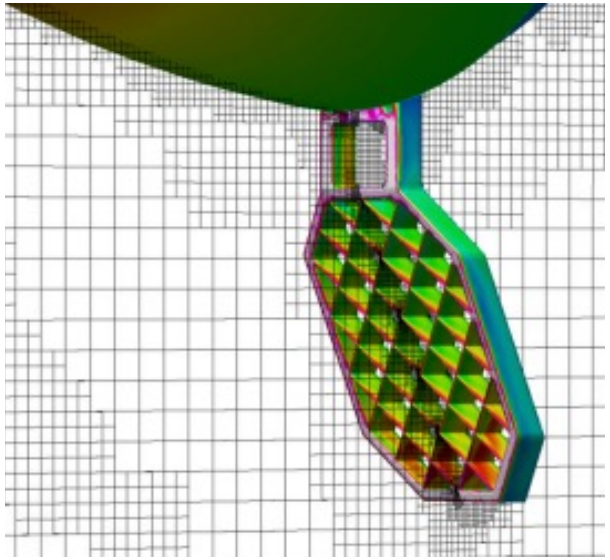
Implicit Preconditioning for Explicit Multigrid Solvers on Cut-Cell Cartesian Meshes

Jonathan Chiew & **Michael Aftosmis**
Computational Aerosciences Branch
Ames Research Center

ICCFD12 – Kobe, Japan
July 18, 2024

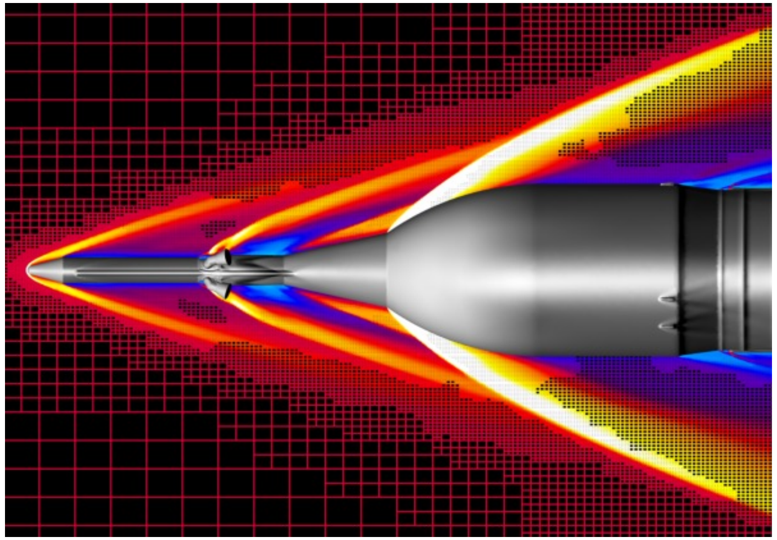
Cut-cell Cartesian meshes enable robust, automated meshing for CFD

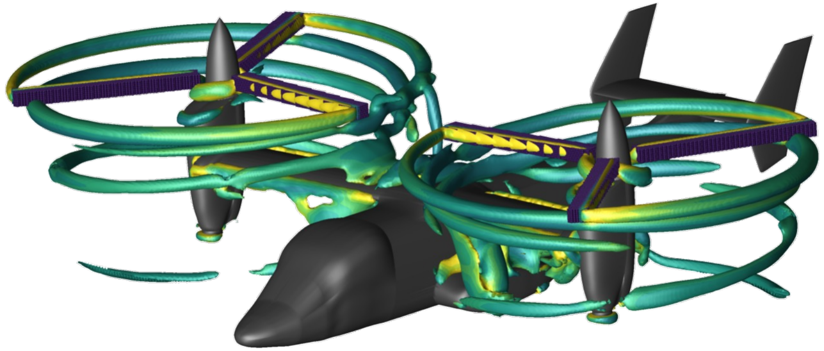


Automated mesh generation and adaptation are essential for vehicle design and optimization

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

Cart3D uses an **explicit multigrid solver** that is well suited for the 3D Euler equations



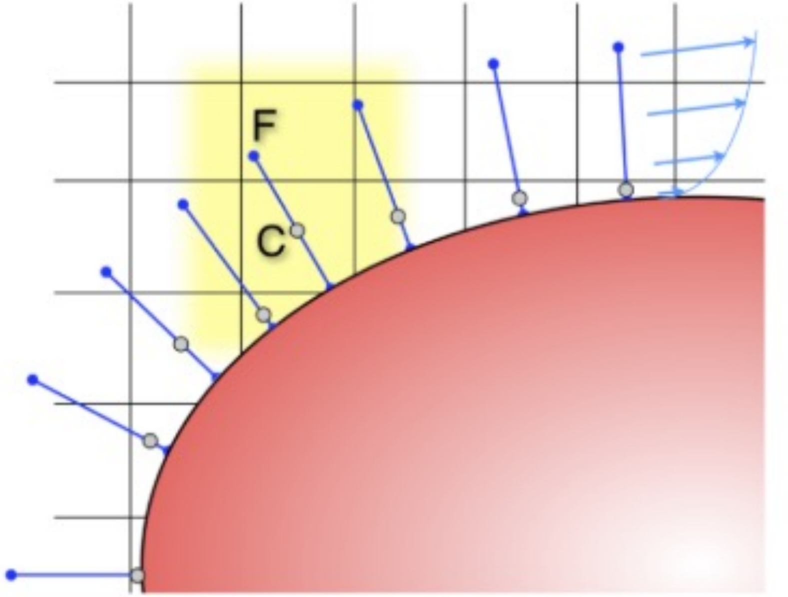


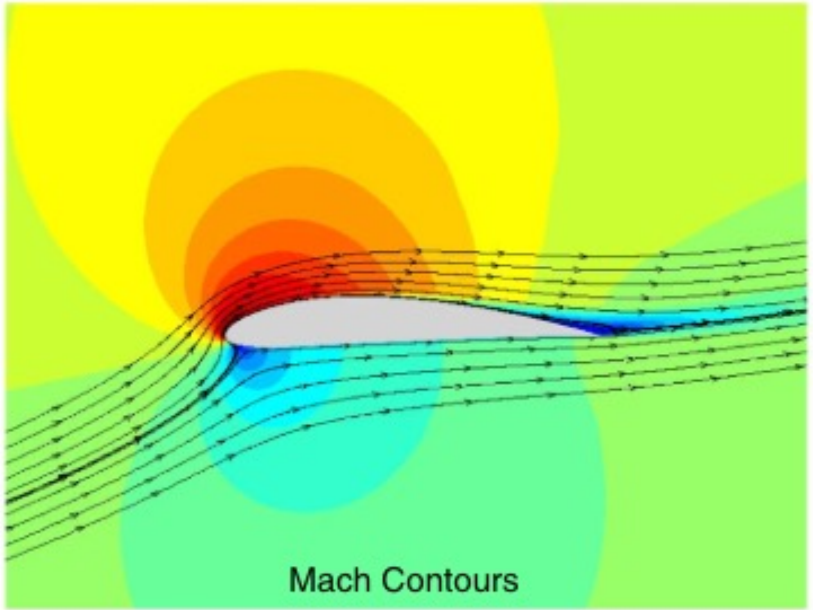
We want to strengthen the existing multigrid solver for an incipient automated RANS capability

Developing capability for **automated, RANS-based** vehicle design

Need a **stronger solver** for stiffness from the viscous equations

Implicit preconditioning to leverage the existing multigrid solver





[Berger & Aftosmis, 2017]

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

Baseline Explicit Multigrid Solver

Implicit Preconditioning

Steady-state Results

Dual Time Stepping Formulation

Time-dependent Results

The baseline multigrid algorithm is fully explicit and matrix-free

Commonly use **W-cycles** with **5-stage** smoother

Gradient evaluations on the **first stage only**

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

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

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

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

Improve the multigrid convergence rate with implicit preconditioning

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

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

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

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

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

Many approaches in literature including IRS, block-Jacobi, etc.

Use Rossow's [2007] global preconditioner based on implicit Euler method

Solve the linear preconditioner equations with Jacobian-free Newton Krylov method

Solve the implicit preconditioning equations

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

with Jacobian-free Newton Krylov (JFNK) using Fréchet derivatives

$$\frac{\partial \mathcal{R}}{\partial \mathbf{U}} v \approx \frac{\mathcal{R}\left(\mathbf{U} + \epsilon v\right) - \mathcal{R}\left(\mathbf{U}\right)}{\epsilon}$$

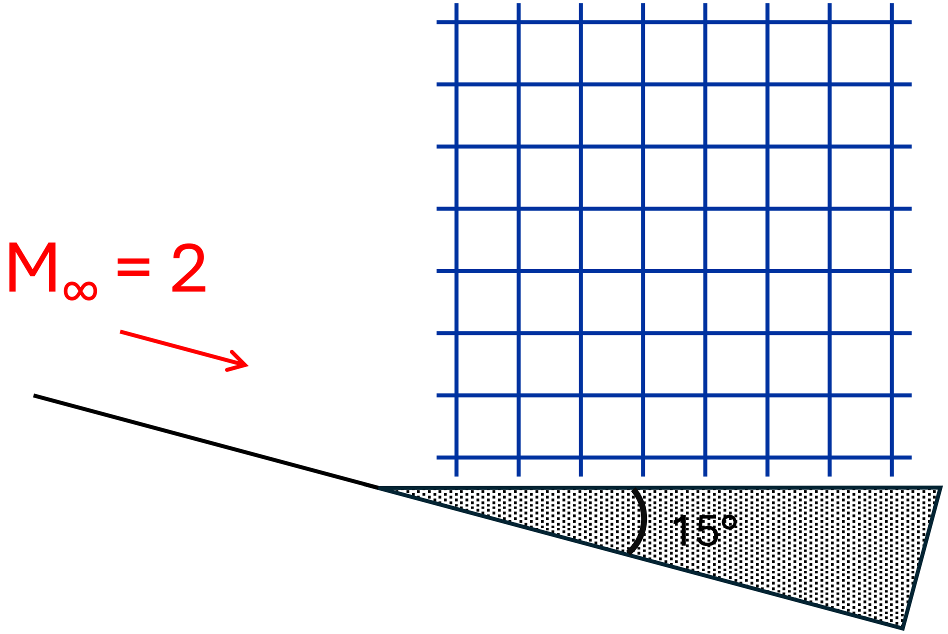
$$\epsilon = \sqrt{\epsilon_{\text{mach}}} \left\langle \mathbf{U}, v \right\rangle$$

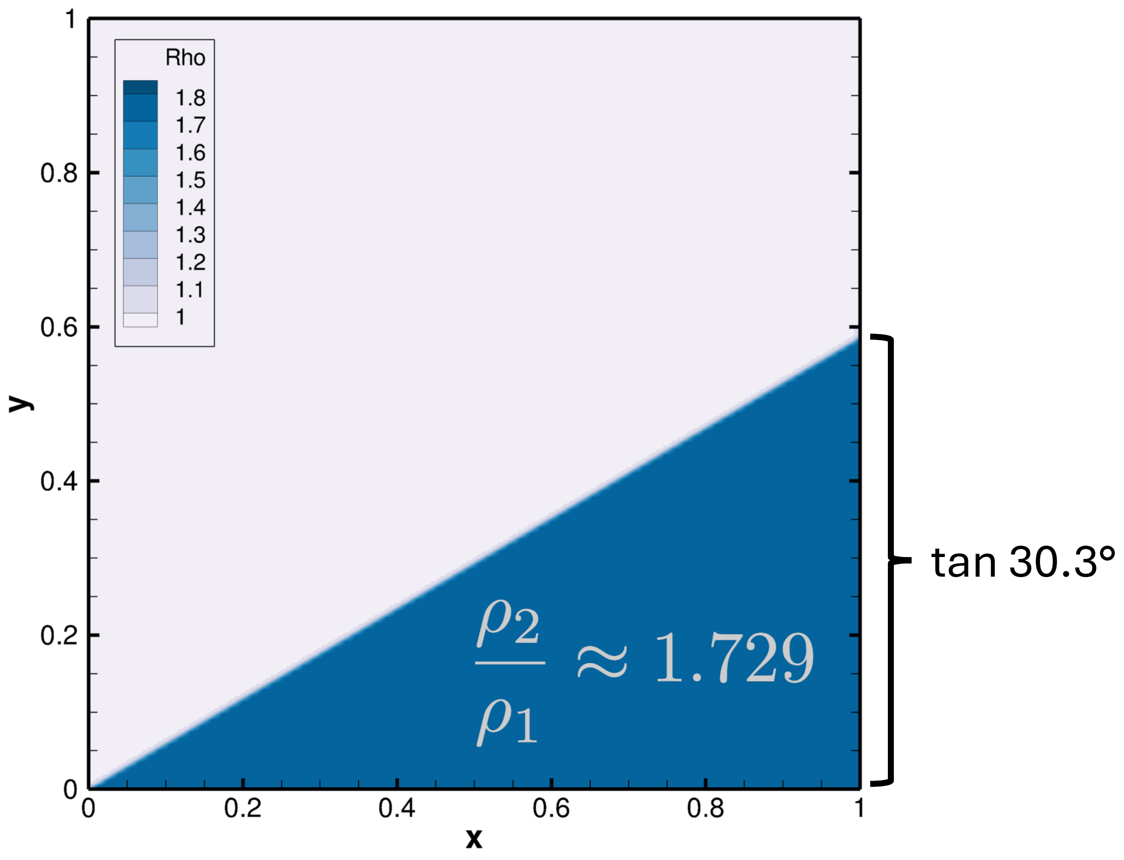
Supersonic wedge case for verification of the baseline and preconditioned solvers

Fully supersonic flow

Uniform 256x256 mesh

15° wedge aligned to grid

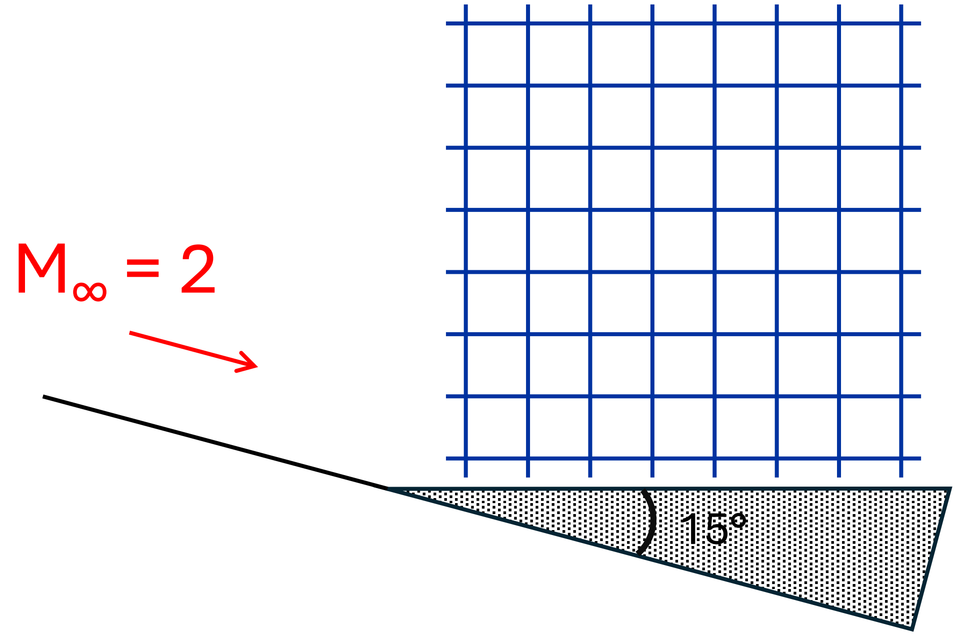


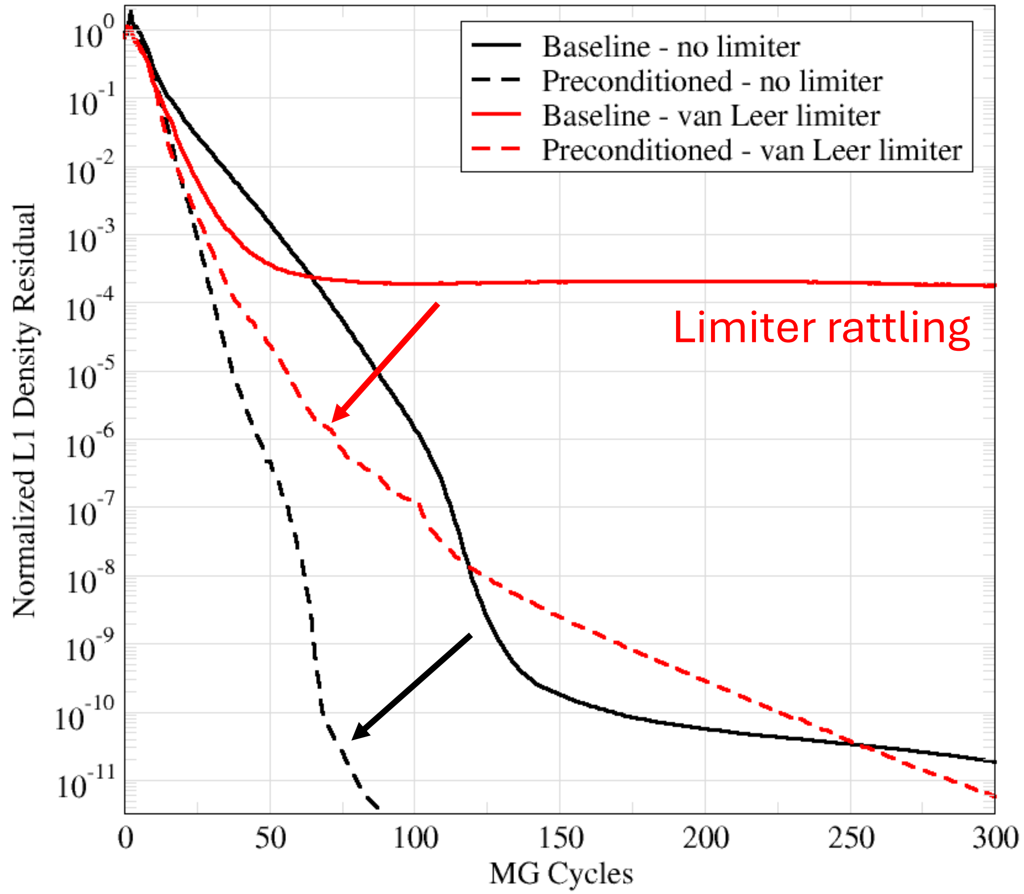


Preconditioned solver improves convergence rate on supersonic wedge case

8 Krylov vectors, max CFL = 1000

No limiter MG convergence rate improves from 0.88 → 0.74





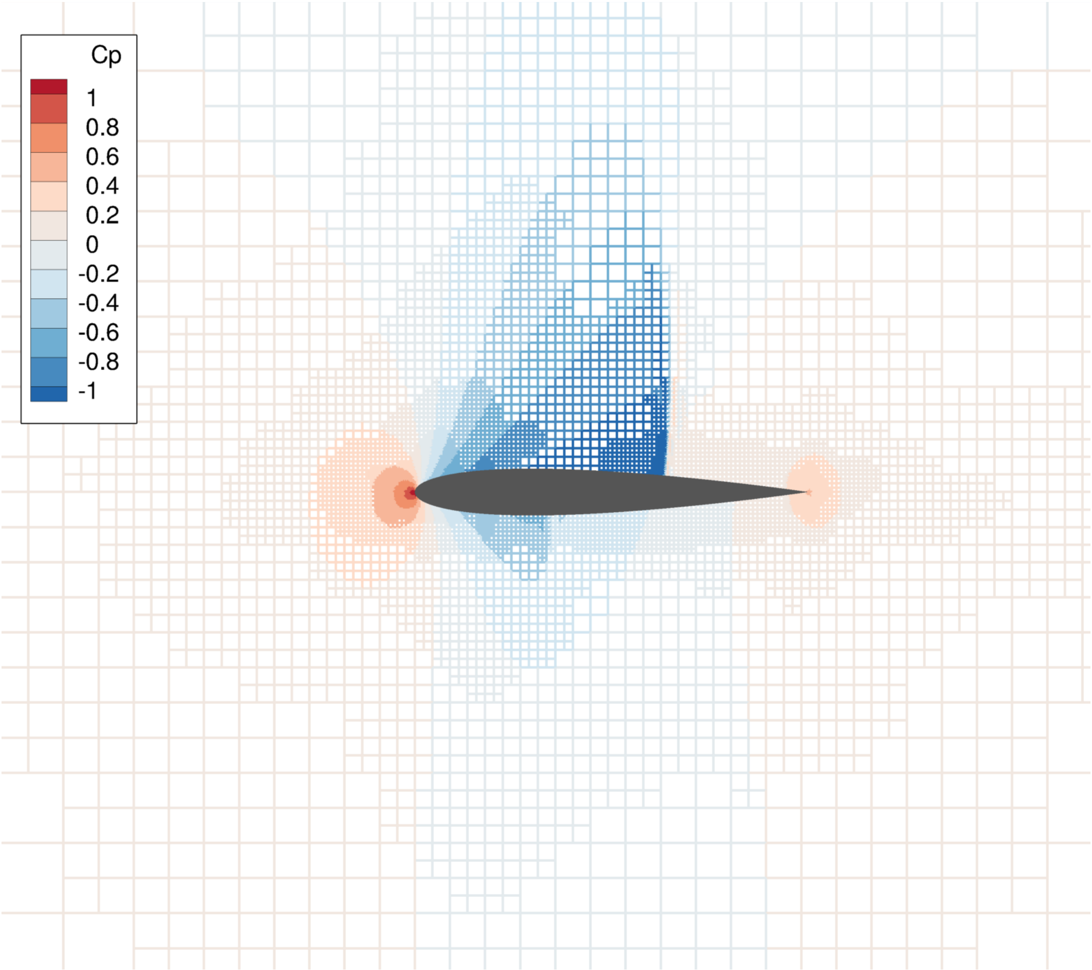
Further evaluation of the preconditioned solver with a more realistic case: NACA 0012 airfoil

NACA 0012 with closed TE

$M_\infty = 0.8$, $\alpha = 1.25°$

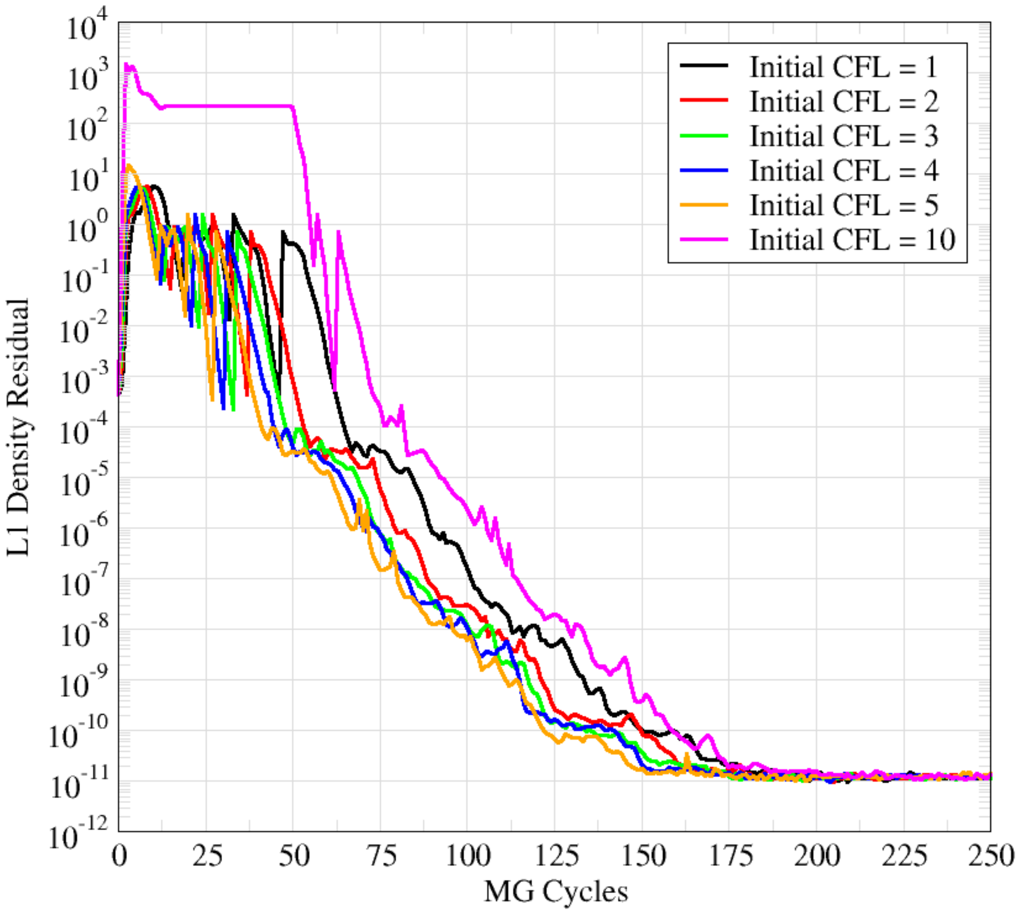
~10k cell mesh generated with 7 adaptation cycles

4 level [1,1] W-cycle

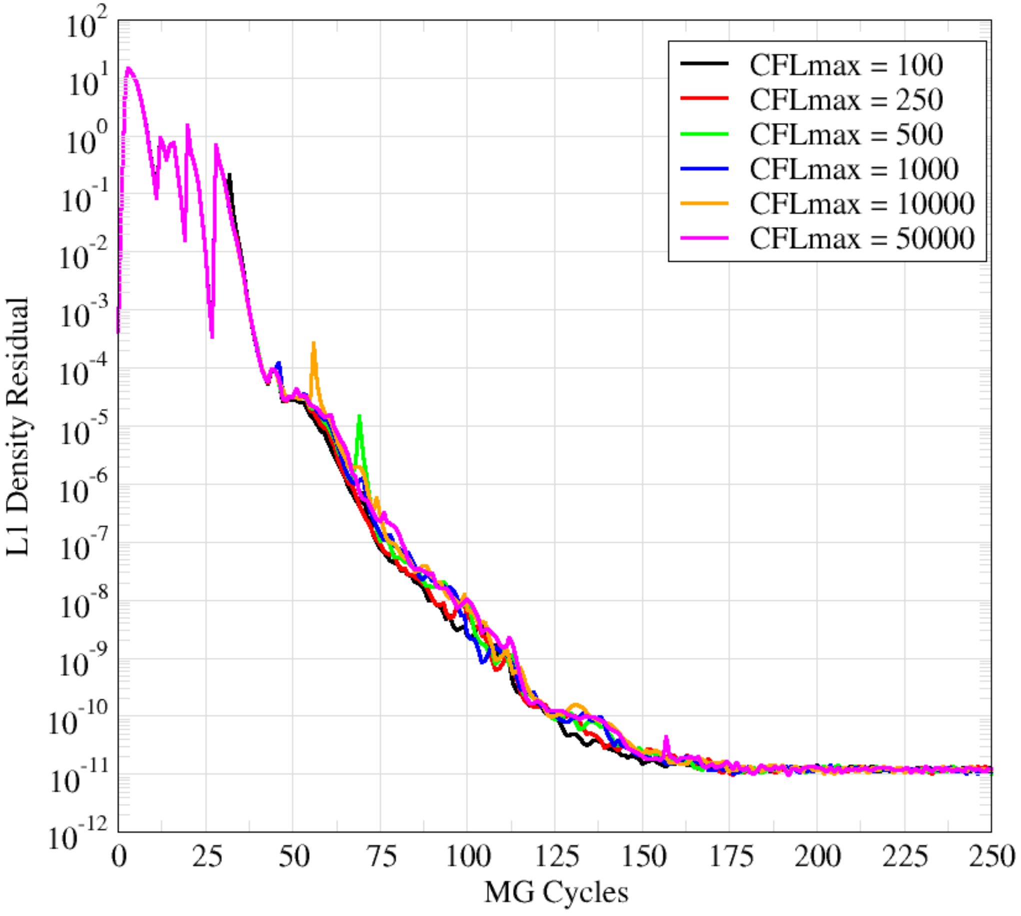


Most free parameters have a small effect on the preconditioned method convergence rate

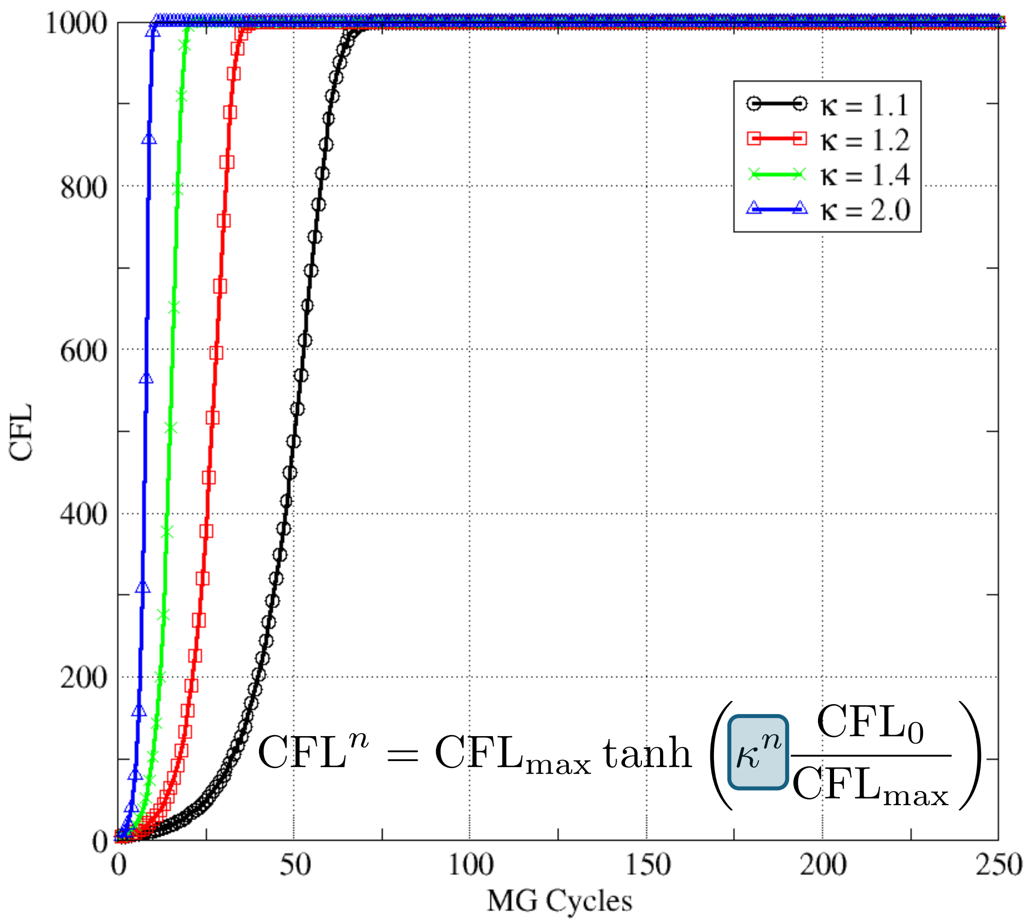
Initial CFL

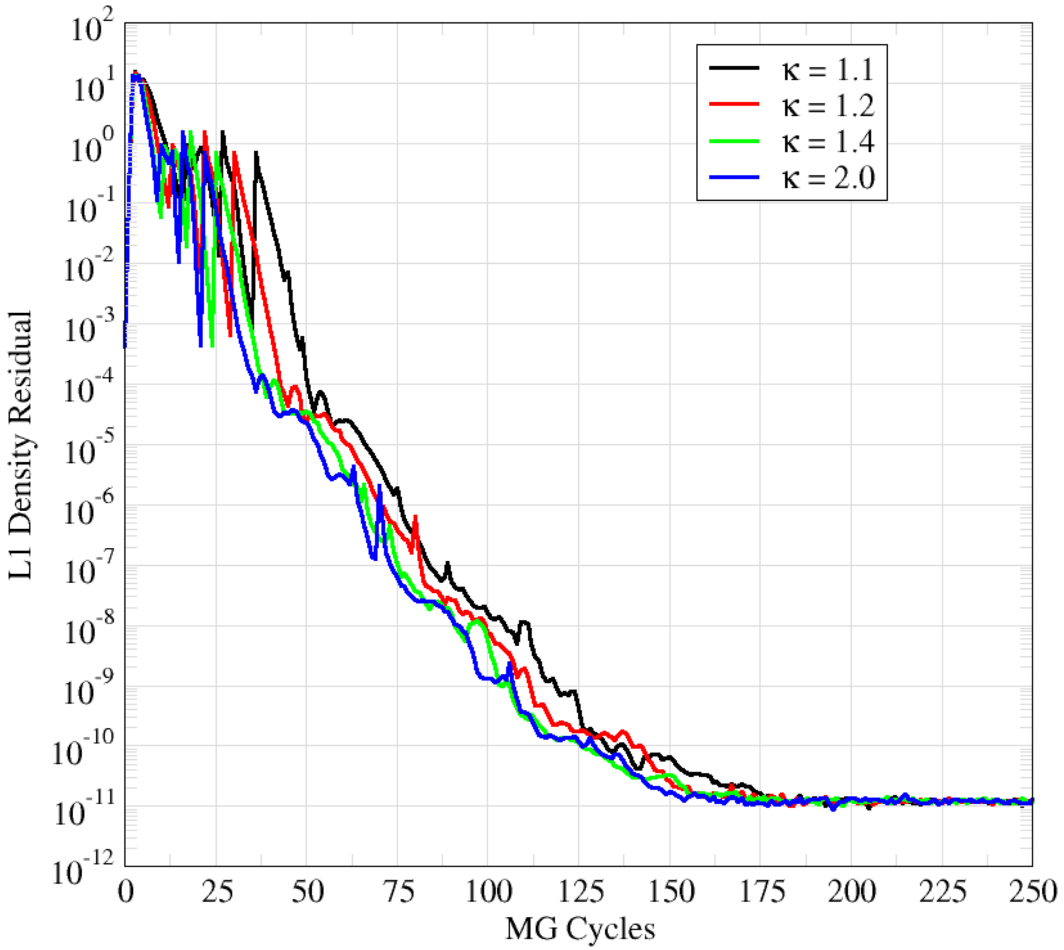


Maximum CFL

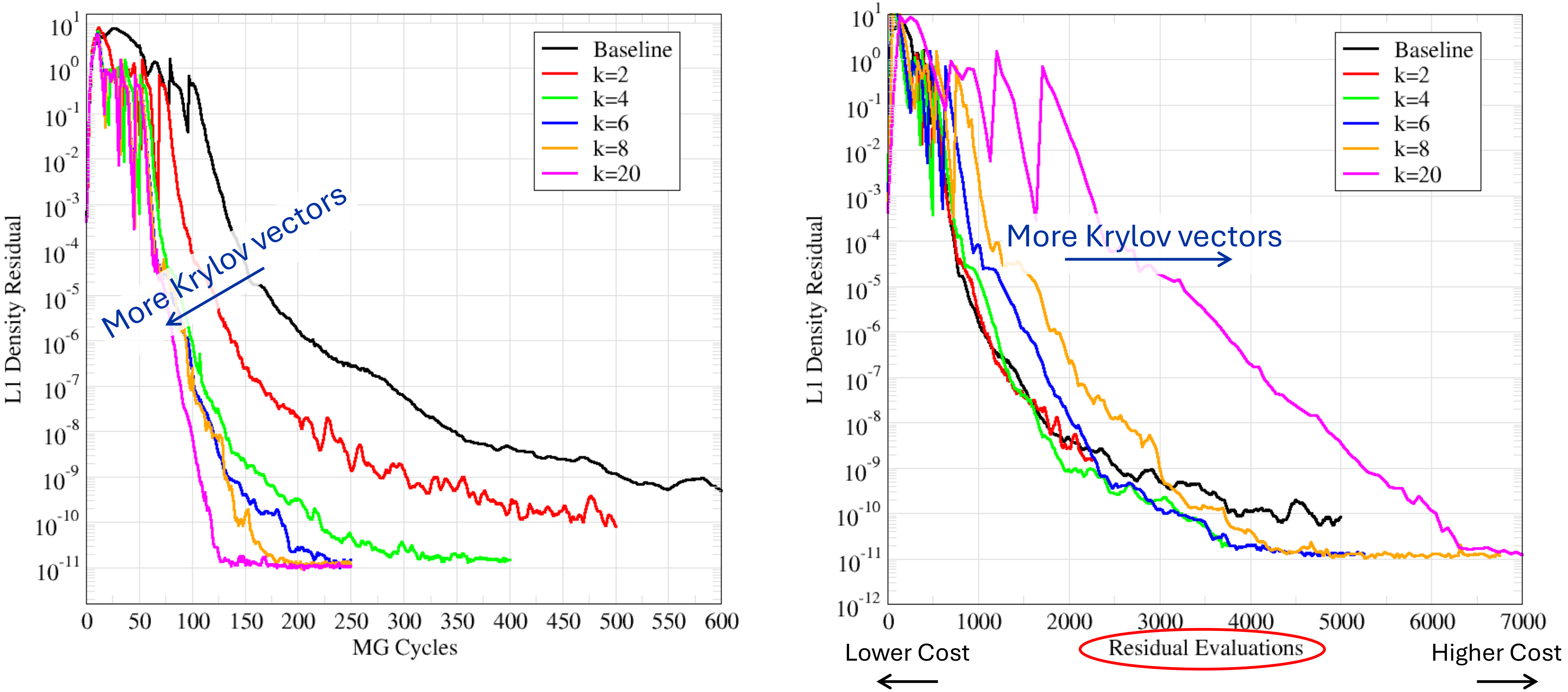


CFL ramping also has a small effect on the preconditioned method convergence rate





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

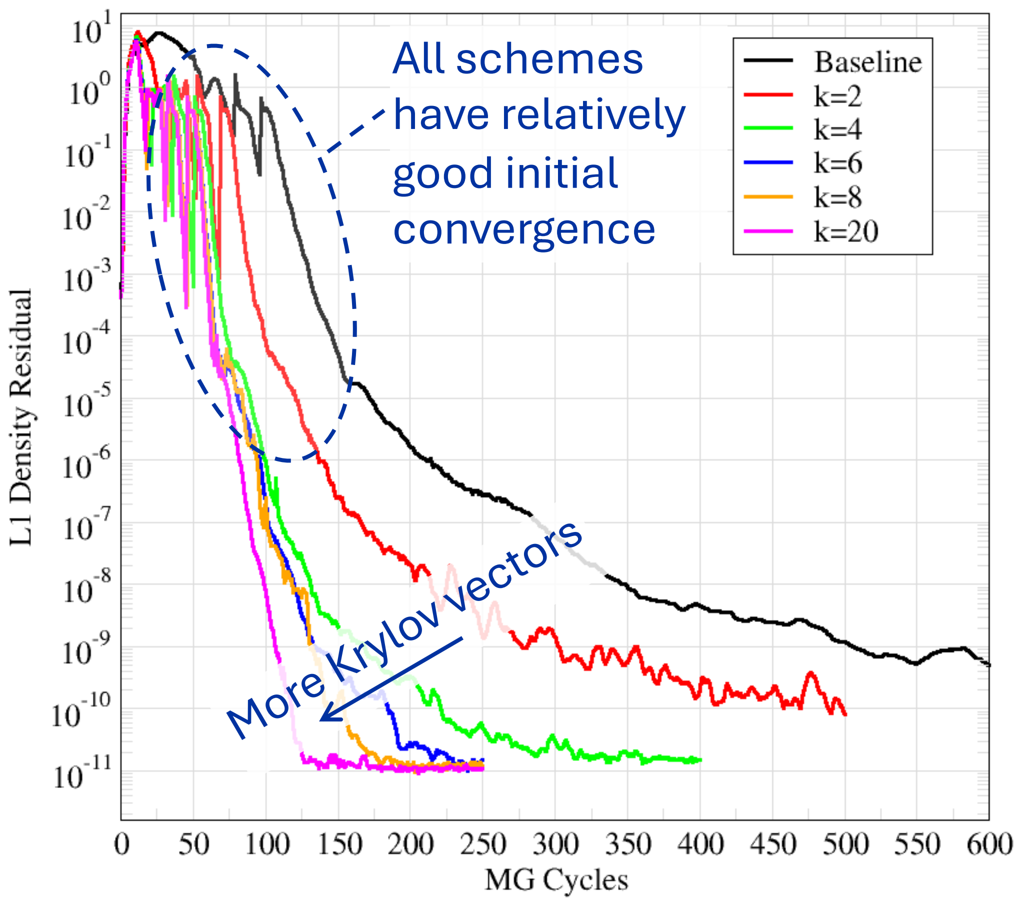


Capitalize on multigrid's fast initial convergence with sequential hybrid preconditioning

Baseline multigrid solver has good initial convergence rate

Faster convergence rate is also maintained longer with preconditioned solver

Idea: start with explicit multigrid before enabling the implicit preconditioning



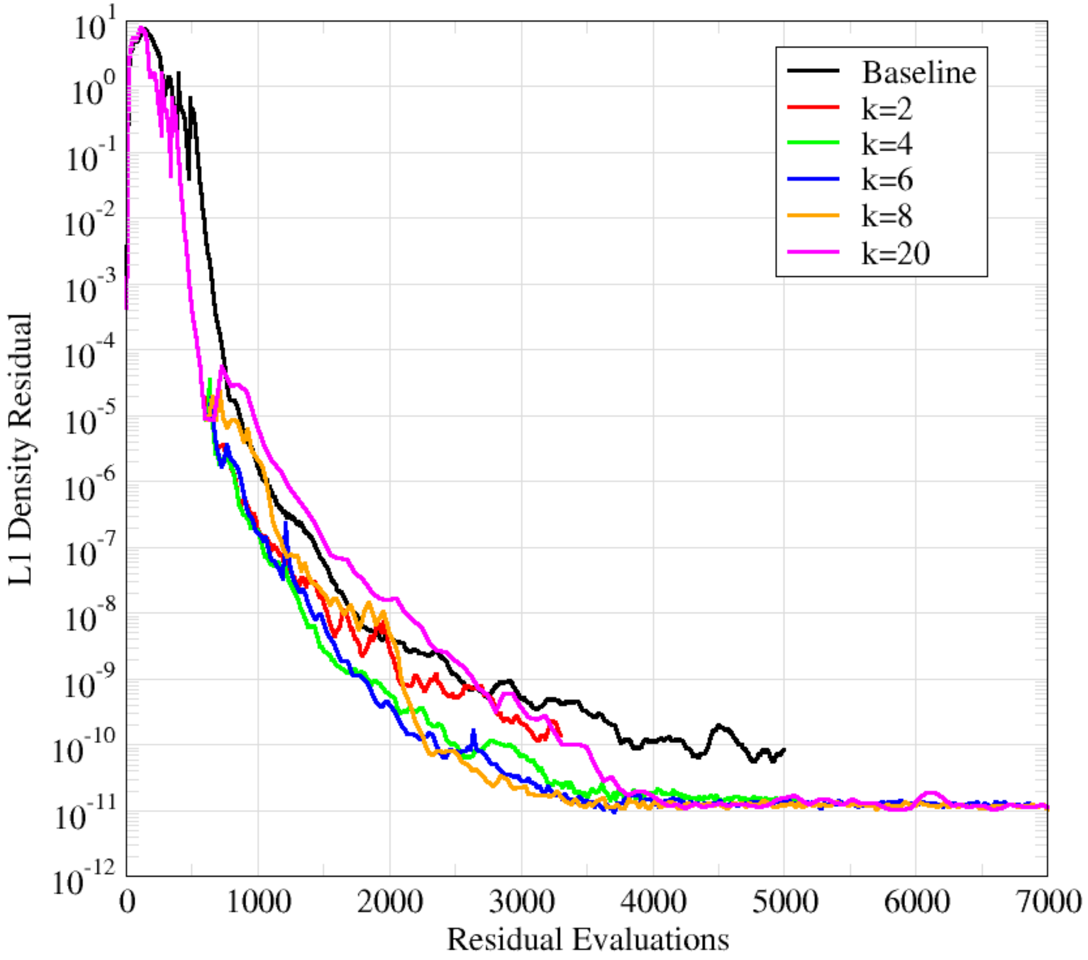
Capitalize on multigrid's fast initial convergence with sequential hybrid preconditioning

Idea: start with explicit multigrid before enabling the implicit preconditioning

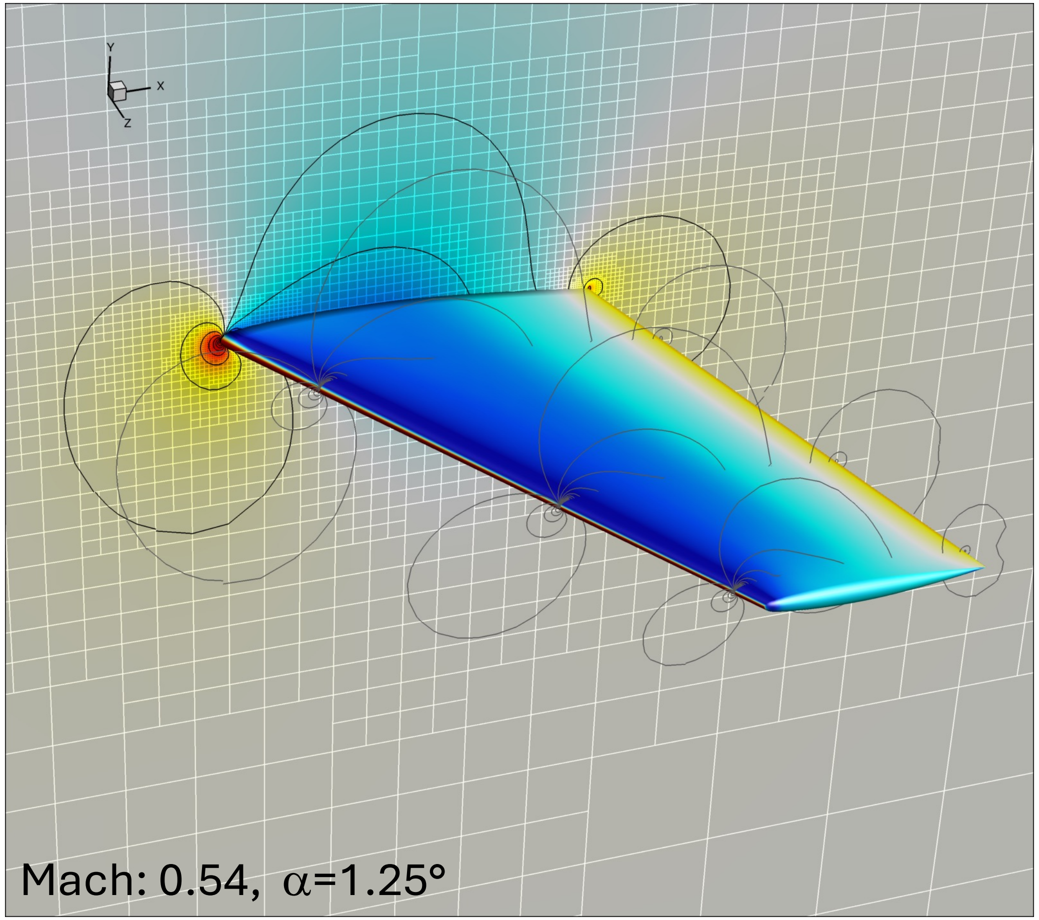
- Run 200 iterations with explicit multigrid solver
- Turn on preconditioner with various # of Krylov vectors

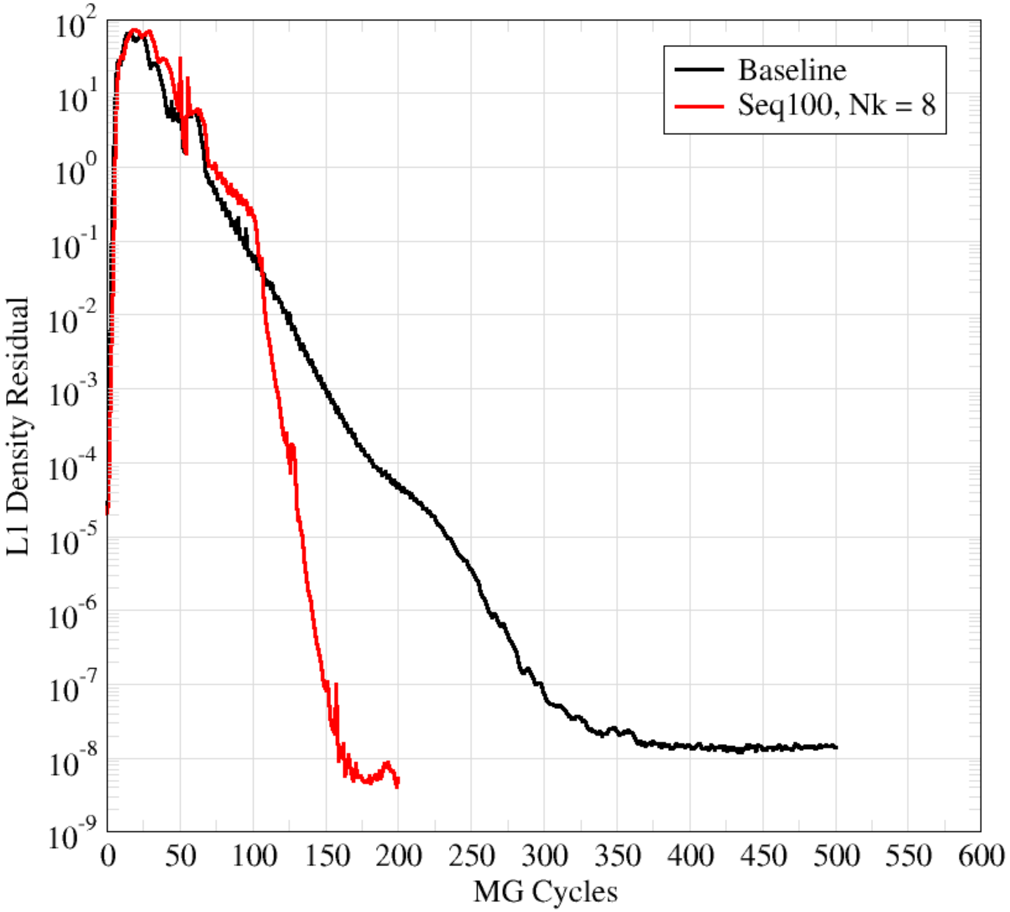
Reduced computational cost of preconditioned algorithm

CFL constraint on initial startup



ONERA M6 wing also improved convergence rate with preconditioning





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

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

A^{-1}-preconditioning stabilizes the pseudo-time iterations

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

Naïve dual time stepping for multi-stage scheme:

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

Unstable for small Δt

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

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

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

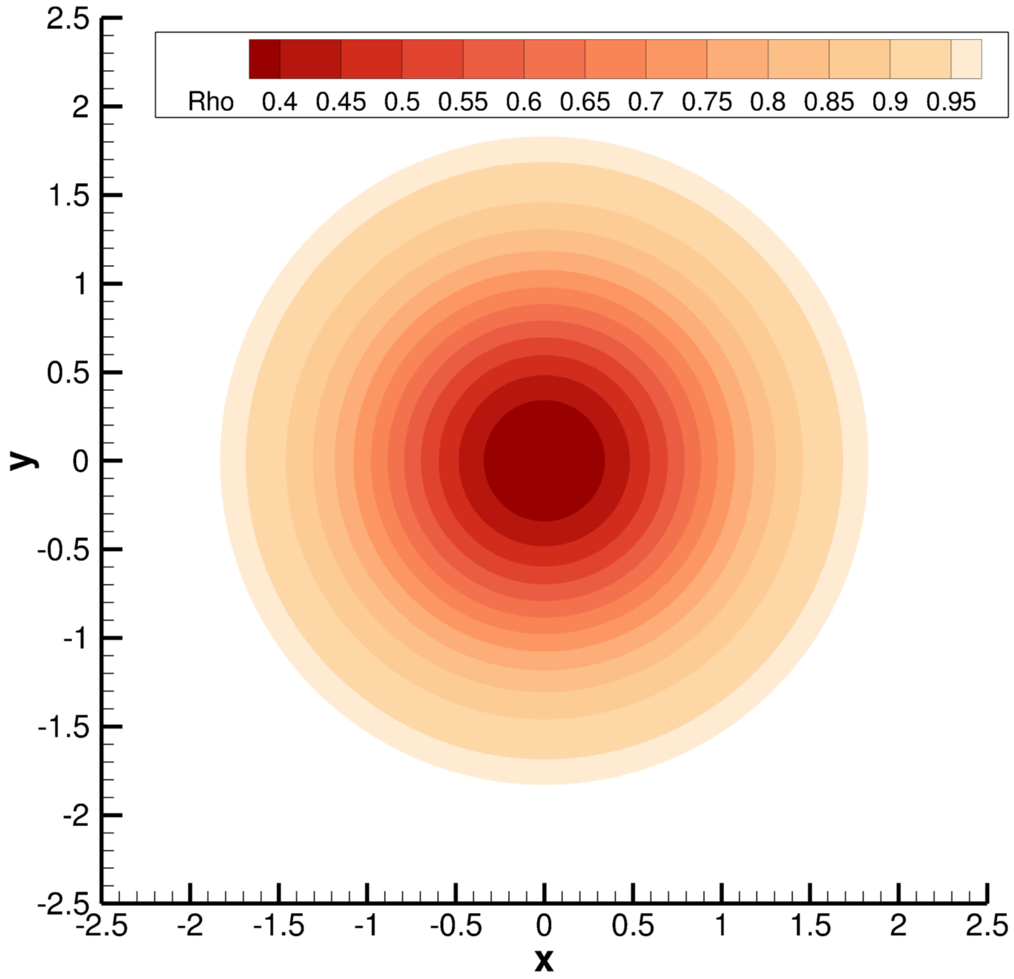
$$A = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}$$

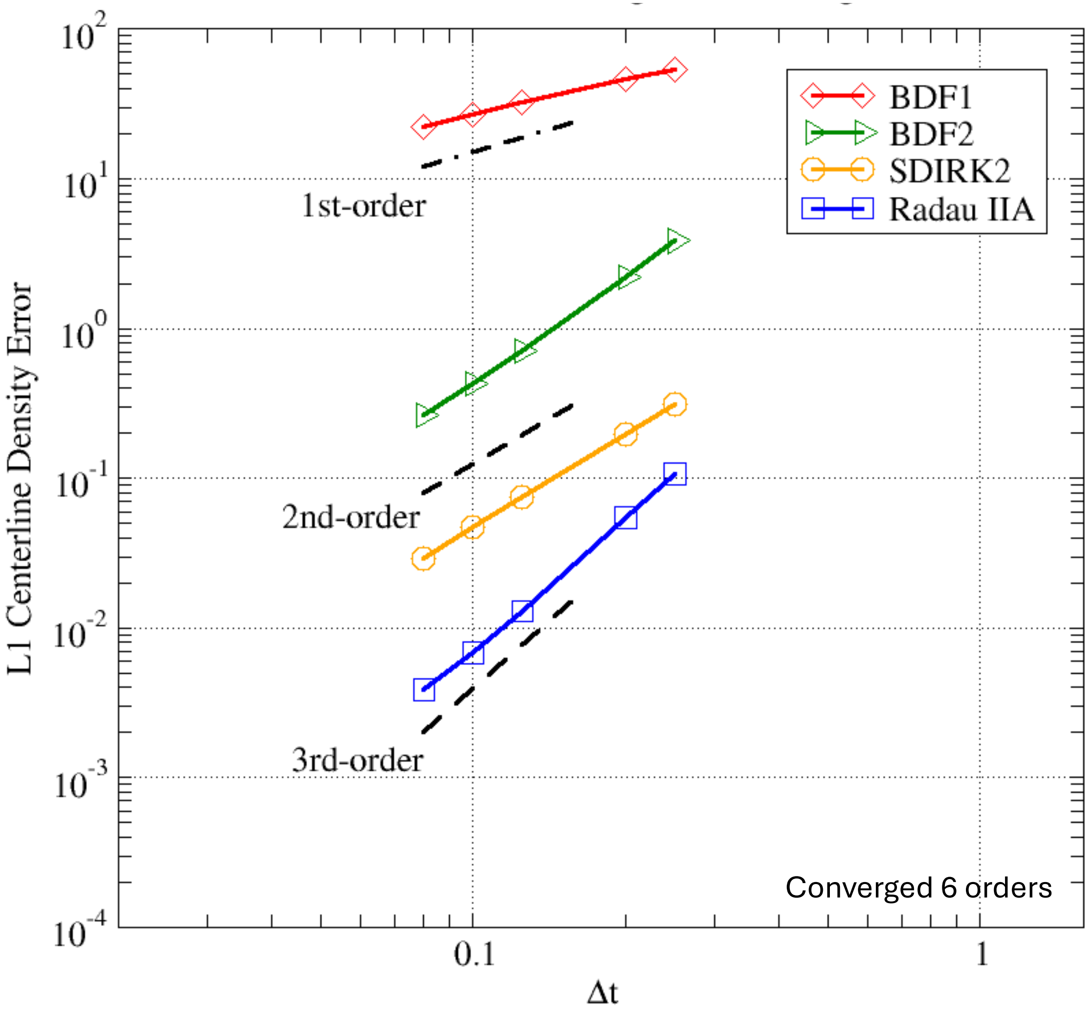
BDF1 (1st-order accurate)

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

BDF2 (2nd-order accurate)

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





Summary

Implicit preconditioning of a multigrid solver significantly improves its convergence rate on several 2D and 3D test cases

Jacobian-free Newton Krylov provides a **low memory, matrix-free** algorithm for solving the linear 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 and limiters

Investigate W-cycle vs V-cycle and **explicit multigrid preconditioning** of the GMRES linear solve

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 JFNK solvers

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

Questions?

Backup

Implicit preconditioning improves the multigrid subiteration convergence rate from 0.71 to 0.54