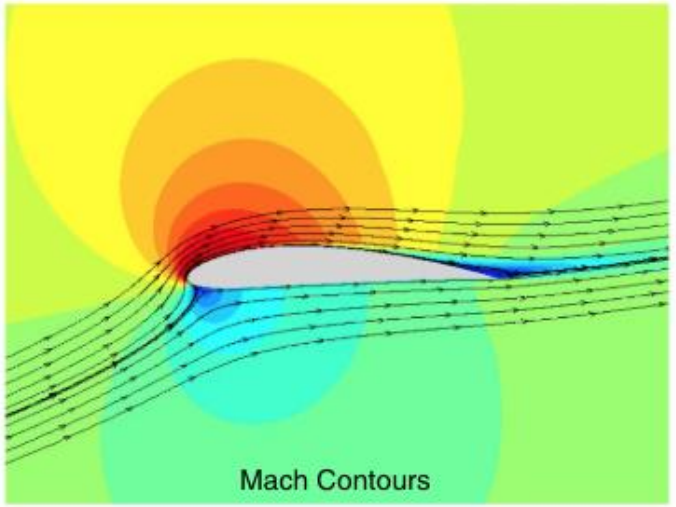
- Steady-state RANS on cut-cell Cartesian meshes
- ODE-based wall model

Improved analysis of non-traditional configurations, off-design considerations

Increased stiffness in the governing equations requires a **more robust solver**



[Berger & Aftosmis, 2017]



Implicit Preconditioning for Explicit Multigrid Solvers on Cut-Cell Cartesian Meshes

General Solver Framework

Baseline Explicit Multigrid Solver

Implicit Preconditioning

Steady-state Results

Dual Time Stepping Formulation

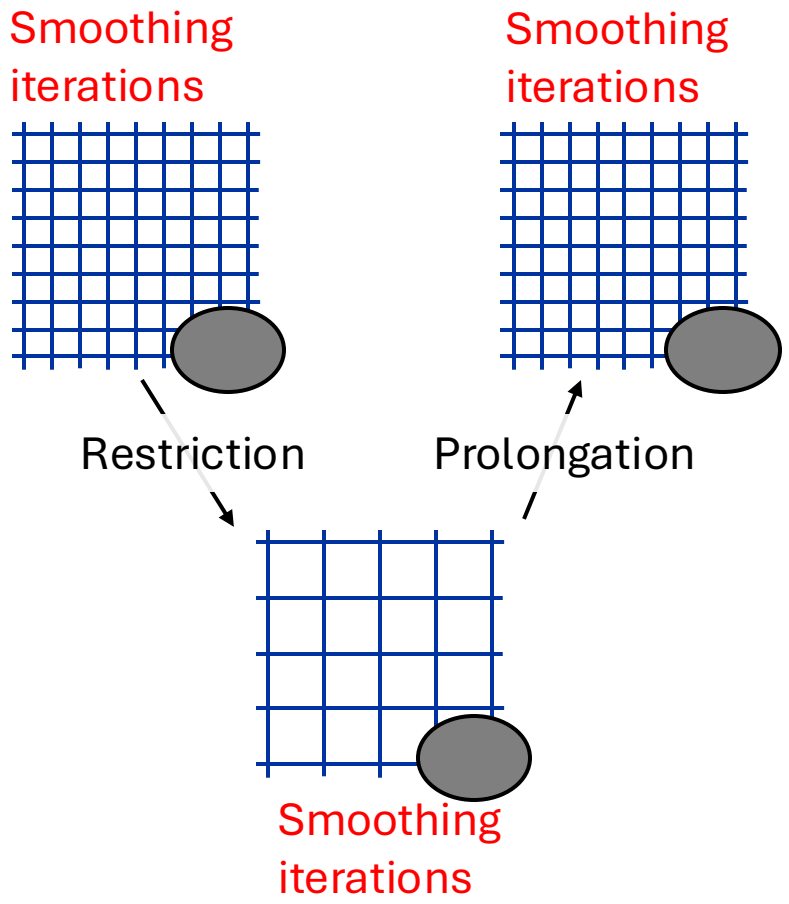
Time-dependent Results

Geometric multigrid acceleration is well suited for Cartesian meshes

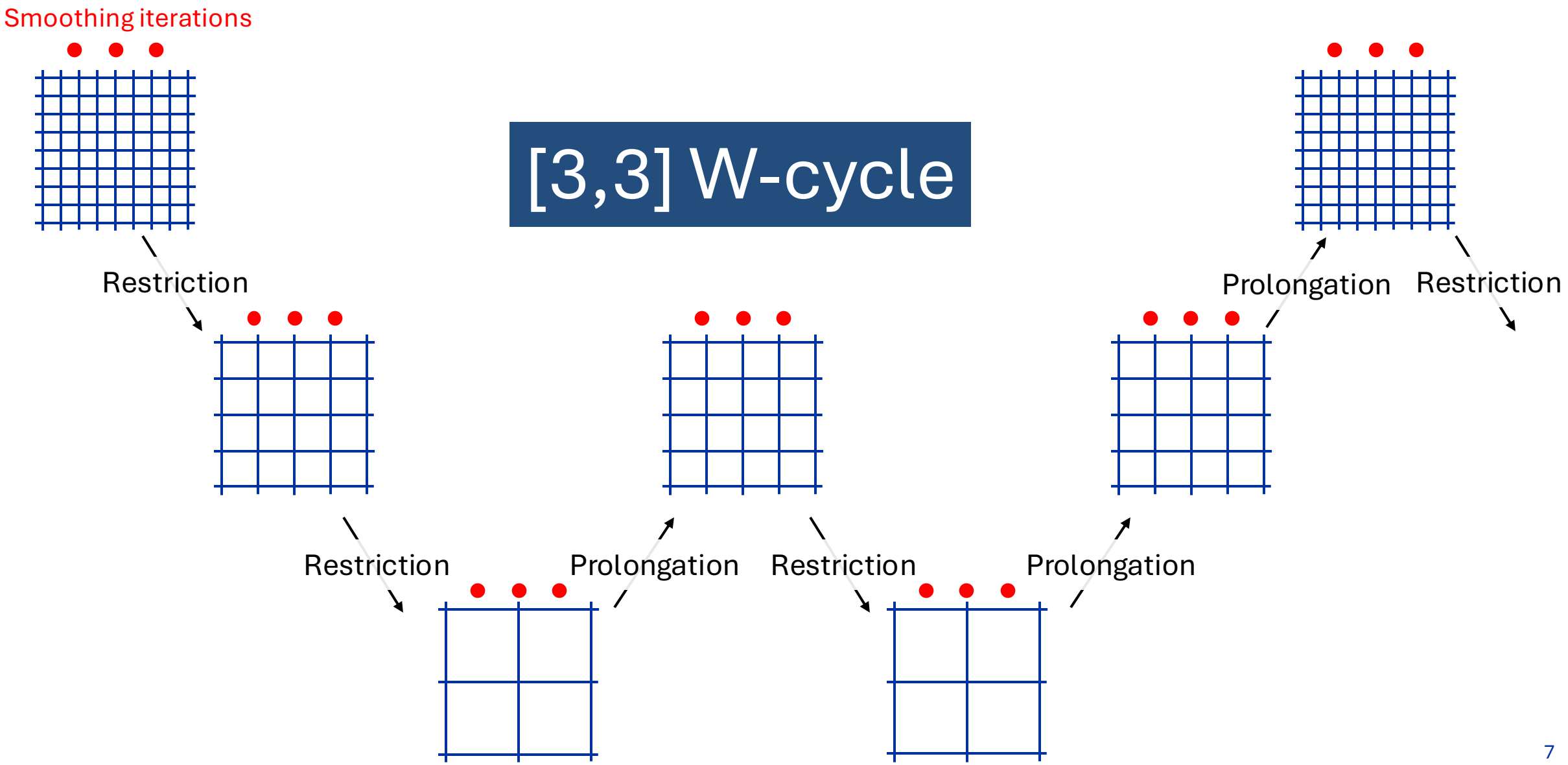
Cart3D's unstructured Cartesian meshes are easily coarsened

Coarsened cells away from the boundary surfaces are still Cartesian

Still need to choose multigrid cycle and smoothing method to reduce high wavenumber errors



The W multigrid cycle is commonly used in Cart3D



Employ a preconditioned multi-stage scheme as a general framework for the MG smoother

Solve $\mathcal{R}(\mathbf{U}) = \mathbf{0}$ with a K-stage scheme:

$$\mathbf{U}_0 = \mathbf{U}^n$$

$$\Delta\mathbf{U}_k = -\frac{\Delta\tau^n}{V}\mathcal{P}^{-1}\mathcal{R}(\mathbf{U}_{k-1}), \qquad k = 1 \ldots K$$

$$\mathbf{U}_k = \mathbf{U}_0 + \alpha_k \Delta\mathbf{U}_k, \qquad k = 1 \ldots K$$

$$\mathbf{U}^{n+1} = \mathbf{U}_K$$

An alternative form of the equations can be convenient to highlight the preconditioner

Solve $\mathcal{R}(\mathbf{U}) = \mathbf{0}$ with a K-stage scheme:

$$\mathbf{U}_0 = \mathbf{U}^n$$

$$\Delta\mathbf{U}_k = -\frac{\Delta\tau^n}{V}\mathcal{R}(\mathbf{U}_{k-1}), \qquad k = 1 \ldots K$$

$$\boxed{\mathcal{P}\overline{\Delta\mathbf{U}_k} = \Delta\mathbf{U}_k} \qquad k = 1 \ldots K$$

$$\mathbf{U}_k = \mathbf{U}_0 + \alpha_k\overline{\Delta\mathbf{U}_k} \qquad k = 1 \ldots K$$

$$\mathbf{U}^{n+1} = \mathbf{U}_K$$

This preconditioned multi-stage formulation encompasses many well known schemes

Consider a one-stage scheme with $\mathrm{P} = \mathrm{I}$ and $\alpha_1 = 1$

$$\mathbf{U}_0 = \mathbf{U}^n$$

$$\Delta\mathbf{U}_1 = -\frac{\Delta\tau^n}{V} I^{-1}\mathcal{R}\left(\mathbf{U}_0\right)$$

$$\mathbf{U}_1 = \mathbf{U}_0 + \Delta\mathbf{U}_1$$

$$\mathbf{U}^{n+1} = \mathbf{U}_K$$

This is the explicit Euler method

This preconditioned multi-stage formulation encompasses many well known schemes

Adding more stages with $\mathrm{P} = \mathrm{I}$ and suitably chosen α_i produces a family of explicit Runge-Kutta methods

$$\mathbf{U}_0 = \mathbf{U}^n$$

$$\Delta \mathbf{U}_k = -\frac{\Delta \tau^n}{V} I^{-1} \mathcal{R}\left(\mathbf{U}_{k-1}\right), \qquad k = 1 \ldots K$$

$$\mathbf{U}_k = \mathbf{U}_0 + \alpha_k \Delta \mathbf{U}_k, \qquad k = 1 \ldots K$$

$$\mathbf{U}^{n+1} = \mathbf{U}_K$$

Define the baseline smoothing algorithm

We choose the following parameters to define the baseline smoother for the remainder of this talk:

- 5 stages
- No preconditioner (i.e. $\mathrm{P}=\mathrm{I}$)
- [1,1] W-cycle
- Gradients only computed on the first stage

Note that this algorithm is **fully explicit** and **matrix-free**

Other choices for preconditioning are well known methods in the literature

Implicit residual smoothing – Laplacian preconditioner

$$\mathcal{P} = 1 - \epsilon \left[\frac{\Delta t}{\Delta x} \right]^2 \delta^2$$

Block-Jacobi

$$\mathcal{P} = \mathrm{diag} \left\{ \frac{\partial \mathcal{R}}{\partial \mathbf{U}} \right\}$$

Preconditioner

$$\mathcal{P} \overline{\Delta \mathbf{U}_k} = \Delta \mathbf{U}_k$$

Using the full flux Jacobian matrix is common for stronger solvers

Flux Jacobian preconditioners with a single-stage (K=1):

Newton's method / Linear multigrid

$$\mathcal{P} = \frac{\Delta\tau^n}{V}\frac{\partial \mathcal{R}}{\partial \mathbf{U}}$$

Linearized implicit Euler / Newton Pseudo-Transient Continuation

$$\mathcal{P} = \left[I + \frac{\Delta\tau^n}{V}\frac{\partial \mathcal{R}}{\partial \mathbf{U}}\right]$$

Preconditioner

$$\mathcal{P}\overline{\Delta\mathbf{U}_k} = \Delta\mathbf{U}_k$$

Multi-stage schemes with the full flux Jacobian matrix are also found in literature

Rosenbrock methods

$$\mathcal{P} = \left[I + \frac{\Delta\tau^n}{V} \frac{\partial \mathcal{R}}{\partial \mathbf{U}} \right]$$

RK / Implicit (Rossow, Swanson)

$$\mathcal{P} = \left[I + \epsilon_{imp} \frac{\Delta\tau^n}{V} \frac{\partial \mathcal{R}}{\partial \mathbf{U}} \right]$$

Preconditioner

$$\mathcal{P}\overline{\Delta \mathbf{U}_k} = \Delta \mathbf{U}_k$$

Implicit preconditioning requires solving the resulting system of linear equations

Consider the RK/Implicit (RKI) preconditioner

$$\left[\mathbf{I} + \epsilon_{\text{imp}} \frac{\Delta\tau}{V} \frac{\partial \mathcal{R}}{\partial \mathbf{U}}\right] \overline{\Delta \mathbf{U}_{\mathbf{k}}} = -\frac{\Delta\tau^n}{V} \mathcal{R}\left(\mathbf{U}_{k-1}\right)$$

Rossow, Swanson, & Jameson all used symmetric Gauss-Seidel iterations with off-block-diagonal terms moved to the RHS

Xu et al. (ILU0)and Langer et al. (SGS) used left-preconditioned Jacobian-free GMRES to solve the implicit equations

The implicit solver: 3-stage RKI scheme with Jacobian-free, unpreconditioned GMRES

The RKI-SGS methods only use 1-3 iterations to stabilize the updates with reasonably large CFL numbers [100,1000]

$$\left[\mathbf{I} + \epsilon_{\text{imp}} \frac{\Delta\tau}{V} \frac{\partial \mathcal{R}}{\partial \mathbf{U}}\right] \overline{\Delta \mathbf{U}_{\mathbf{k}}} = -\frac{\Delta\tau^n}{V} \mathcal{R}\left(\mathbf{U}_{k-1}\right)$$

Investigate whether similar convergence acceleration can be obtained with a matrix-free, scalable algorithm

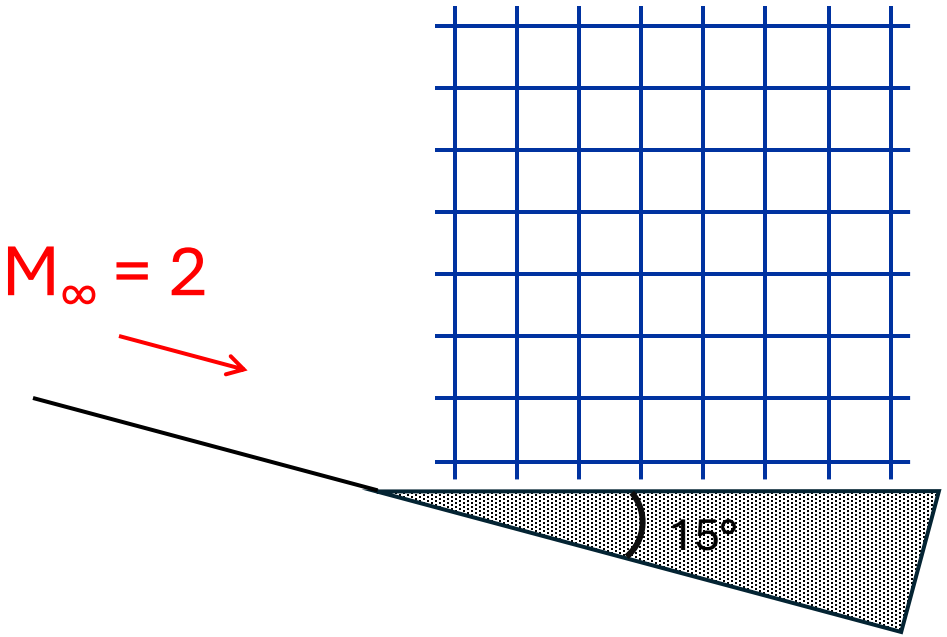
$$\frac{\partial \mathcal{R}}{\partial \mathbf{U}} v \approx \frac{\mathcal{R}\left(\mathbf{U} + \epsilon v\right) - \mathcal{R}\left(\mathbf{U}\right)}{\epsilon} \qquad \epsilon = \sqrt{\epsilon_{\text{mach}}} \left\langle \mathbf{U}, v \right\rangle$$

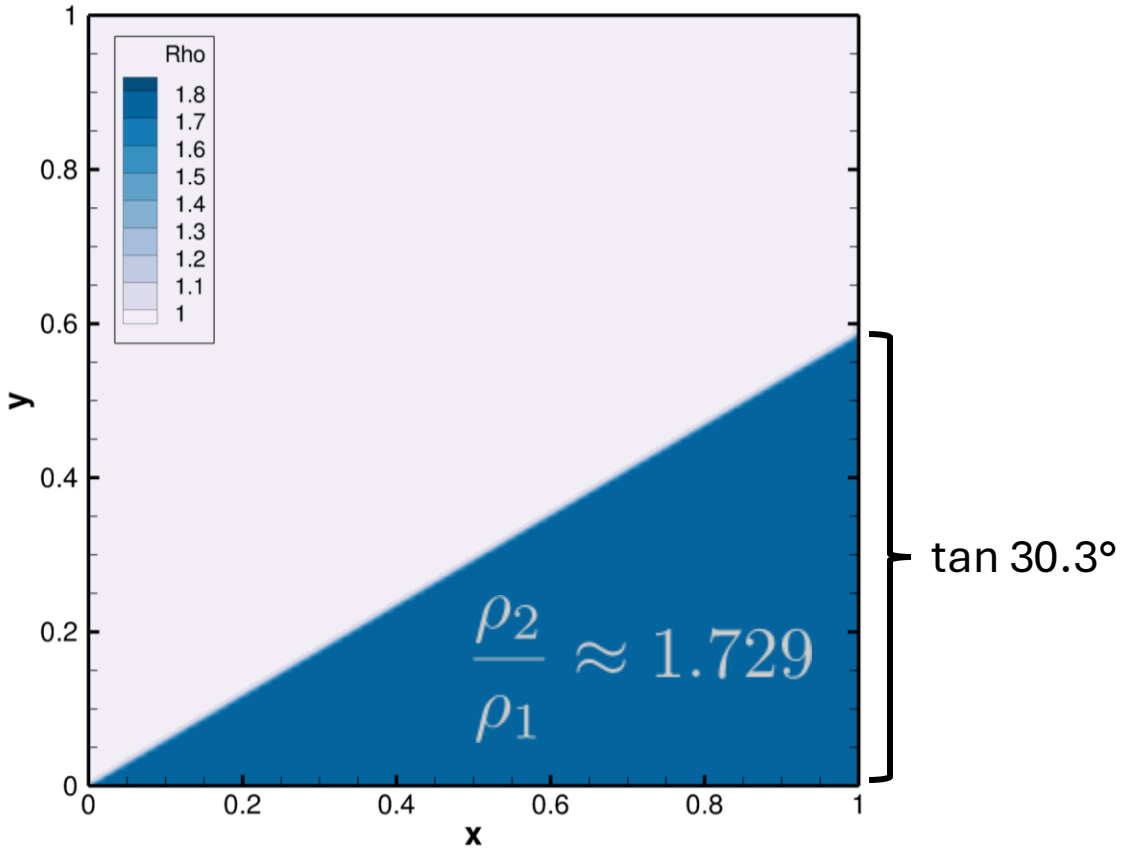
Supersonic wedge case for verification of the baseline and preconditioned solvers

Fully supersonic flow

Uniform 256x256 mesh

15° wedge aligned to grid

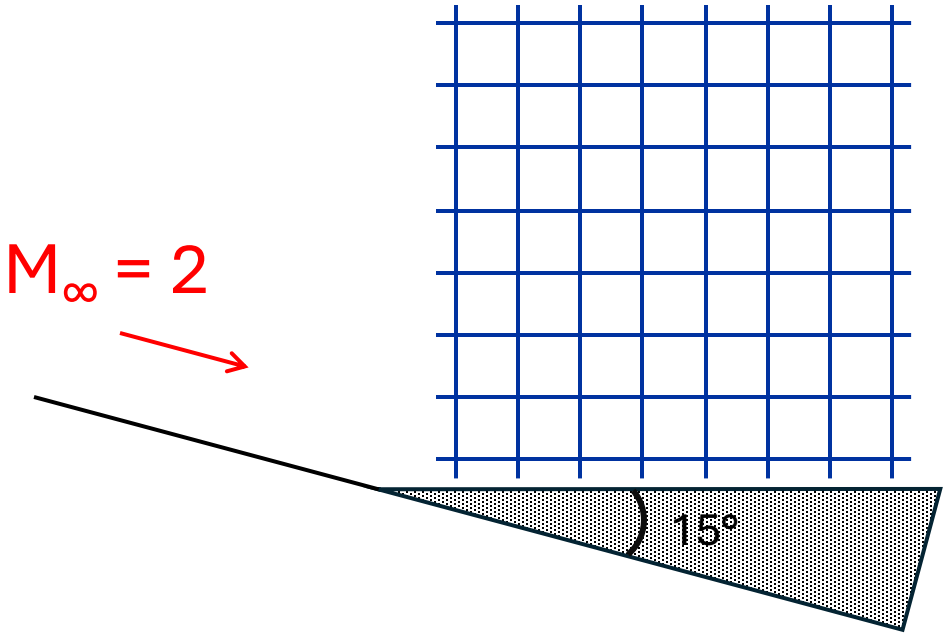


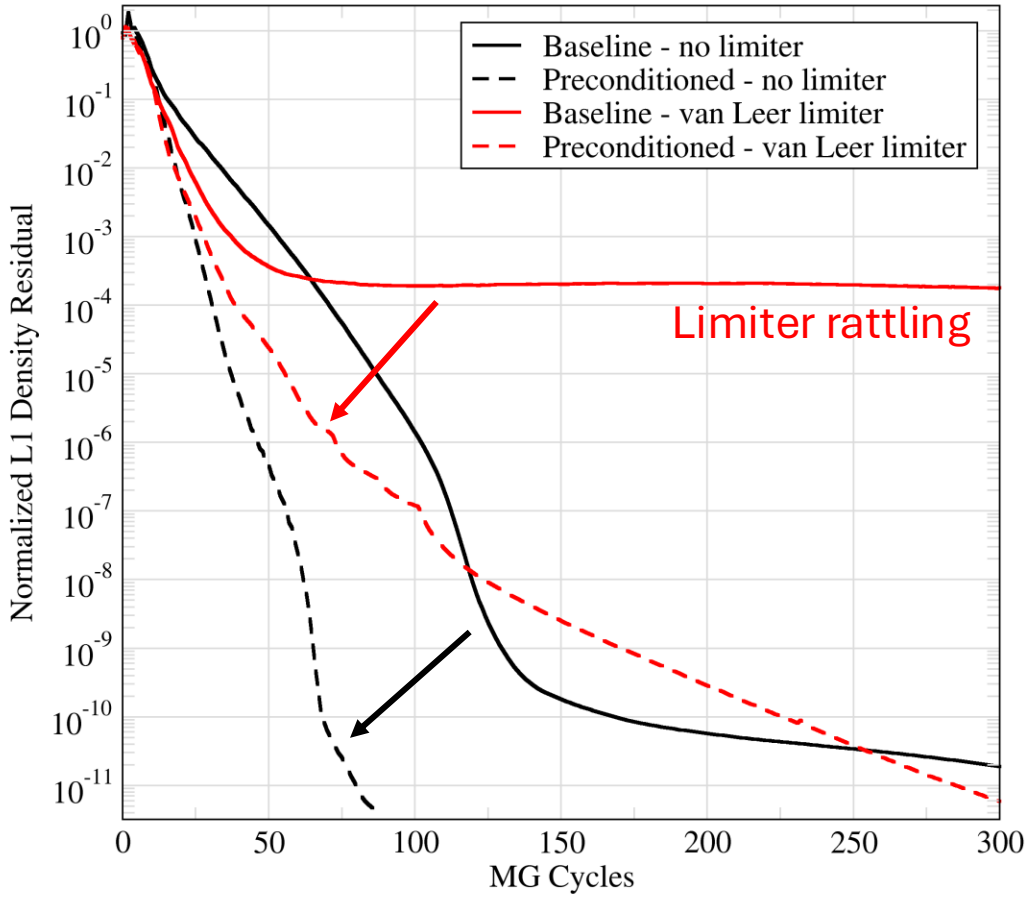


Preconditioned solver improves convergence rate on supersonic wedge case

8 Krylov vectors, max CFL = 1000

No limiter MG convergence rate improves from 0.88 → 0.74





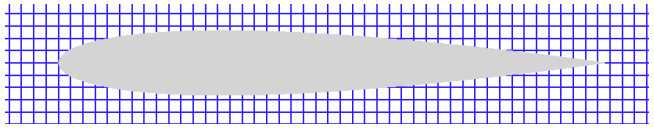
Further evaluation of the preconditioned solver with a more realistic case: NACA 0012 airfoil

NACA 0012 with closed TE

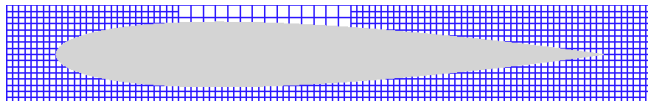
$M_\infty = 0.8$, $\alpha = 1.25°$

~10k cell mesh generated with 7 adaptation cycles (L/D)

4 level [1,1] W-cycle

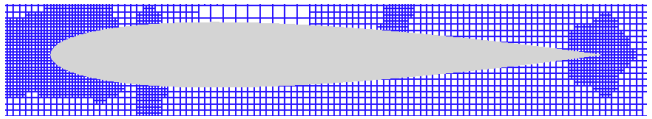


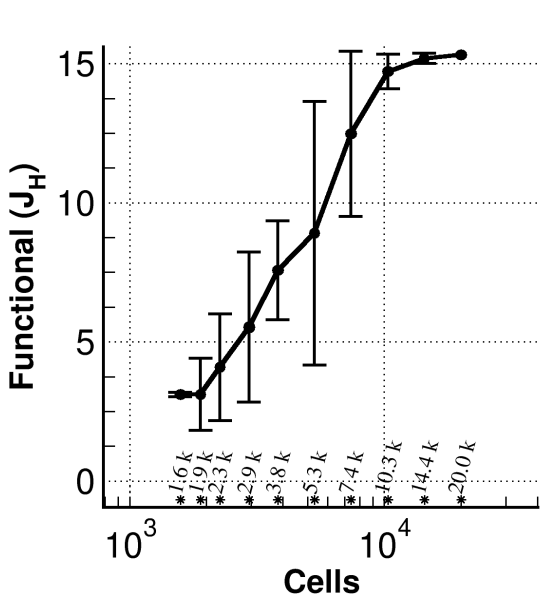
5th adaptation

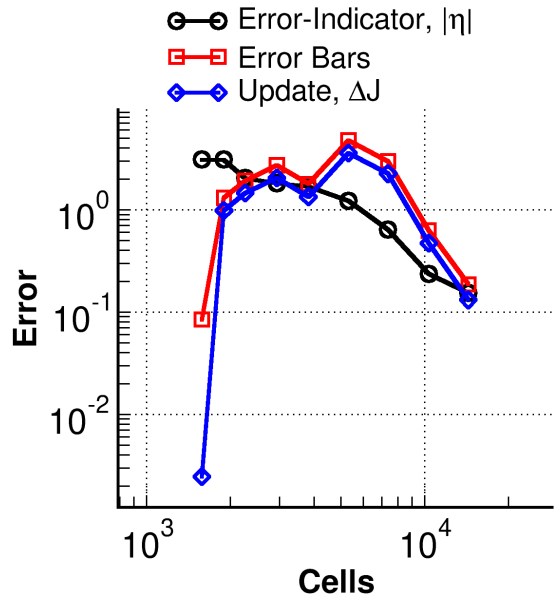


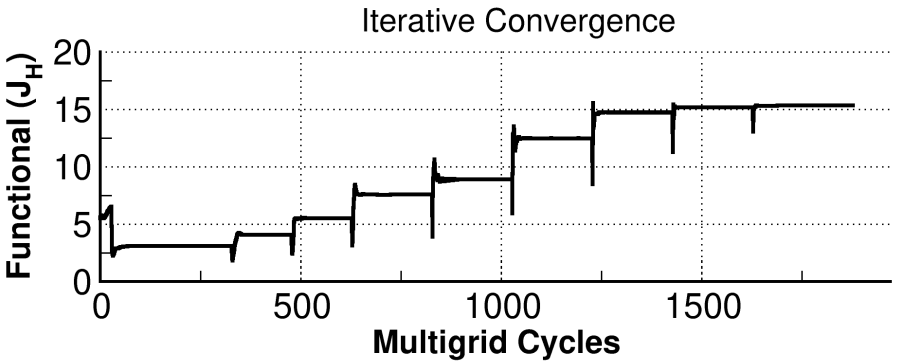
6th adaptation

7th adaptation









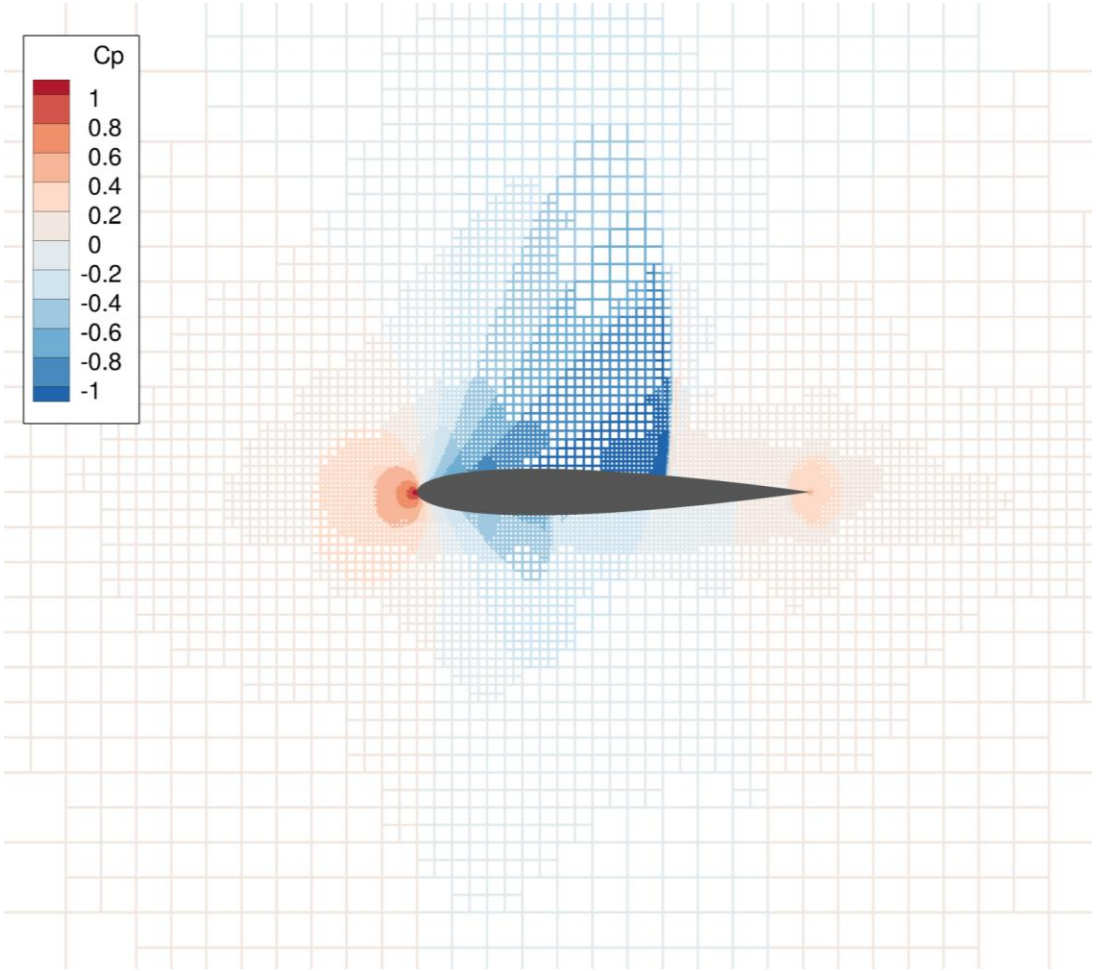
Further evaluation of the preconditioned solver with a more realistic case: NACA 0012 airfoil

NACA 0012 with closed TE

$M_\infty = 0.8$, $\alpha = 1.25°$

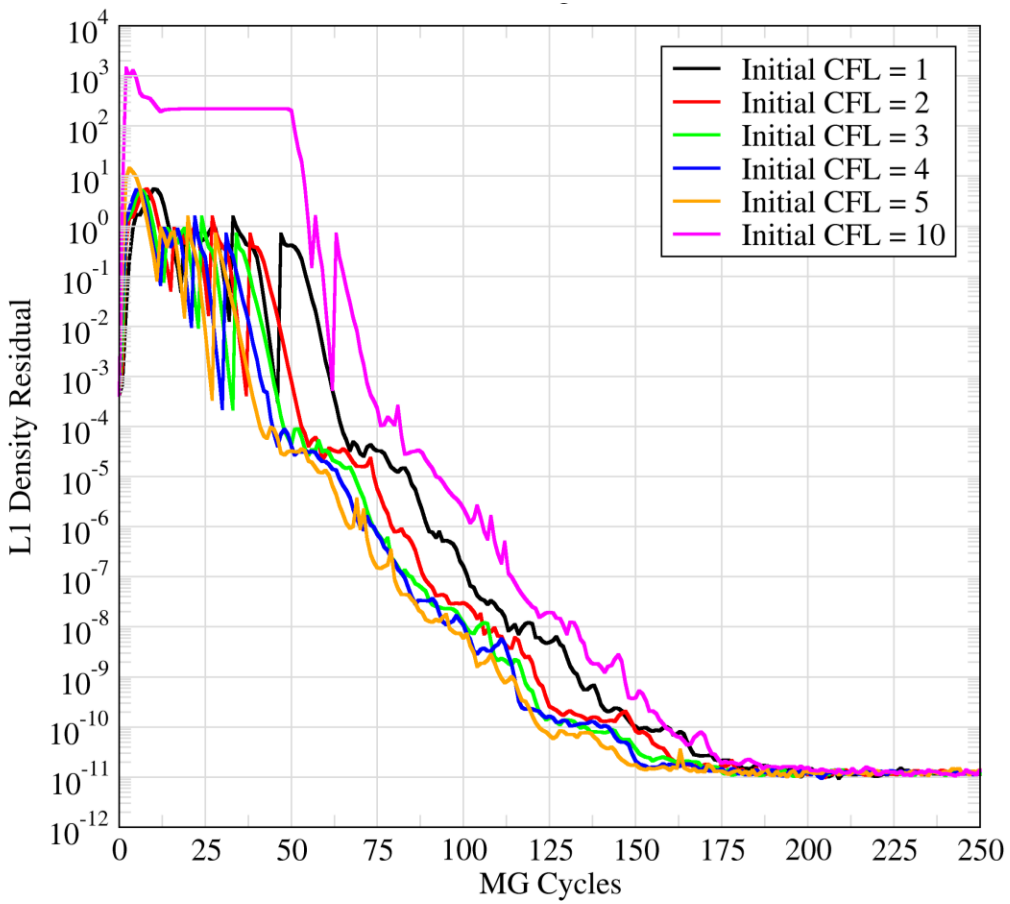
~10k cell mesh generated with 7 adaptation cycles

4 level [1,1] W-cycle

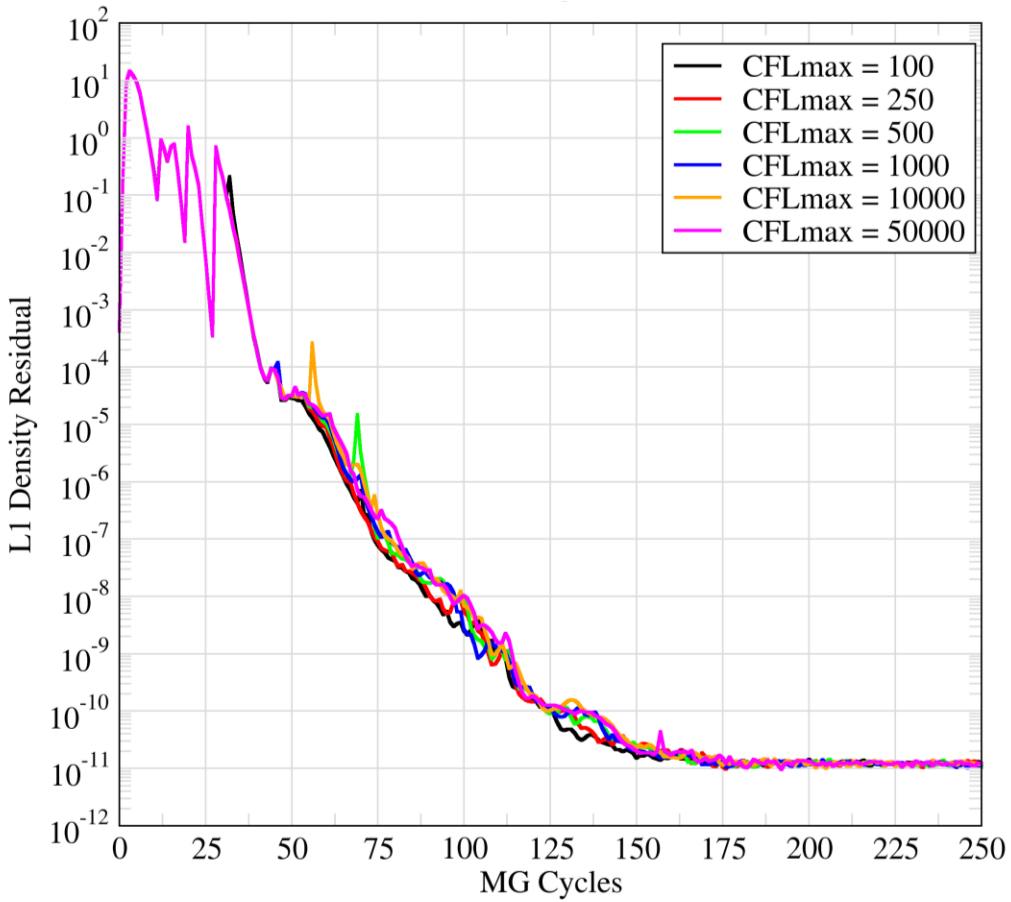


Most free parameters have a small effect on the preconditioned method convergence rate

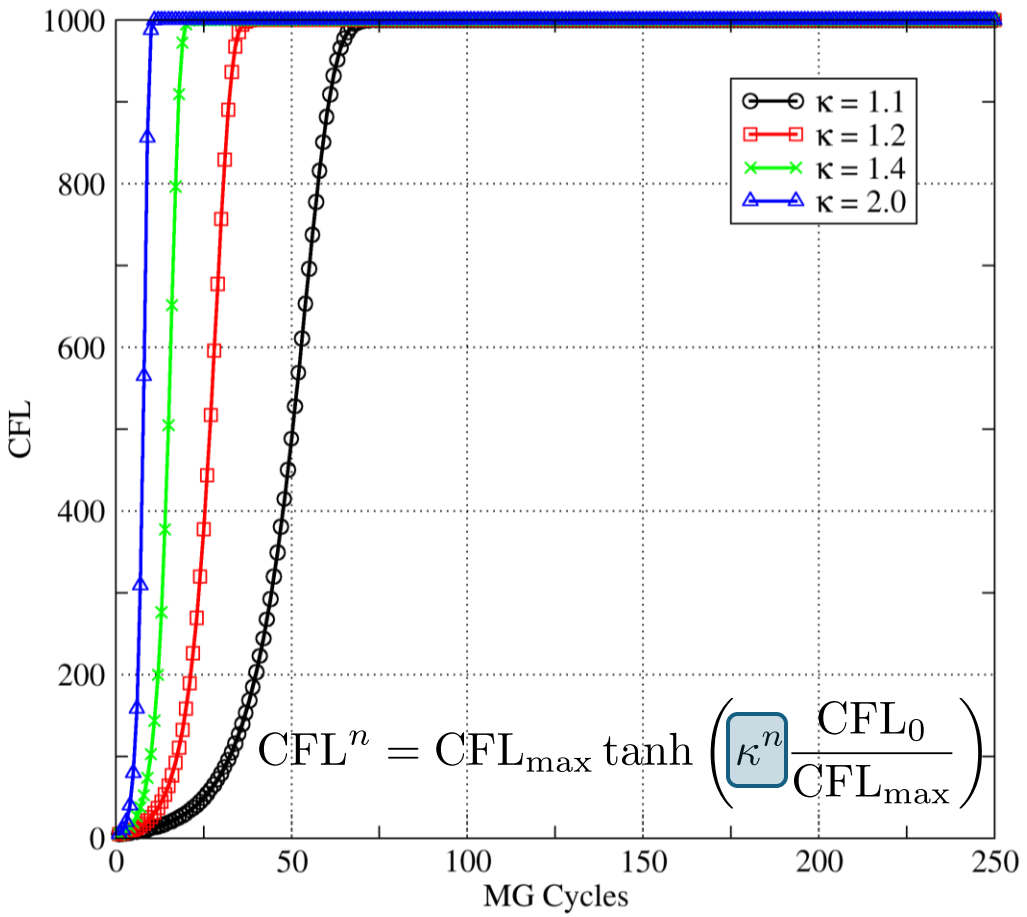
Initial CFL

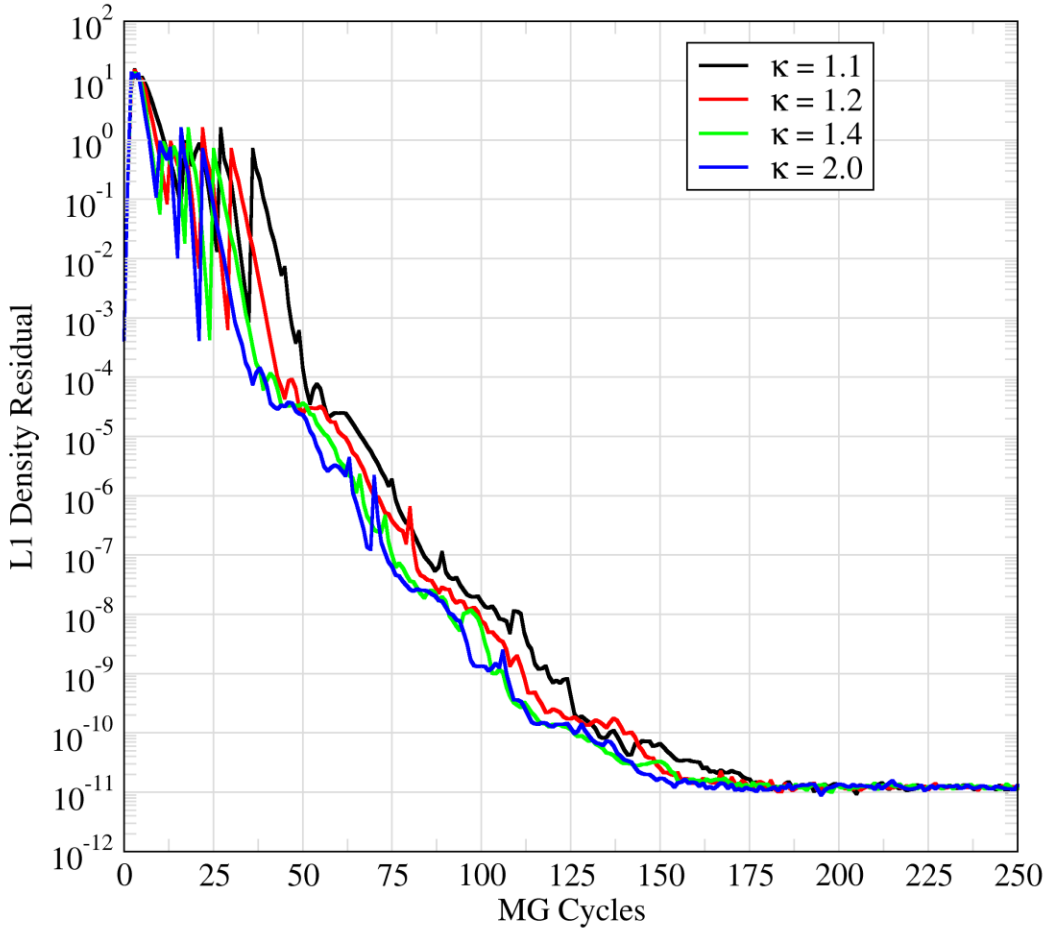


Maximum CFL

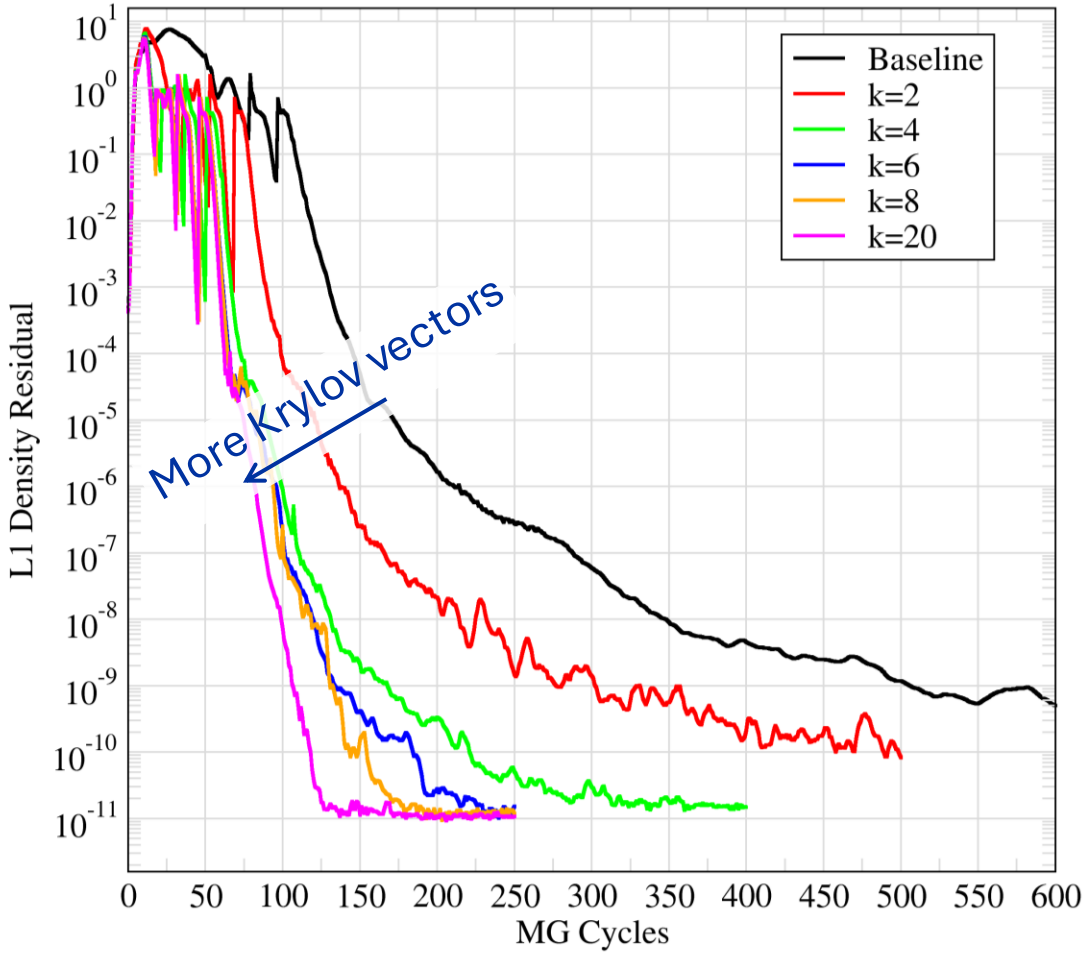


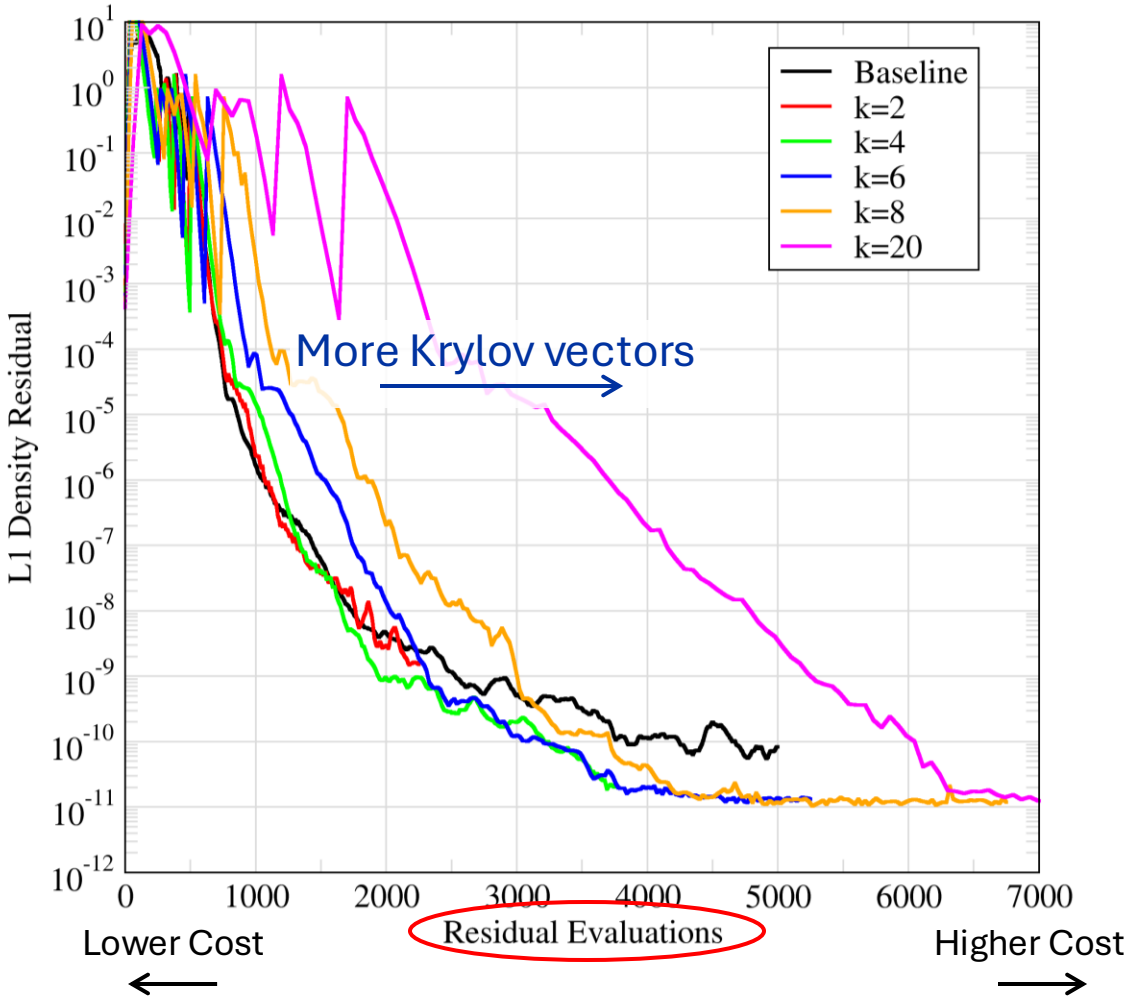
CFL ramping also has a small effect on the preconditioned method convergence rate





Larger Krylov subspaces improve the multigrid convergence rate but increase computational cost



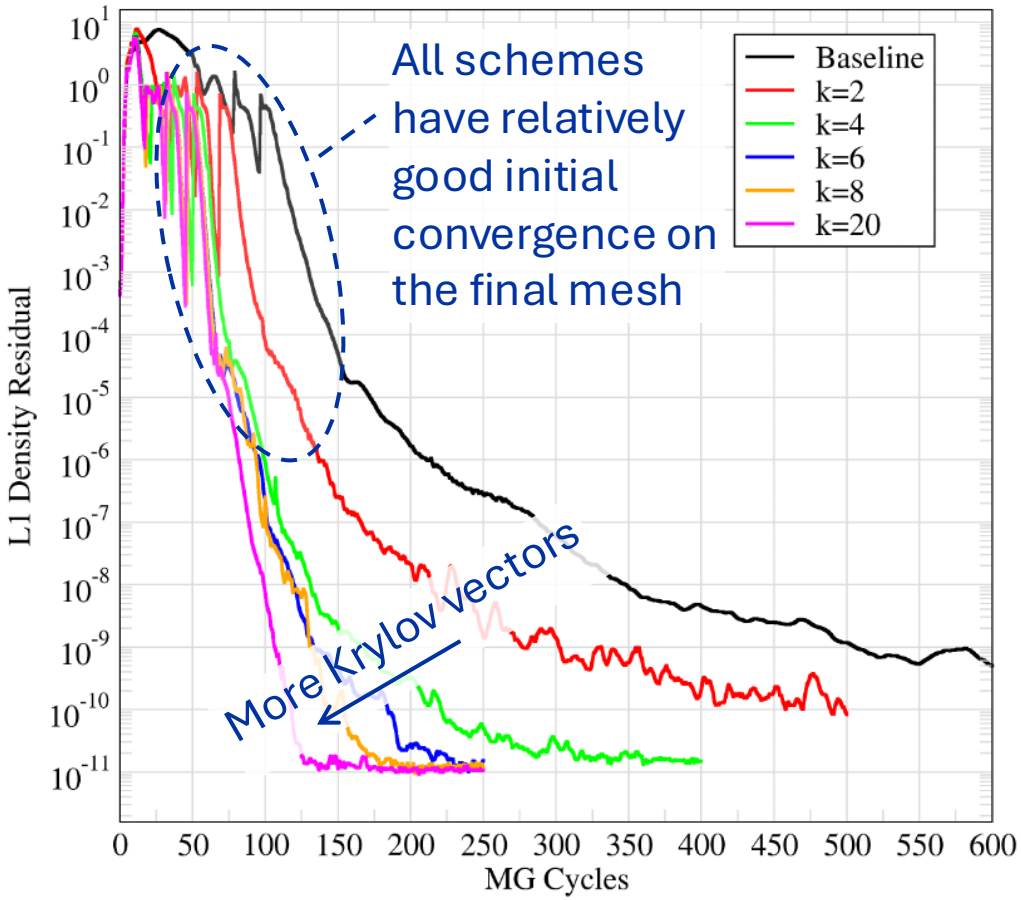


Capitalize on multigrid's fast initial convergence with sequential hybrid preconditioning

Baseline multigrid solver has good initial convergence rate

Faster convergence rate is maintained longer with preconditioned solver

Idea: start with explicit multigrid before enabling the implicit preconditioning



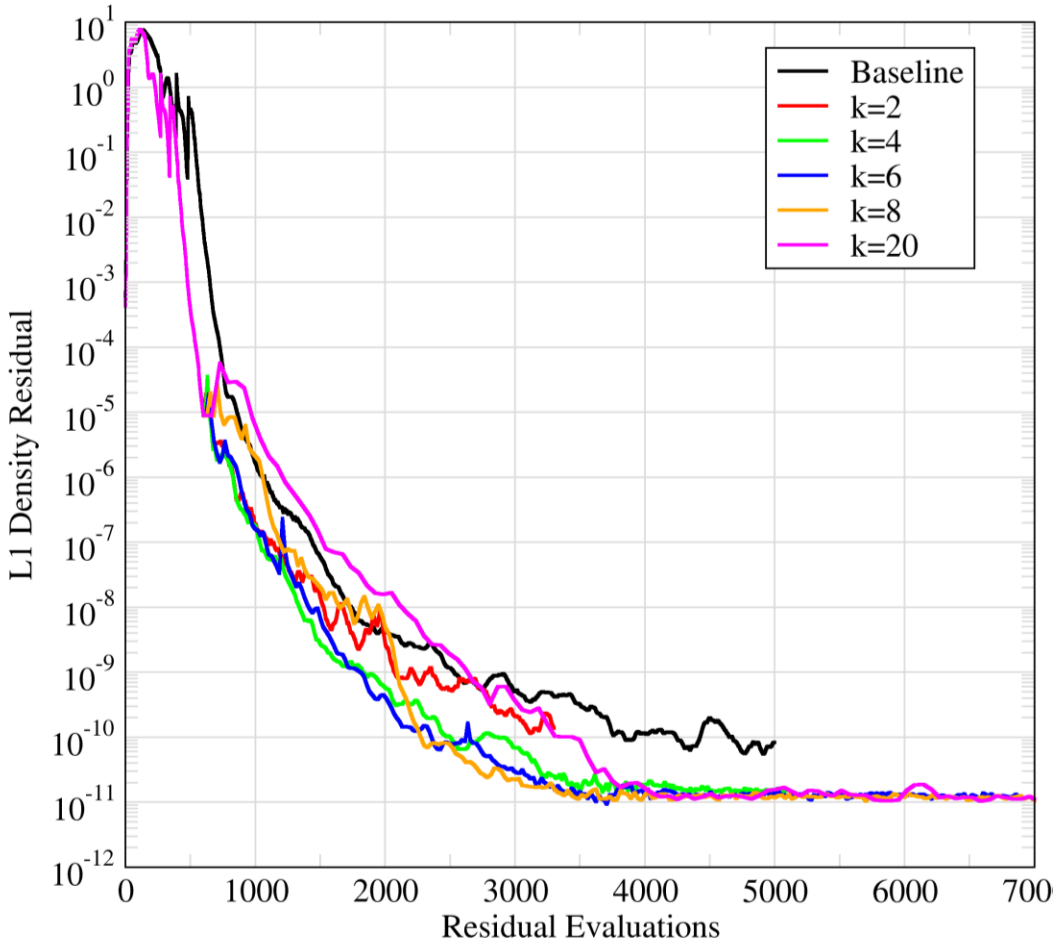
Capitalize on multigrid's fast initial convergence with sequential hybrid preconditioning

Idea: start with explicit multigrid before enabling the implicit preconditioning

- Run 200 iterations with explicit multigrid solver
- Turn on preconditioner with various # of Krylov vectors

Reduced computational cost of preconditioned algorithm

CFL constraint on initial startup



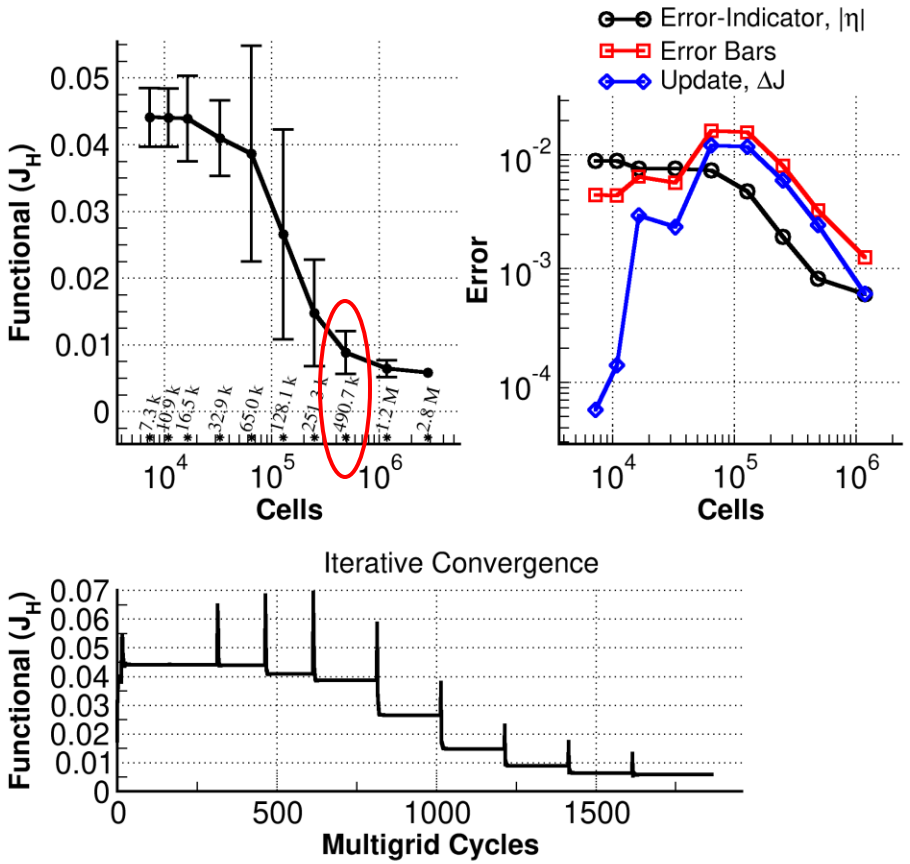
Investigate convergence on 3D case with more degrees of freedom: ONERA M6 wing

Nine adaptation cycles targeting drag coefficient

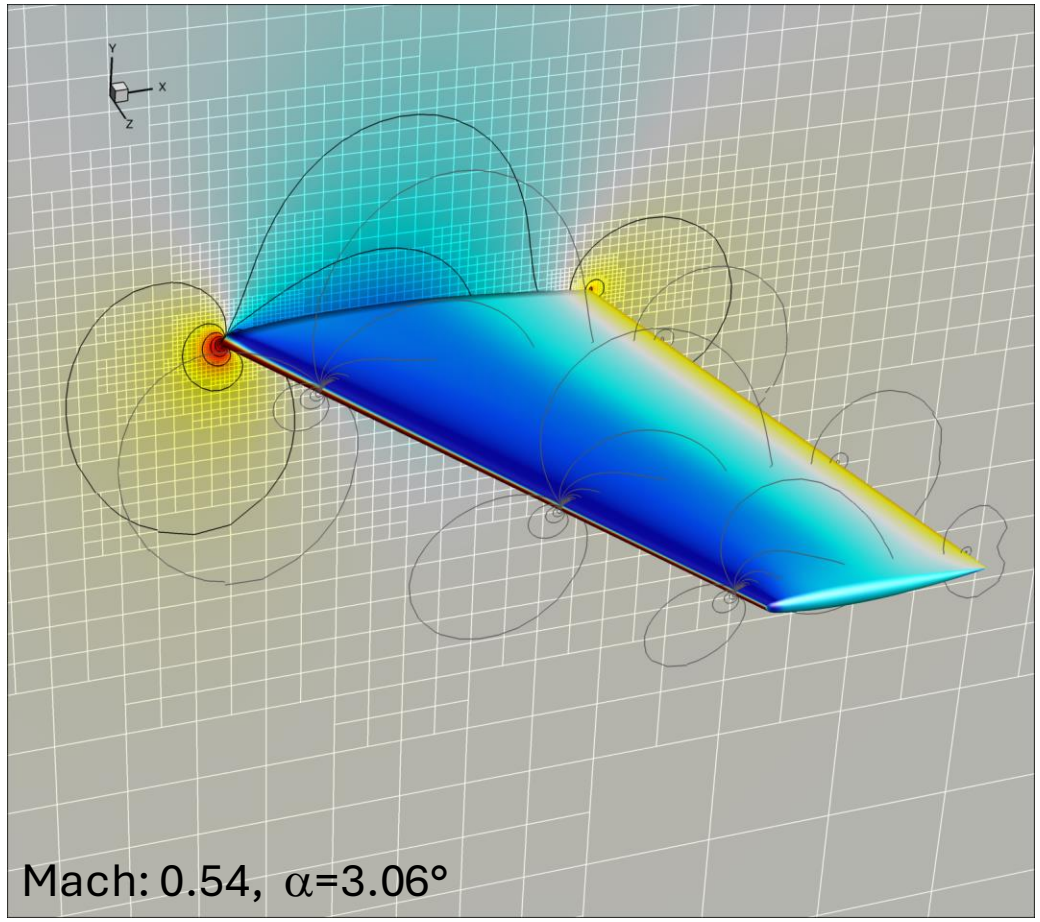
M_∞=0.54, α=3.06°

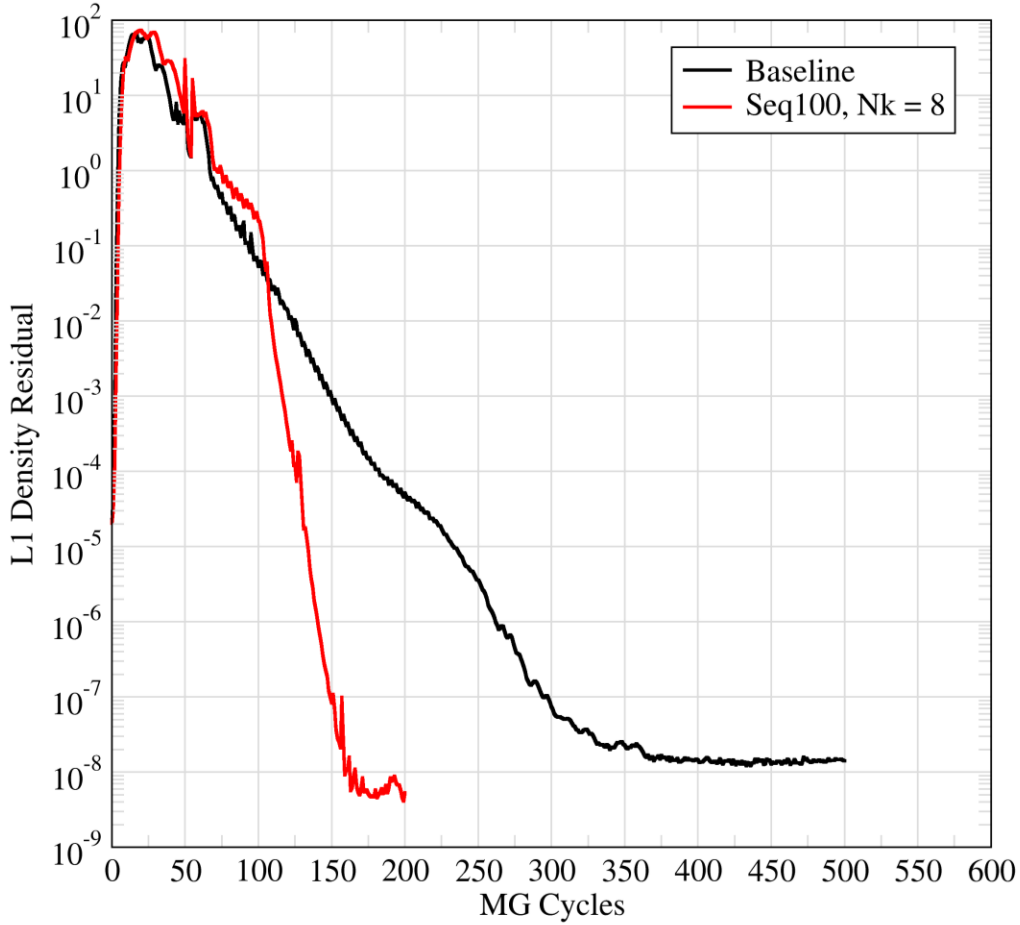
Use 7th adapted mesh to reduce computational cost

~500k cells (50x NACA 0012)



ONERA M6 wing also demonstrates improved convergence rate with preconditioning



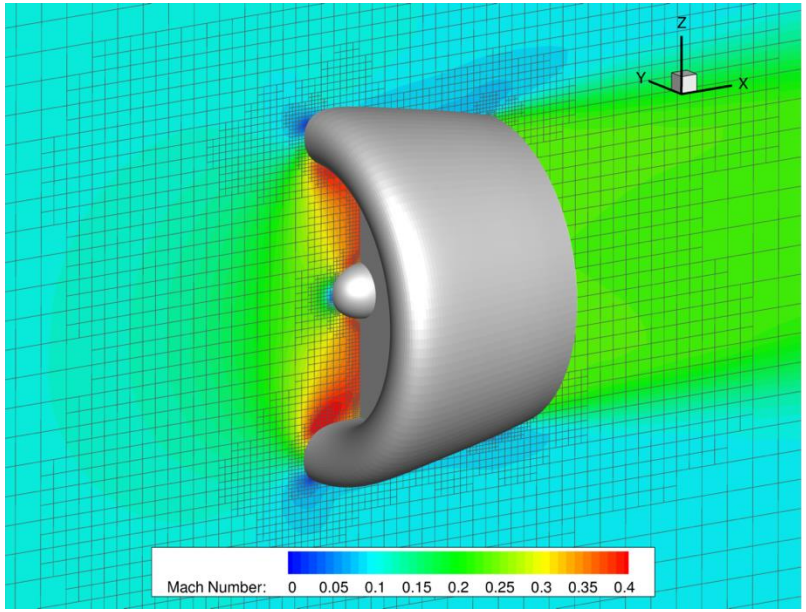


Explore ducted fan example with more stiffness from the power boundary conditions

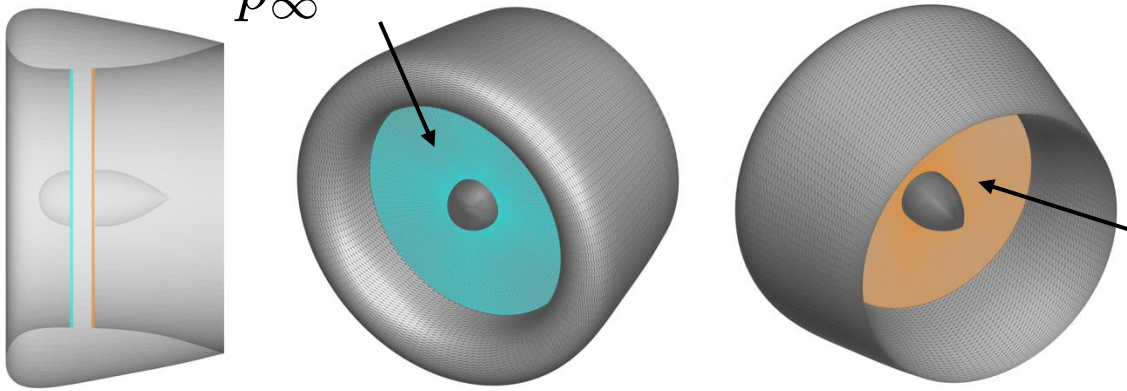
M_∞=0.1, α=10°

Six adaptation cycles
targeting duct thrust and lift

131k cells

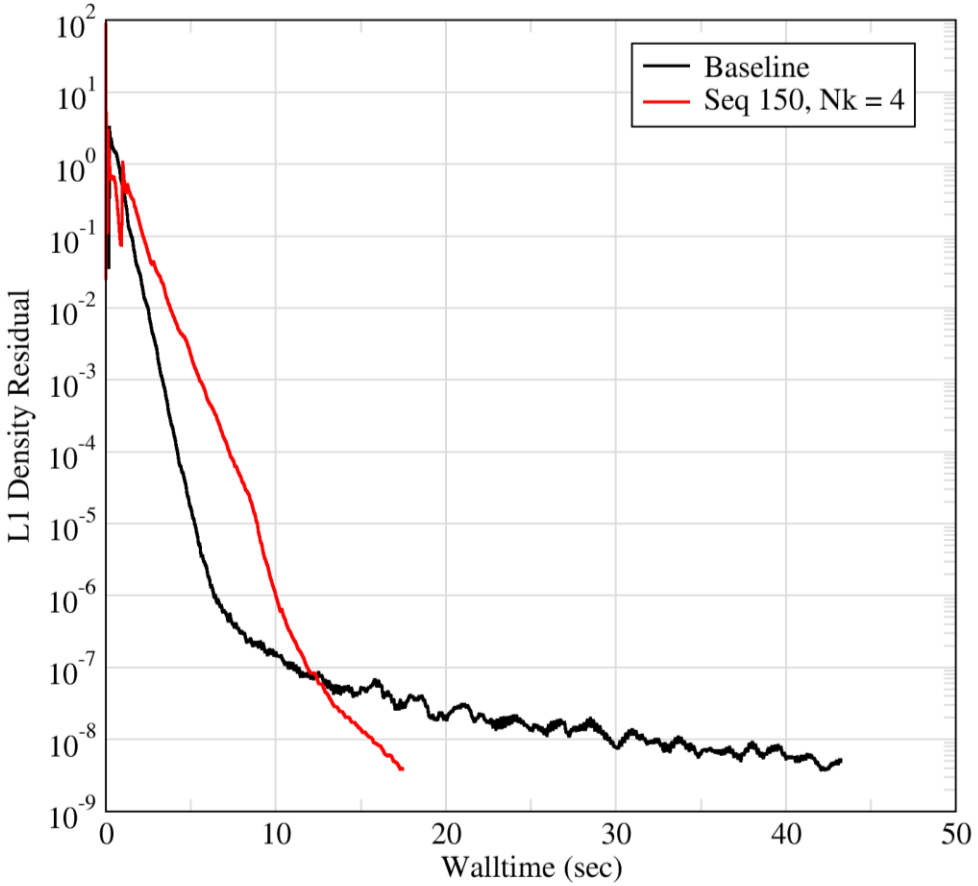


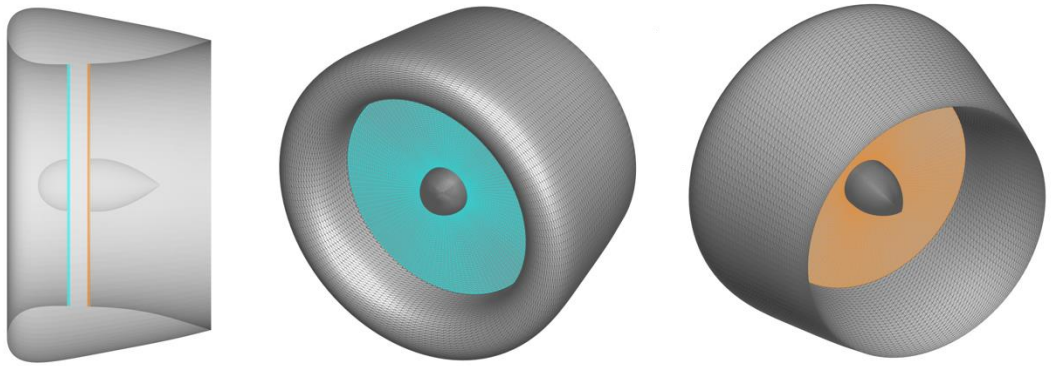
$$\frac{p}{p_\infty} = 0.91$$

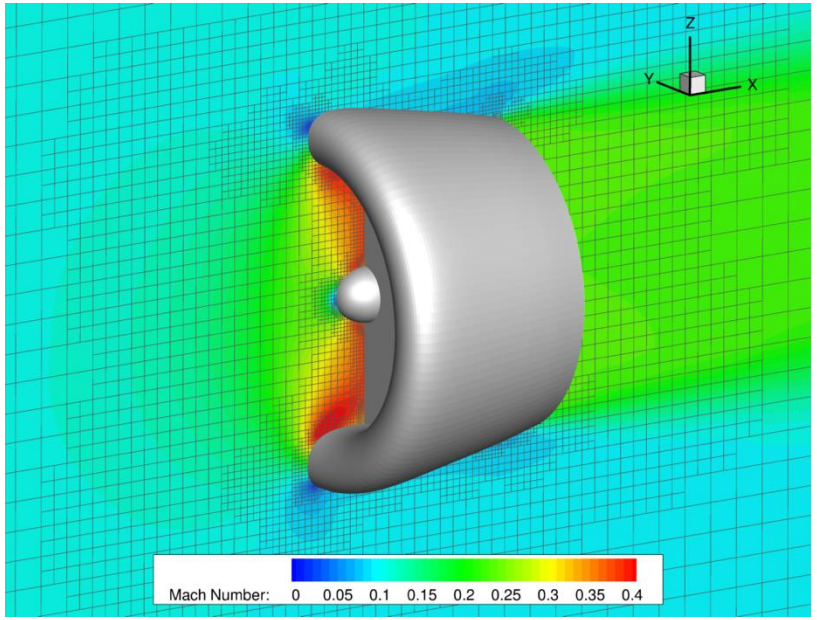


$$\frac{p_t}{p_{t,\infty}} = 1.03, \quad \frac{T_t}{T_{t,\infty}} = 1.01$$

Implicit preconditioner enables [1,0] V-cycles that show competitive time to solution







Implicit preconditioning strengthens an existing multigrid steady-state solver

Significantly improved convergence rates on a variety of 2D and 3D example problems

Jacobian-free Krylov subspace method is a **low memory, matrix-free** algorithm for solving the preconditioning equations

Most free parameters have small effects on the solver except for the **number of Krylov vectors**

Another use case: employ the stronger steady-state solver for high-order time integration via dual time stepping

Strong steady-state solvers enable implicit high-order time integration via dual time stepping

Implicit methods are advantageous since cut-cells can have **arbitrarily-small volumes**

A-stable (or L-stable) linear multistep methods are commonly used but **cannot exceed 2nd-order** accuracy

Multi-stage methods can be A- or L-stable and better than 2nd order accurate but often require deep convergence

Explore the potential of a **matrix-free** unsteady implementation with **fully-implicit Runge-Kutta** methods

A brief review of implicit Runge-Kutta methods with dual time stepping

Consider the model equation: $\dot{u} = \lambda u, \ \lambda \in \mathbb{C}$

Single-stage, 1st order, implicit RK method (implicit Euler):

$$\mathcal{D}_\tau u = \frac{u^n - u}{\Delta t} + \mathbf{A}\lambda u, \qquad \mathbf{A} = |\,1\,|$$

Now consider a multi-stage scheme

$$\mathcal{D}_\tau \overrightarrow{u_s} = \frac{\overrightarrow{u^n} - \overrightarrow{u_s}}{\Delta t} + \mathbf{A}\lambda \overrightarrow{u_s}$$

$$A = \left| \begin{array}{cc} \sqrt{2}/2 & 0 \\ 1\text{-}\sqrt{2}/2 & \sqrt{2}/2 \end{array} \right|$$

SDIRK2 (2nd-order accurate)

A brief review of implicit Runge-Kutta methods with dual time stepping

Consider the model equation: $\dot{u} = \lambda u, \ \lambda \in \mathbb{C}$

Single-stage, 1st order, implicit RK method (implicit Euler):

$$\mathcal{D}_\tau u = \frac{u^n - u}{\Delta t} + \mathbf{A}\lambda u, \qquad \mathbf{A} = |\,1\,|$$

Now consider a multi-stage scheme

$$\mathcal{D}_\tau \overrightarrow{u_s} = \frac{\overrightarrow{u^n} - \overrightarrow{u_s}}{\Delta t} + \mathbf{A}\lambda \overrightarrow{u_s}$$

$$A = \left| \begin{array}{cc} 5/12 & -1/12 \\ 3/4 & 1/4 \end{array} \right|$$

$\text{Radau IIA (3rd-order accurate)}$

A^{-1}-preconditioning stabilizes the pseudo-time iterations for some fully-implicit RK methods

Naïve dual time stepping is provably unstable for small Δt for various fully-implicit Runge-Kutta methods

$$\mathcal{D}_\tau \overrightarrow{u_s} = \frac{\overrightarrow{u^n} - \overrightarrow{u_s}}{\Delta t} + \mathbf{A}\lambda\overrightarrow{u_s}$$

A^{-1}-preconditioning:

$$\mathcal{D}_\tau \overrightarrow{u_s} = \boxed{\mathbf{A}^{-1}} \left(\frac{\overrightarrow{u^n} - \overrightarrow{u_s}}{\Delta t} + \mathbf{A}\lambda\overrightarrow{u_s} \right)$$

Stable for SDIRK2, Radau IIA, and Gauss-Legendre time integration

[Jameson, 2017]

A^{-1}-preconditioning simplifies the implementation by diagonalizing the stage equations

A^{-1}-preconditioning:

$$\mathcal{D}_\tau \overrightarrow{u_s} = \mathbf{A}^{-1}\left(\frac{\overrightarrow{u^n} - \overrightarrow{u_s}}{\Delta t} + \mathbf{A}\lambda\overrightarrow{u_s}\right)$$

$$= \mathbf{A}^{-1}\left(\frac{\overrightarrow{u^n} - \overrightarrow{u_s}}{\Delta t}\right) + \boxed{\mathbf{I}\lambda\overrightarrow{u_s}}$$

Decoupled residuals reduces the memory footprint

Diagonalized stage equations enable simple inclusion of BDF schemes into formulation

Can easily include classical backwards difference methods

$$\mathcal{D}_\tau \overrightarrow{u_s} = \mathbf{A}^{-1}\left(\frac{\overrightarrow{u^n} - \overrightarrow{u_s}}{\Delta t}\right) + \mathbf{I}\lambda\overrightarrow{u_s}$$

Solve one stage equation but leverage other stages for storage of solutions at previous timesteps

Currently use a two-stage formulation that includes BDF1 and BDF2

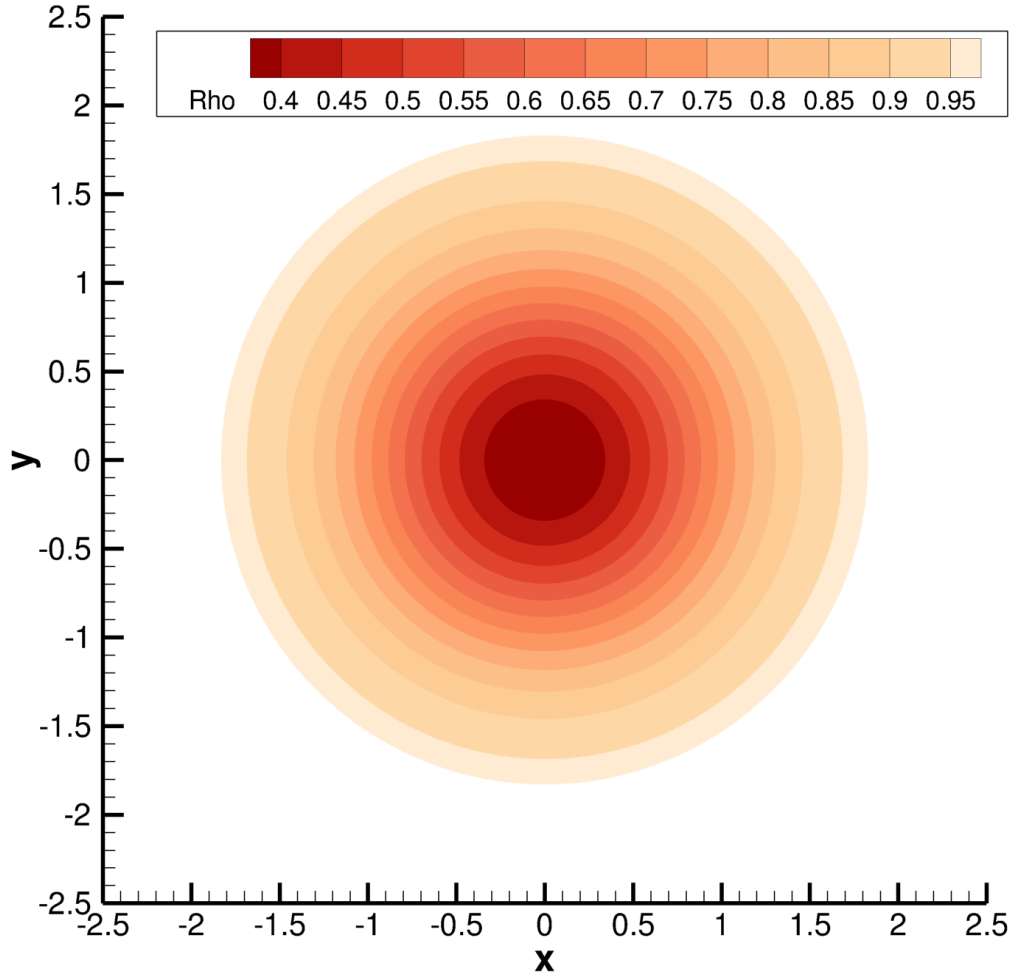
$$A = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}$$

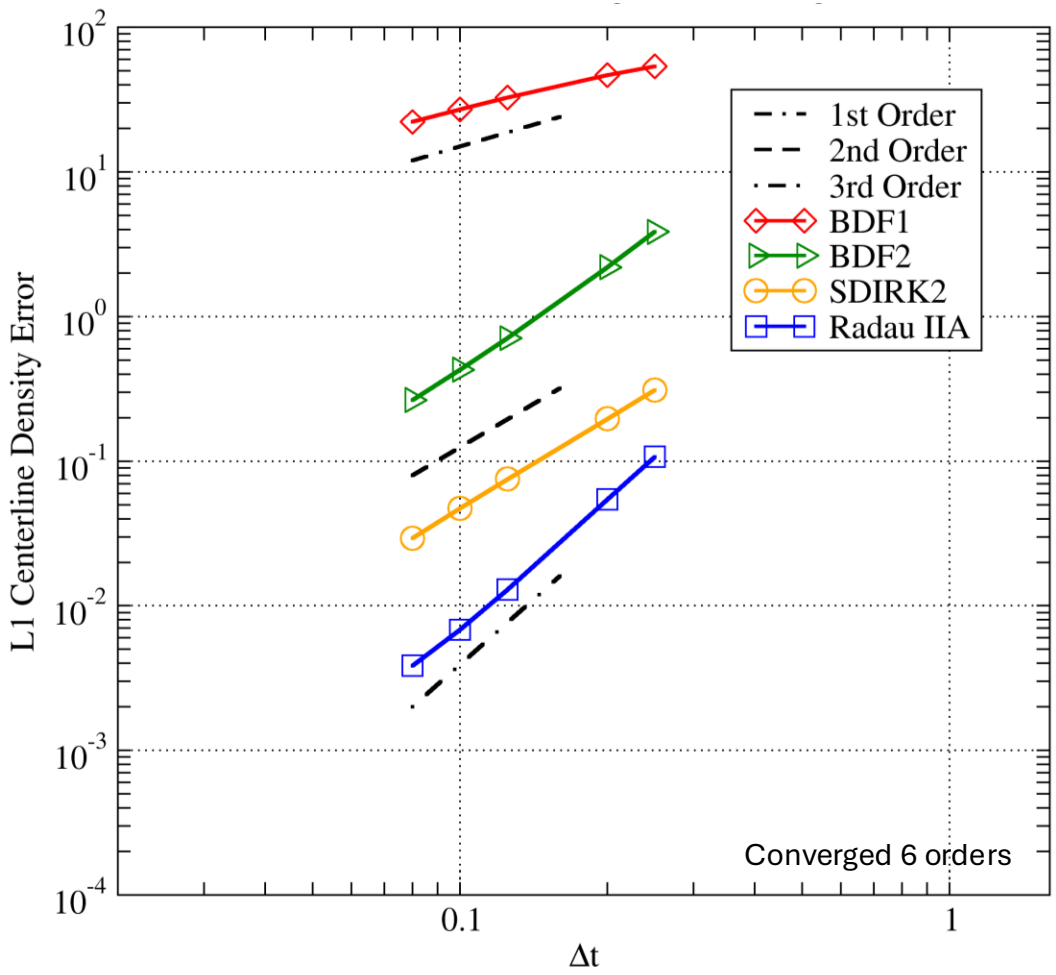
BDF1 (1st-order accurate)

$$A = \begin{vmatrix} 1/2 & -1/2 \\ 1/2 & 3/2 \end{vmatrix}$$

BDF2 (2nd-order accurate)

This formulation demonstrates proper order of convergence on 2D vortex convection





Summary

Significantly improved convergence rates on a variety of 2D and 3D example problems with preconditioned multigrid solver

Jacobian-free Krylov subspace method is a **low memory, matrix-free** algorithm for solving the preconditioning equations

Most free parameters have small effects on the solver except for the **number of Krylov vectors**

Described and verified a **matrix-free dual time stepping** implementation that includes both **fully-implicit Runge-Kutta** and backwards difference (BDF) methods

Outlook

Implement the implicit preconditioning in the **RANS solver** and assess solver convergence rates

Explore **alternative formulations** of the Jacobian-vector product

Investigate JFNK smoother with preconditioned **FGMRES** and multigrid preconditioner

Assess convergence criteria and efficiency of implicit RK methods on **more complex unsteady cases**

Acknowledgments

ARMD's Transformational Tools and Technologies (T^3) project provided funding for this work

Computer resources were provided by the NASA Advanced Supercomputing Division

Dr. Marian Nemec for helpful discussions about Jacobian-free implicit solvers

Implicit Preconditioning for Explicit Multigrid Solvers on Cut-Cell Cartesian Meshes

Questions?

