An Efficient Low Memory Implicit DG Algorithm for Time Dependent Problems

P.-O. Persson and J. Peraire

Massachusetts Institute of Technology

2006 AIAA Aerospace Sciences Meeting, Reno, Nevada

January 9, 2006

Background

- High-order Discontinuous Galerkin methods have many attractive features
- However:
  - Very stiff systems for practical applications
  - High computational cost and storage requirements
- Objective: Identify effective solution strategy for implicit solution of realistic problems
  - Experiment with several approaches
  - Identify “optimal” strategy

Equations and Discretization

- We consider the time-dependent compressible Navier-Stokes equations:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \boldsymbol{F}_i(\boldsymbol{u}) - \nabla \cdot \boldsymbol{F}_v(\boldsymbol{u}, \nabla \boldsymbol{u}) = 0$$

- Discretization in space using DG gives system of ODEs:

$$\boldsymbol{M}\dot{\boldsymbol{U}} = \boldsymbol{F}_v(\boldsymbol{U}) - \boldsymbol{F}_i(\boldsymbol{U}) \equiv \boldsymbol{R}(\boldsymbol{U})$$

- Discretization by BDF-k in time leads to nonlinear system

$$\boldsymbol{R}_{\mathrm{BDF}}(\boldsymbol{U}_n) \equiv \boldsymbol{M} \sum_{i=0}^{k} \alpha_i \boldsymbol{U}_{n-i} - \Delta t \boldsymbol{R}(\boldsymbol{U}_n) = 0$$

with mass-matrix $\boldsymbol{M}$ and residual $\boldsymbol{R}(\boldsymbol{U}_n)$

Equations and Discretization

- Newton's method requires solving

$$\boldsymbol{U}_n^{(j+1)} = \boldsymbol{U}_n^{(j)} + \beta \Delta \boldsymbol{U}_n^{(j)}$$

  with the Jacobian

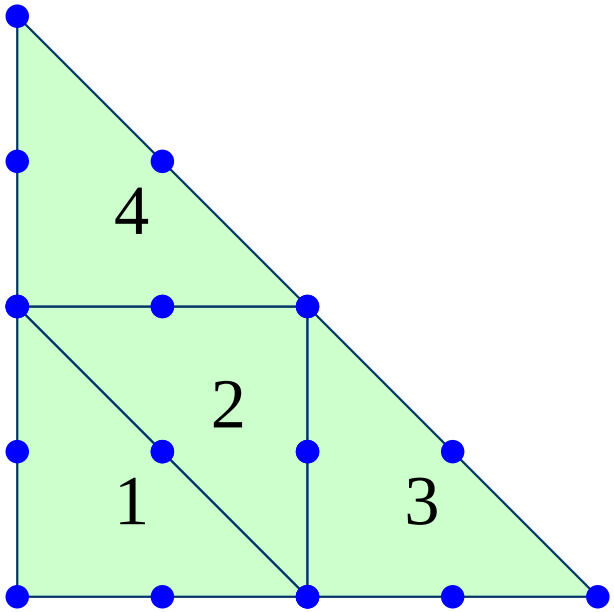
$$J(\boldsymbol{U}_n) = \frac{d\boldsymbol{R}_{\mathrm{BDF}}}{d\boldsymbol{U}_n} = \alpha_0 \boldsymbol{M} - \Delta t \frac{d\boldsymbol{R}}{d\boldsymbol{U}_n} \equiv \alpha_0 \boldsymbol{M} - \Delta t \boldsymbol{K}$$

- $\alpha_0 \approx 1$ for $k = 0, \ldots, 4$
- Study the solution of $\boldsymbol{A} = \boldsymbol{M} - \Delta t \boldsymbol{K}$ using iterative techniques

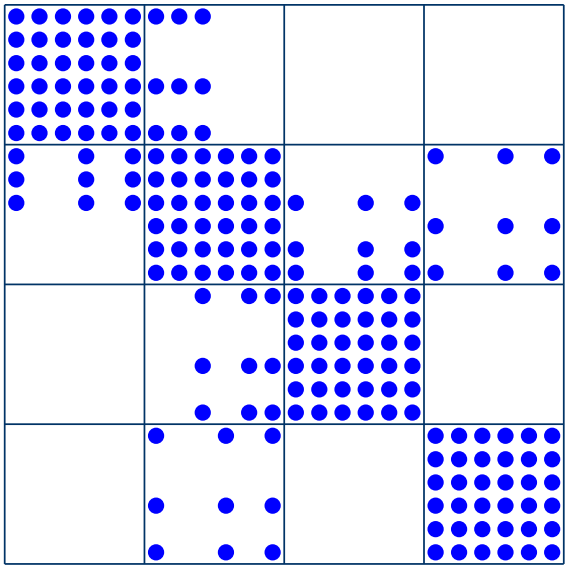
Matrix Structure – First-Order Systems

- DG discretization of first-order equations connects elements with neighboring face nodes
- Can be stored with compact dense block storage
  - Separate treatment of diagonal blocks and off-diagonal connections

Mesh



Jacobian



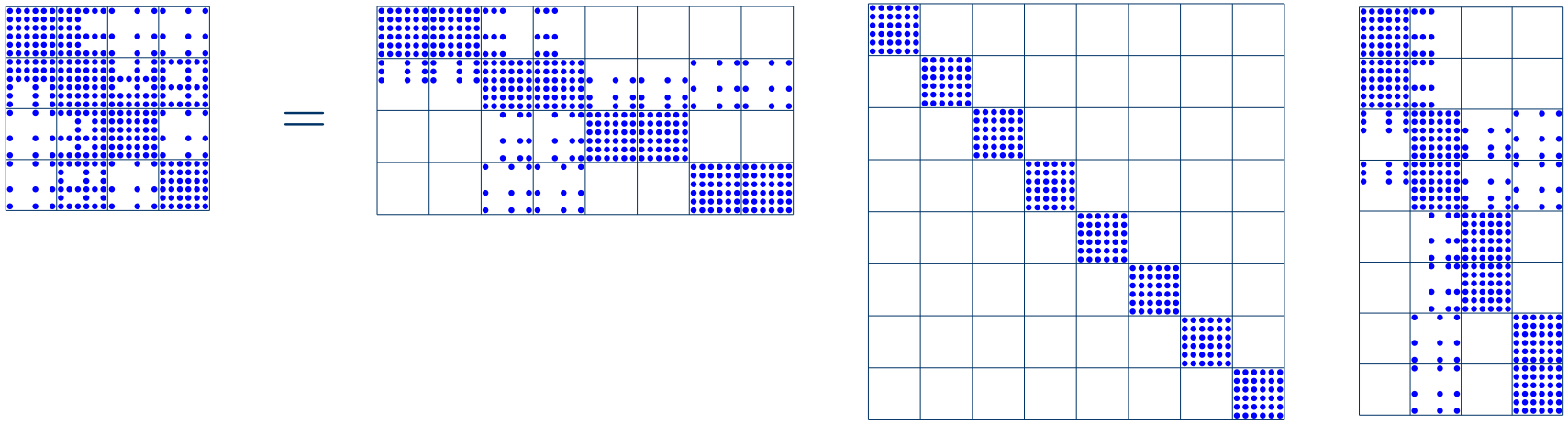
Matrix Structure – Viscous Terms

- The LDG method introduces additional variables $\boldsymbol{q} = \nabla \boldsymbol{u}$
- Resulting system can be written:

$$\boldsymbol{K}_v = \frac{\partial \boldsymbol{F}_v}{\partial \boldsymbol{U}} + \frac{\partial \boldsymbol{F}_v}{\partial \boldsymbol{Q}} M^{-1} \boldsymbol{C}$$

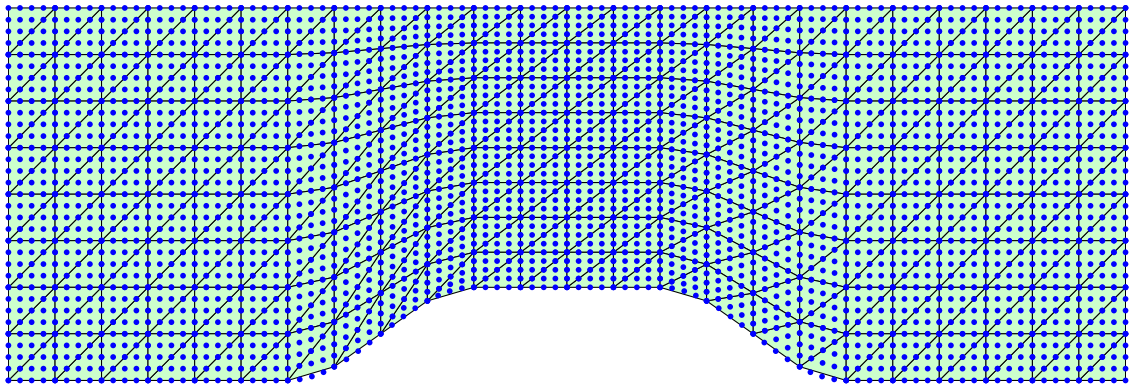
  where each matrix in the RHS has first-order structure (only neighbors)

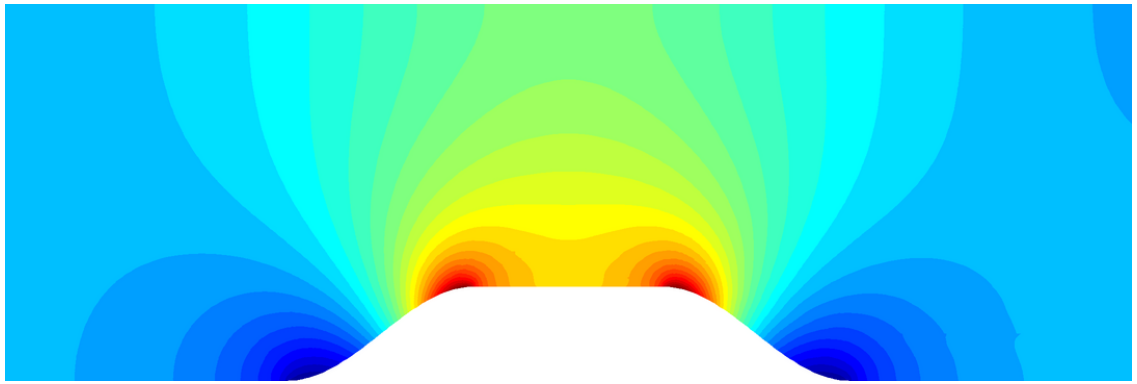
- Multiplication connects neighbors' neighbors $\Longrightarrow$ Wide stencil
- Proposed representation: Keep factors, store in structured format



Test Problem - Flow over Duct

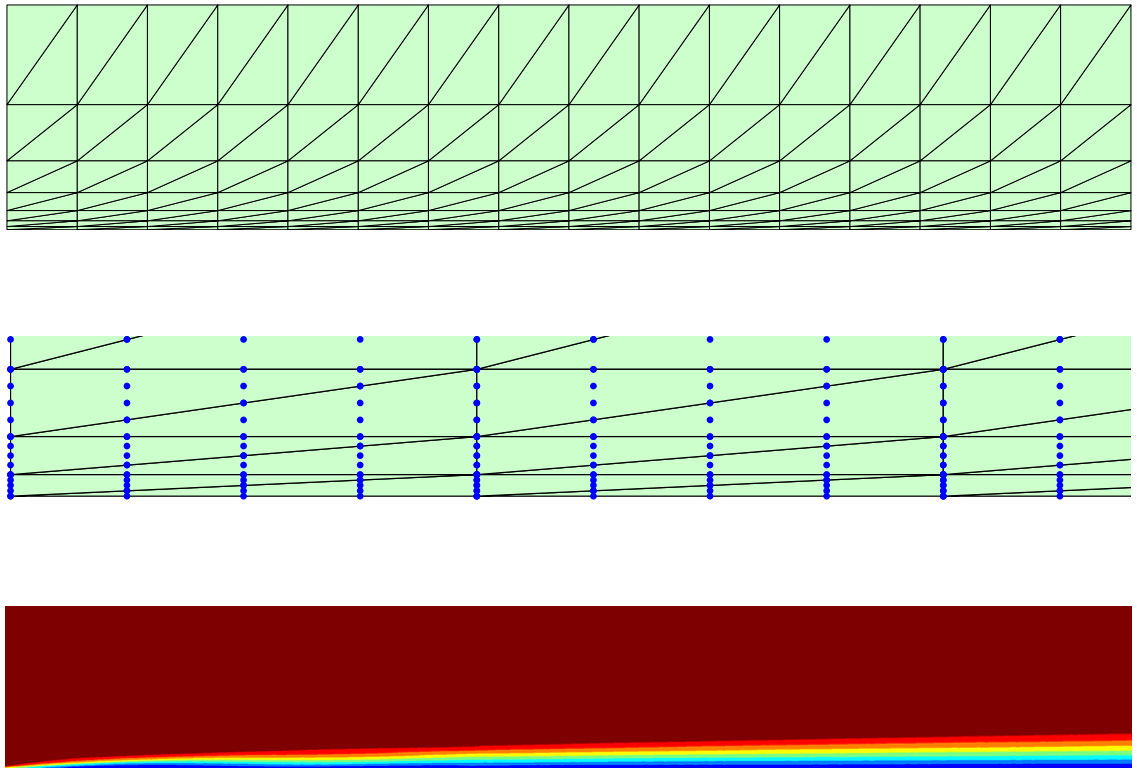
- Inviscid flow
- Structured, almost uniform mesh, 384 elements, $p = 4$





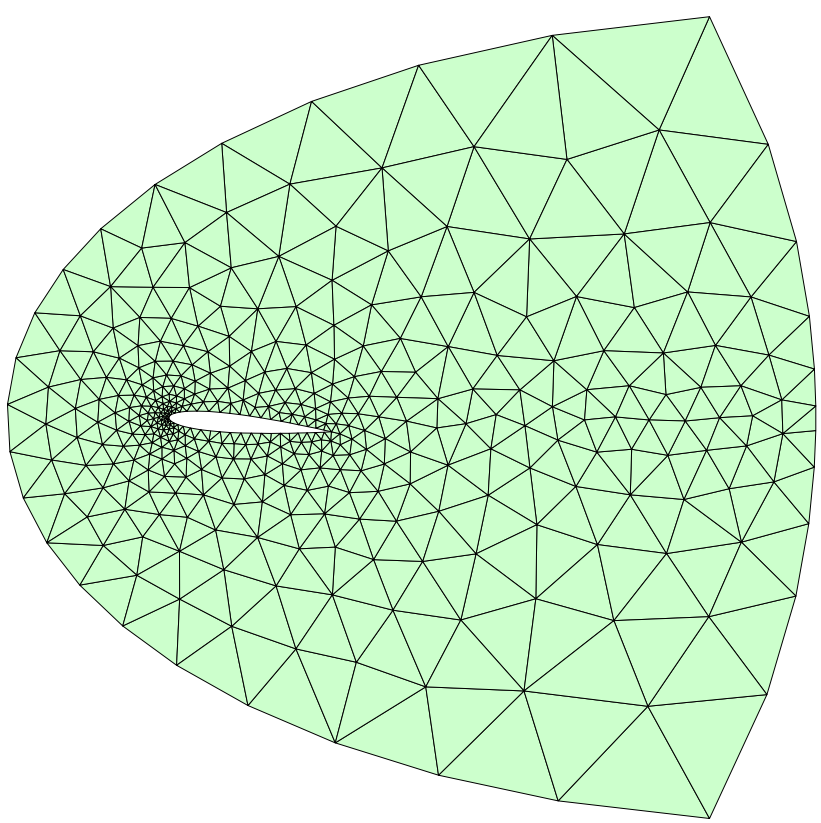
Test Problem - Flat Plate Boundary Layer

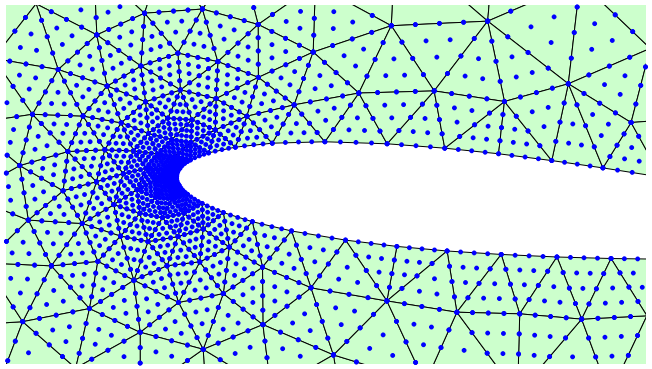
- Reynolds number 50,000
- Structured, graded mesh, 224 elements, stretching ≈ 20, $p = 4$

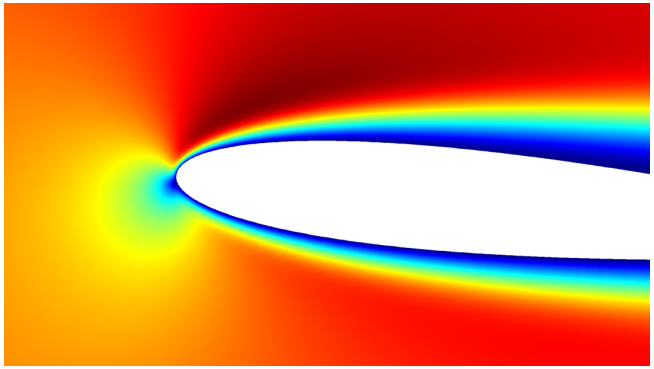


Test Problem - Flow around NACA Wing

- Reynolds number 1000
- Unstructured, graded but isotropic mesh, 735 elements, $p = 4$





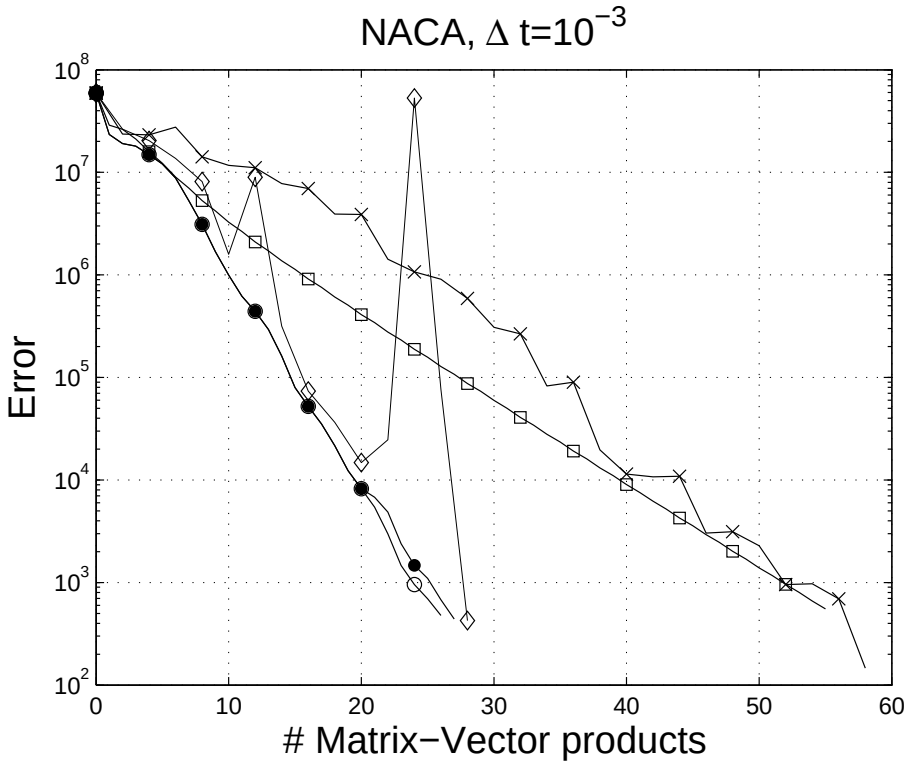


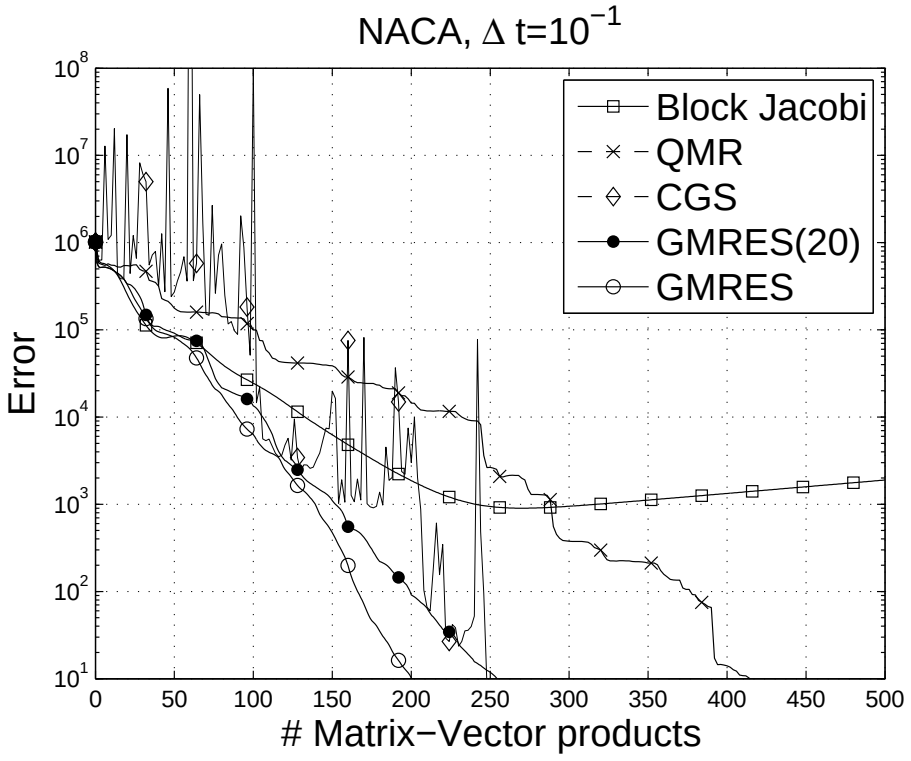
Iterative Solvers for Linear Systems

- Block-Jacobi
  - Simple, but can be efficient for certain problems
  - Convergence depends only weekly on approximation order p
  - Asymptotically poor convergence for viscous model problems
- Krylov subspace methods
  - Generally better convergence, in particular for model problems
  - Can be improved by preconditioning
  - For unsymmetric matrices: GMRES, GMRES(k), QMR, CGS, etc
- Computational cost for high-order discretizations dominated by matrix-vector products and preconditioning
  - Except possibly GMRES for large number of iterations

Convergence of Iterative Solvers

- Convergence of various solvers with block-diagonal preconditioner
- GMRES/GMRES(20)/CGS comparable cost, QMR about twice as much
- Block Jacobi great when it works, but unreliable
- From now on, focus on GMRES(20) for studying preconditioners





Preconditioners for Krylov Methods

- Performance of Krylov methods greatly improved by preconditioning
- Essentially find approximate solver for $\boldsymbol{Au} = \boldsymbol{b}$
- Various methods:
  - Simple matrix-based, e.g. block-diagonal $\boldsymbol{A} \approx \mathrm{blockdiag}(\boldsymbol{A})$
  - Complex matrix-based, e.g. block-Incomplete LU $\boldsymbol{A} \approx \tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}}$
  - Physics-based, e.g. multi-scale or low-degree preconditioner
- We are only interested in low-memory methods (storage $\leq \boldsymbol{A}$ itself):
  - Block-ILU(0) – Gaussian elimination without fill-in
    - Cost of factorization similar to inverting block diagonal
    - Cost of back-solves similar to matrix-vector product
    - Ignore neighbors' neighbors in LDG discretization
  - Coarse-scale – Precondition with $p = 1$ discretization

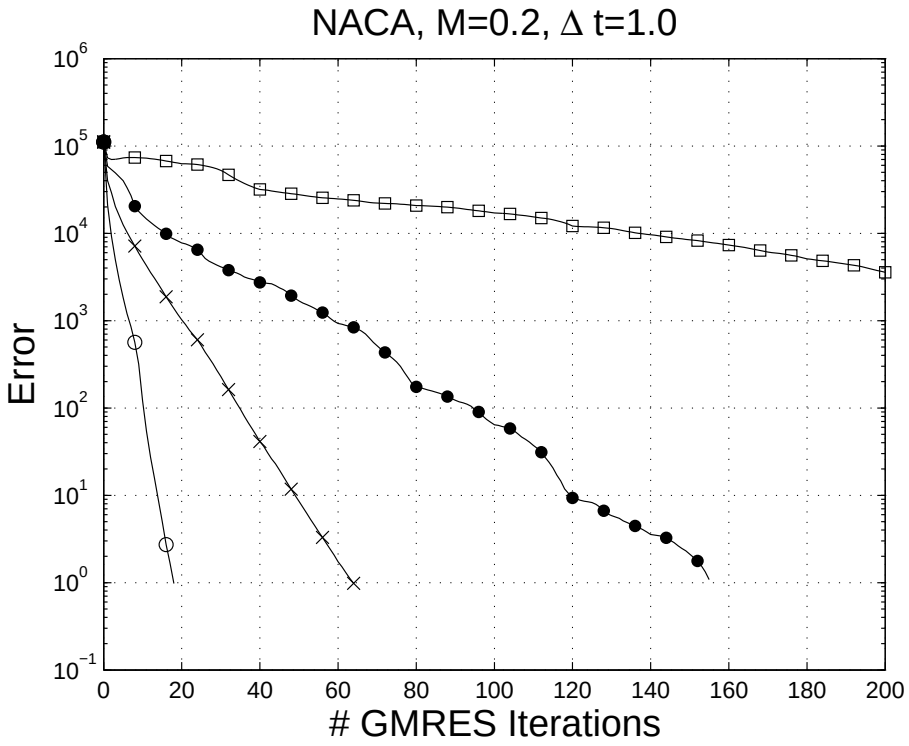
Block-ILU(0) Smoothing for $p = 1$ Preconditioner

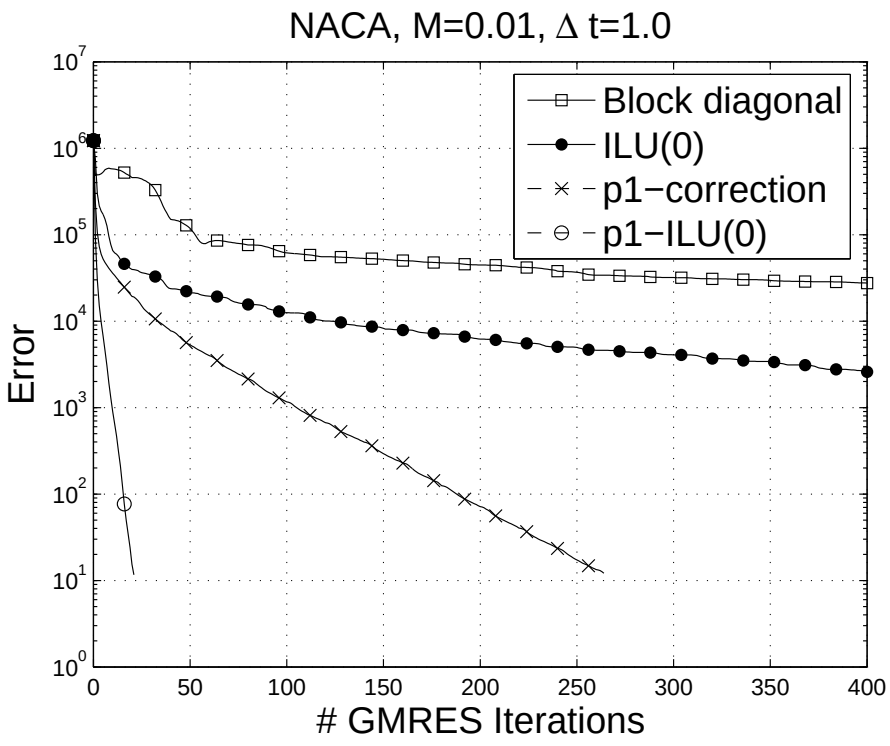
- In multigrid applications it it well-known ILU can be an effective smoother (it reduces high frequency errors)
- We have found that pre-smoothing a low-degree preconditioner for DG systems with Block-ILU(0) gives a remarkably effective method:

1. Solve $\tilde{L}\tilde{U}\boldsymbol{u}' = \boldsymbol{b}$ with block ILU(0) factorization $\boldsymbol{A} \approx \tilde{L}\tilde{U}$
2. Compute the residual $\boldsymbol{r}' = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{u}'$
3. Compute $p = 1$ projection $\boldsymbol{r}'_L$ of $\boldsymbol{r}'$ using the Koornwinder basis
4. Solve exactly (e.g. with direct solver) $\boldsymbol{A}_L \boldsymbol{e}'_L = \boldsymbol{r}'$, with $\boldsymbol{A}_L$ projected from $\boldsymbol{A}$
5. Compute prolongation $\boldsymbol{e}'$ from $\boldsymbol{e}'_L$
6. Add correction $\boldsymbol{u}'' = \boldsymbol{u}' + \boldsymbol{e}'$
7. Compute $\boldsymbol{u}$ by applying a Jacobi step to $\boldsymbol{u}''$

Convergence with Preconditioners

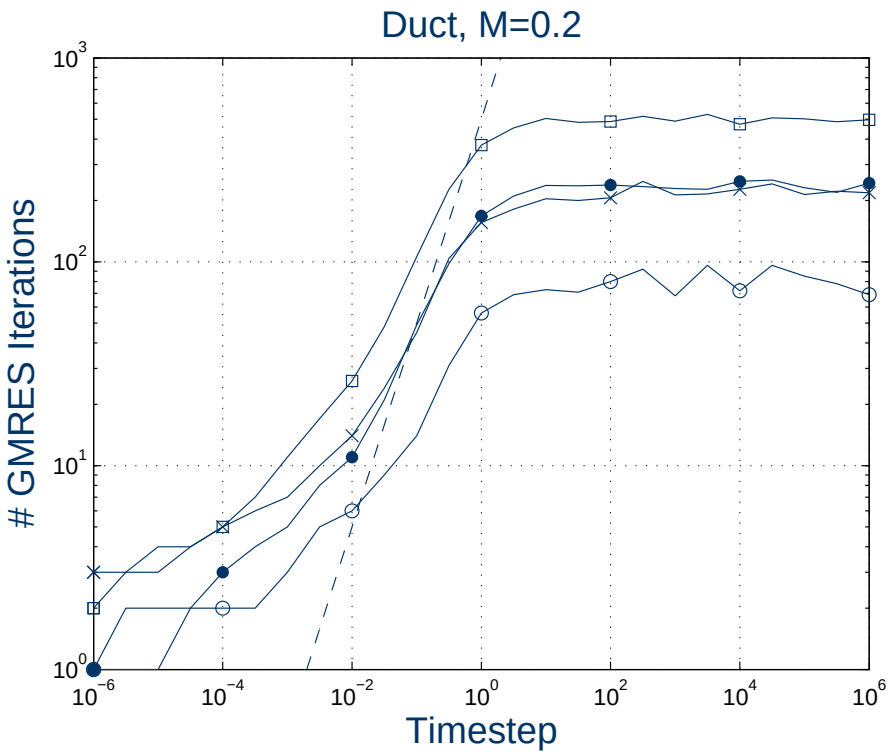
- Convergence of GMRES(20) for various preconditioners
- $p1$ preconditioning with Jacobi smoothing better than Block-ILU(0) (but likely more expensive)
- $p1$-ILU(0) excellent performance, in particular for low Mach

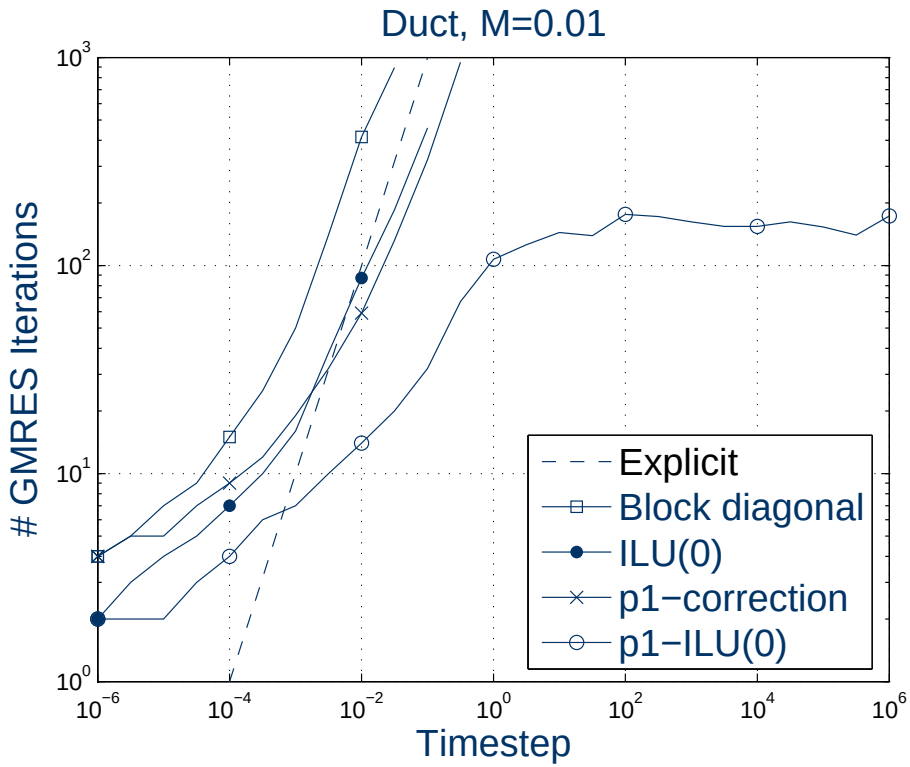




Timestep Dependence - Duct Problem

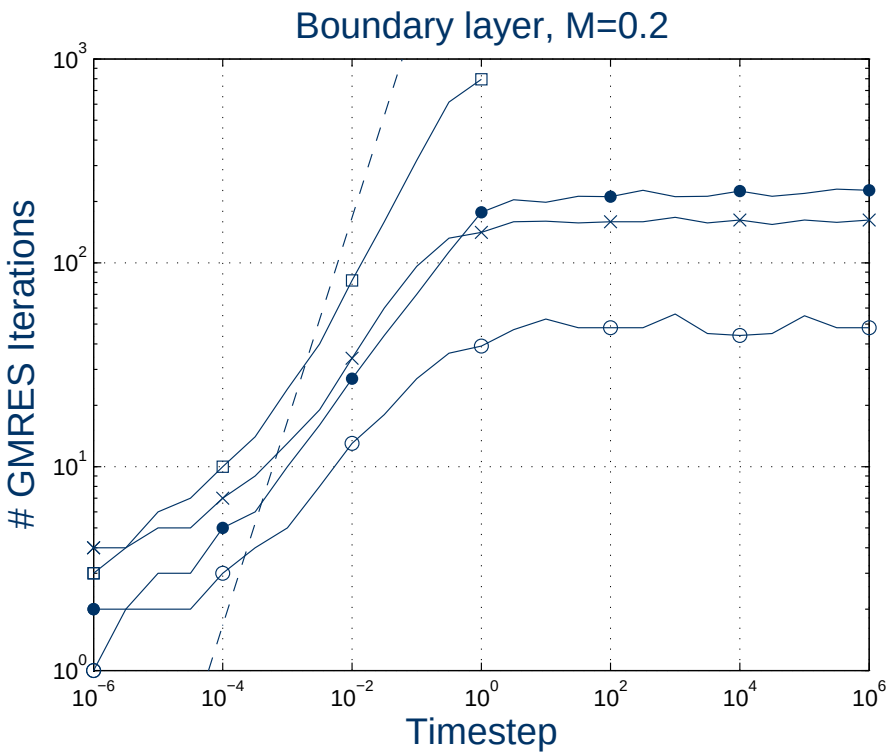
- #GMRES iterations as function of timestep Δt for various preconditioners
- Performances scale similarly for wide range of Δt, except for low Mach
- Also – need very large timesteps to beat explicit techniques (dashed line), because of uniform mesh and no viscosity

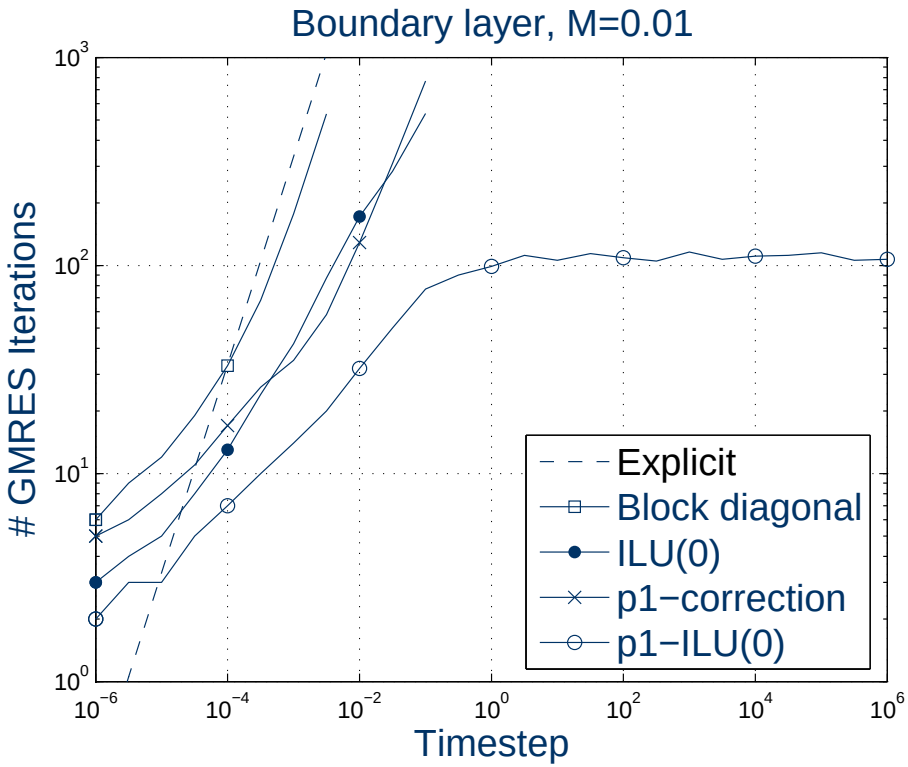




Timestep Dependence - Boundary Layer Problem

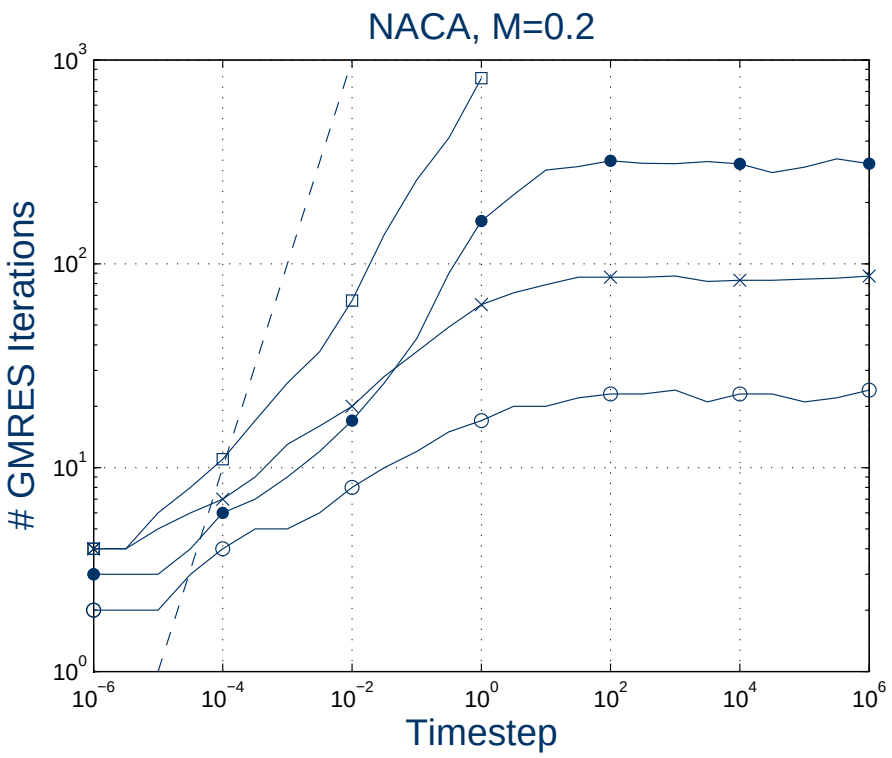
- Results similar to before, but implicit methods more favorable than explicit because of viscosity and mesh grading

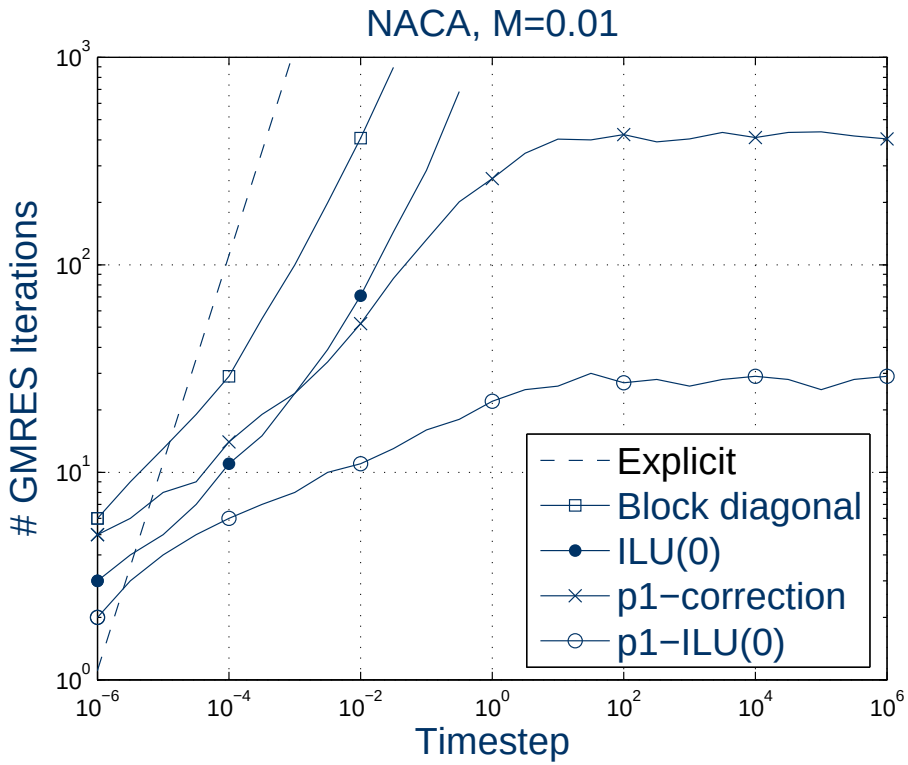




Timestep Dependence - NACA Problem

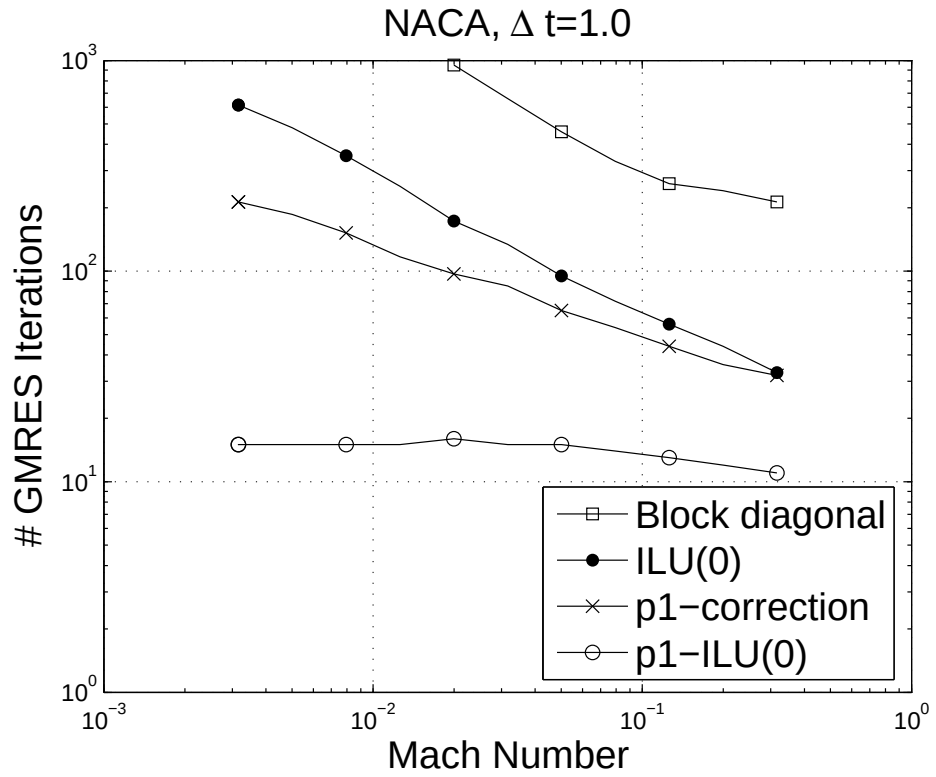
- Again similar to before, but $p1$ with Jacobi now better than ILU(0), likely because of higher viscosity and/or less graded mesh





Mach Number Dependence (NACA Problem)

- Most methods perform very poor on low Mach problems
- However, the $p1$-ILU(0) preconditioner appears to make convergence almost independent of Mach number



Conclusions

- Efficient implicit timestepping of high-order DG problems
- Total memory requirement of same order as Jacobian matrices
- Wide stencil of LDG circumvented by structured block storage and ILU preconditioning of compact stencil
- Block-ILU(0) smoothing for low-degree preconditioner ("$p1$-ILU(0)") a promising scheme
  - Great performance and low cost
  - Handles low Mach numbers remarkably well
- Future work includes further verification of the performance of $p1$-ILU(0) (3-D problems, higher Reynolds, turbulence modeling, etc)