Implicit Solution of Viscous Aerodynamic Flows using the Discontinuous Galerkin Method

Per-Olof Persson and Jaime Peraire

Massachusetts Institute of Technology

7th World Congress on Computational Mechanics

Los Angeles, July 16-22, 2006

Background

- Objective: Solve large scale problems in fluid mechanics efficiently and accurately on complex geometries
- Requirements:
  - Low dispersion/high accuracy discretization
  - Methods based on unstructured simplex meshes
  - Robust and fully automated solution process
  - Efficient solvers/simple parallelization

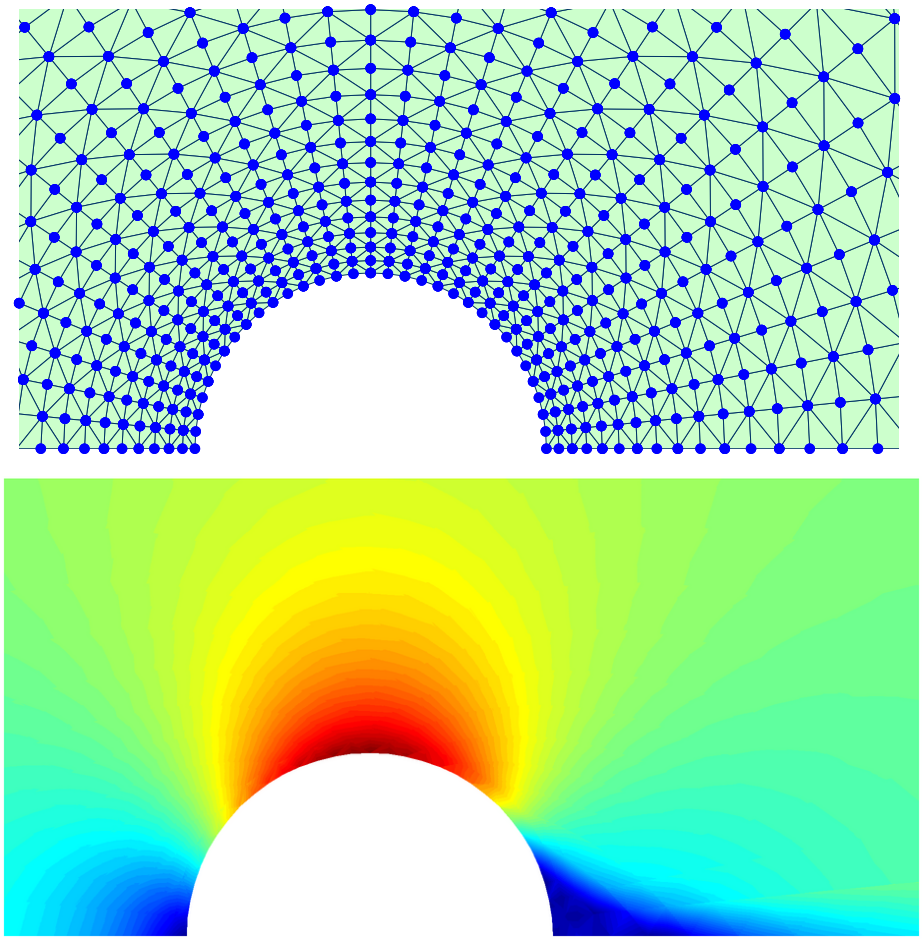
Application Areas

- Computational aeroacoustics
  - Direct solution of the compressible Navier-Stokes equations
  - No modeling of acoustic waves (Lighthill, linearized Euler, etc)
  - High accuracy essential to capture turbulent noise sources, propagate acoustic waves, and to ensure stability at low Mach numbers
  - Complex geometries, e.g. in wind noise prediction for vehicles
- Other DNS/LES/DES problems
  - High order methods a clear benefit in DNS/LES region
  - Need flexible meshes e.g. in the boundary layers
- Drag prediction
  - Solve high Reynolds number transonic flow for various geometries
  - Determine drag coefficient accurately

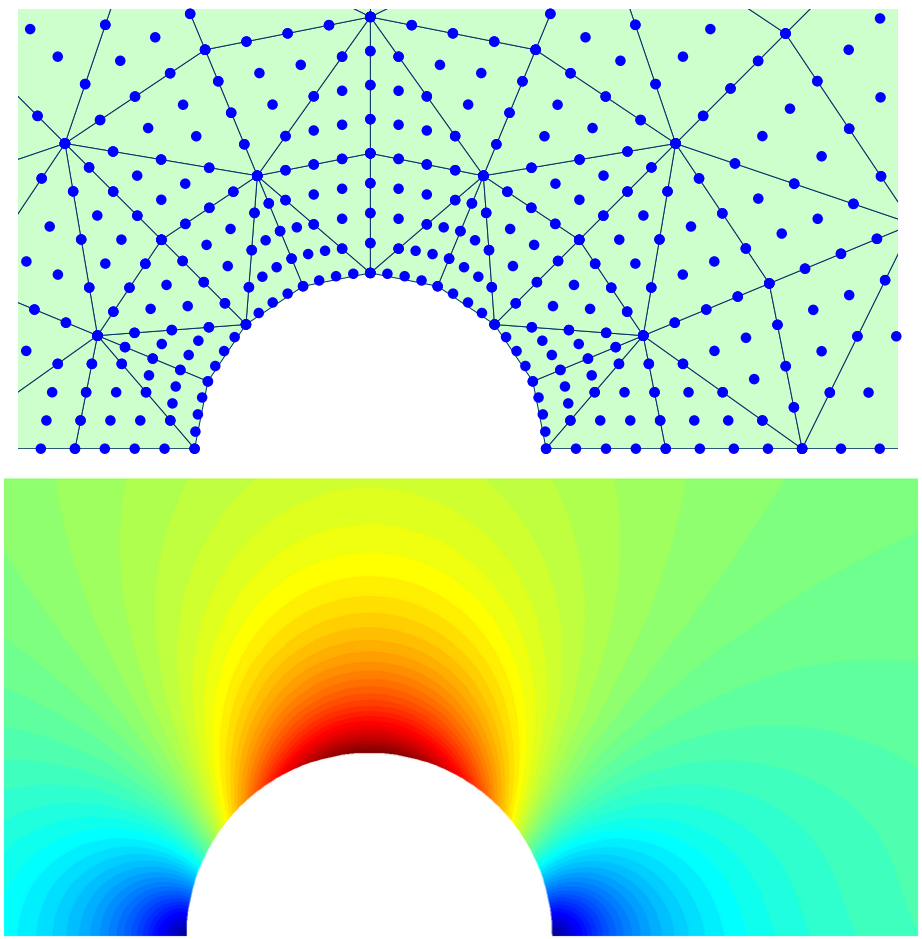
Example: Inviscid 2-D Flow

- Lower dissipation with high order schemes $\longrightarrow$ Higher accuracy/DOF

24576 DOFs, $p = 1$

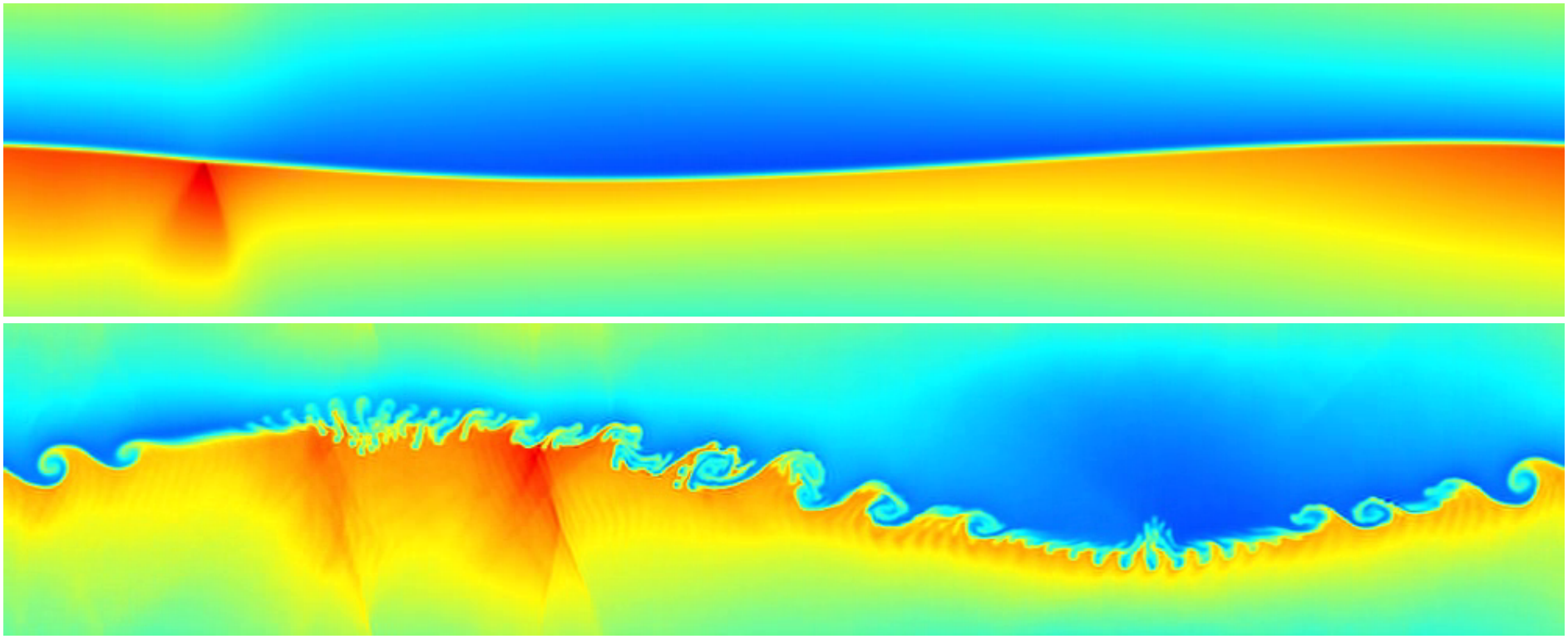


7680 DOFs, $p = 4$



Example: Aeroacoustics and K-H Instability

- Inspired by calculations of Munz et al (2003) (linearized)
- Nonlinear behavior: Large scale acoustic wave interacts with small scale flow features, leading to vorticity generation
- $p = 7$ (8th order accuracy), 100-by-20 square elements



The Discontinuous Galerkin Method

- (Reed/Hill 1973, Lesaint/Raviart 1974, Cockburn/Shu 1989-, etc)
- Consider non-linear hyperbolic system in conservative form:

$$\boldsymbol{u}_t + \nabla \cdot \mathcal{F}_i(\boldsymbol{u}) = 0$$

- Triangulate domain Ω into elements $\kappa \in T_h$
- Seek approximate solution $\boldsymbol{u}_h$ in space of element-wise polynomials:

$$\mathcal{V}_h^p = \{\boldsymbol{v} \in L^2(\Omega) : \boldsymbol{v}|_\kappa \in P^p(\kappa) \ \forall \kappa \in T_h\}$$

- Multiply by test function $\boldsymbol{v}_h \in \mathcal{V}_h^p$ and integrate over element κ:

$$\int_\kappa \left[(\boldsymbol{u}_h)_t + \nabla \cdot \mathcal{F}_i(\boldsymbol{u}_h)\right] \boldsymbol{v}_h \, d\boldsymbol{x} = 0$$

The Discontinuous Galerkin Method

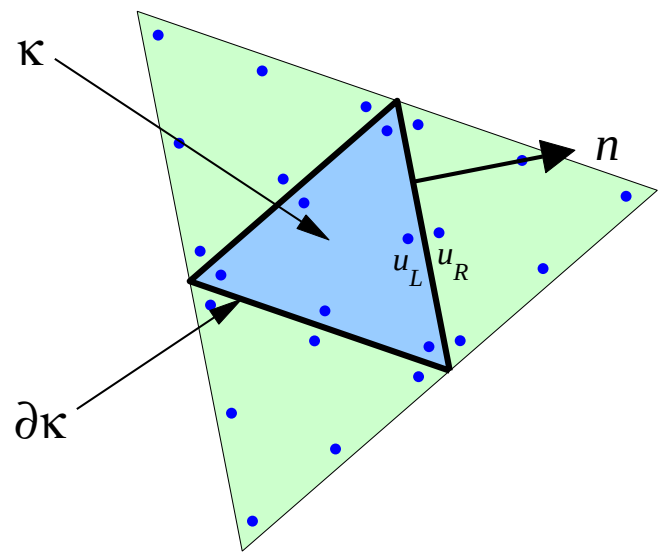
- Integrate by parts:

$$\int_\kappa [(\boldsymbol{u}_h)_t]\, \boldsymbol{v}_h \, d\boldsymbol{x} - \int_\kappa \mathcal{F}_i(\boldsymbol{u}_h) \nabla \boldsymbol{v}_h \, d\boldsymbol{x} + \int_{\partial\kappa} \mathcal{H}_i(\boldsymbol{u}_h^+, \boldsymbol{u}_h^-, \hat{\boldsymbol{n}}) \boldsymbol{v}_h^+ \, ds = 0$$

  with numerical flux function $\mathcal{H}_i(\boldsymbol{u}_L, \boldsymbol{u}_R, \hat{\boldsymbol{n}})$ for left/right states $\boldsymbol{u}_L$,$\boldsymbol{u}_R$ in direction $\hat{\boldsymbol{n}}$ (Godunov, Roe, Osher, Van Leer, Lax-Friedrichs, etc)

- Global view: Find $\boldsymbol{u}_h \in \mathcal{V}_h^p$ such that this weighted residual is zero for all $\boldsymbol{v}_h \in \mathcal{V}_h^p$

- Error $\mathcal{O}(h^{p+1})$ for smooth problems



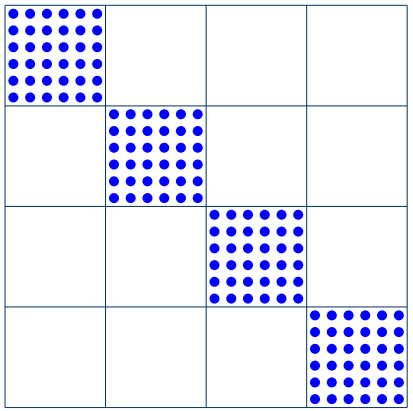
The DG Method – Observations

- Reduces to the finite volume method for $p = 0$:

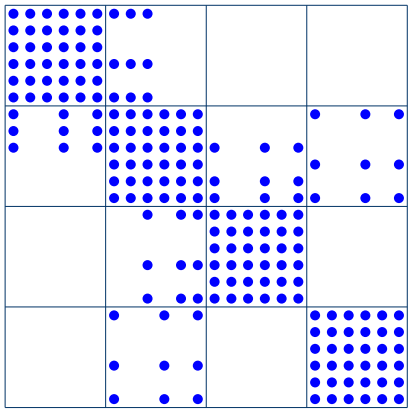
$$(\boldsymbol{u}_h)_t A_\kappa + \int_{\partial\kappa} \mathcal{H}_i(\boldsymbol{u}_h^+, \boldsymbol{u}_h^-, \hat{\boldsymbol{n}})\, ds = 0$$

- Boundary conditions enforced naturally for any polynomial degree p
- Mass matrix block-diagonal (no overlap between element basis functions)
- Edge integrals connect neighboring elements – block-wise compact stencil

Mass Matrix



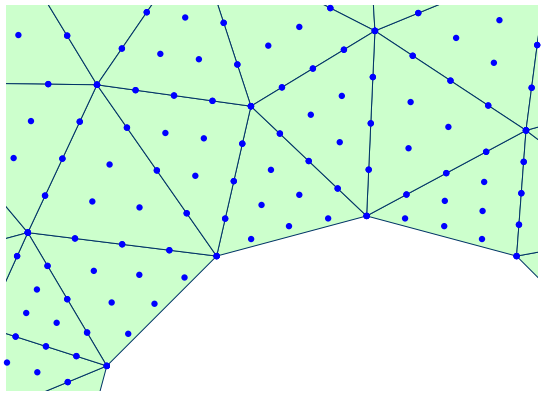
Jacobian



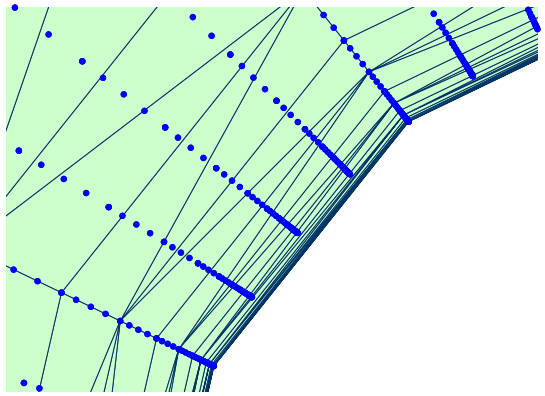
Nodal Basis and Curved Boundaries

- Nodal basis within each element $\longrightarrow$ sparse element connectivities
- Isoparametric treatment of curved elements (not only at boundaries)

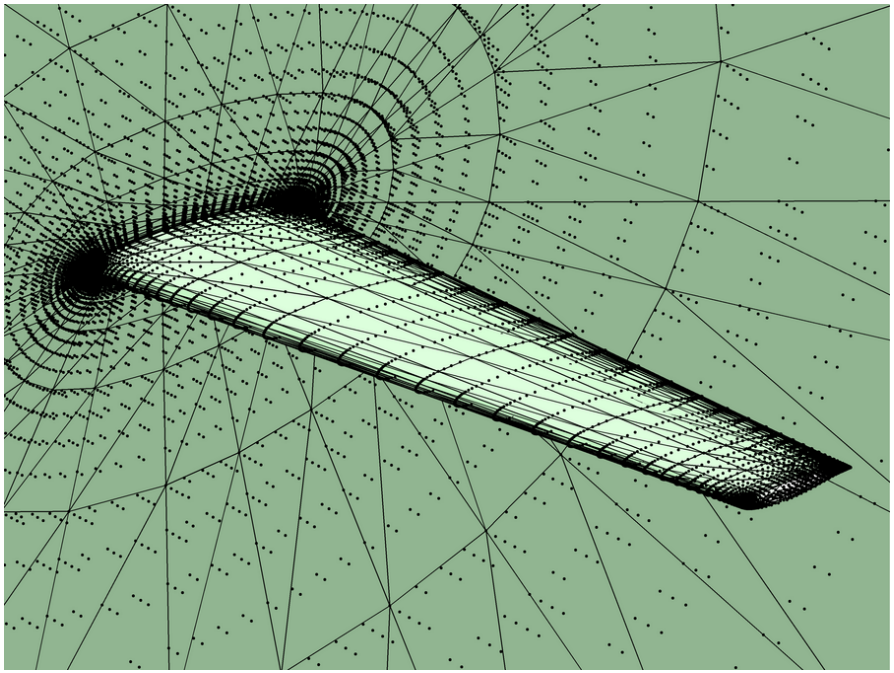
Isotropic mesh



Anisotropic mesh

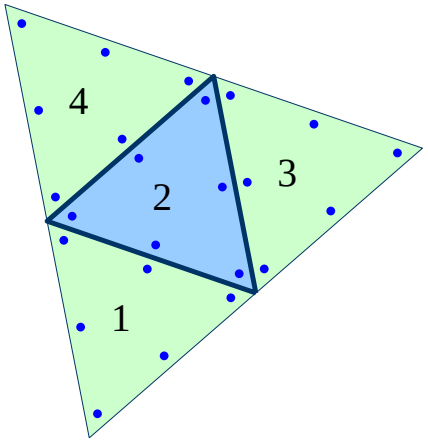


W1 wing (Drag Prediction Workshop)

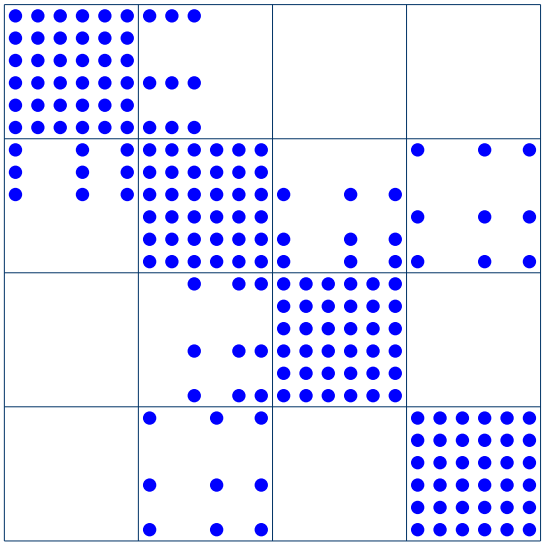


Viscous Discretization

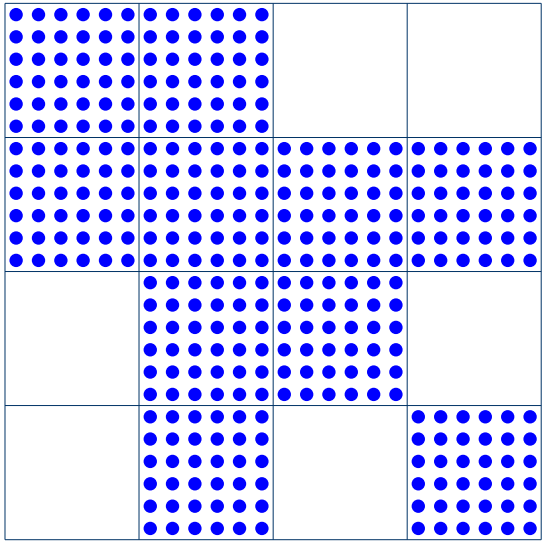
- Our *Compact DG* (CDG) scheme reduces the connectivities for the viscous discretization
- About 50% fewer DOFs than full connectivity (3-D tetrahedra, $p = 4$)



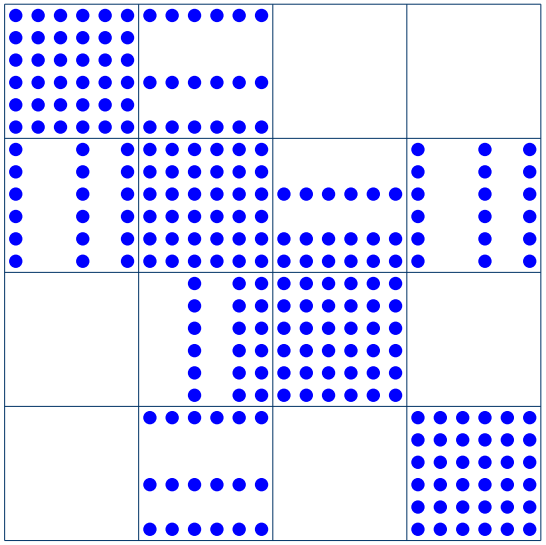
Inviscid



Full Connectivity



Compact DG



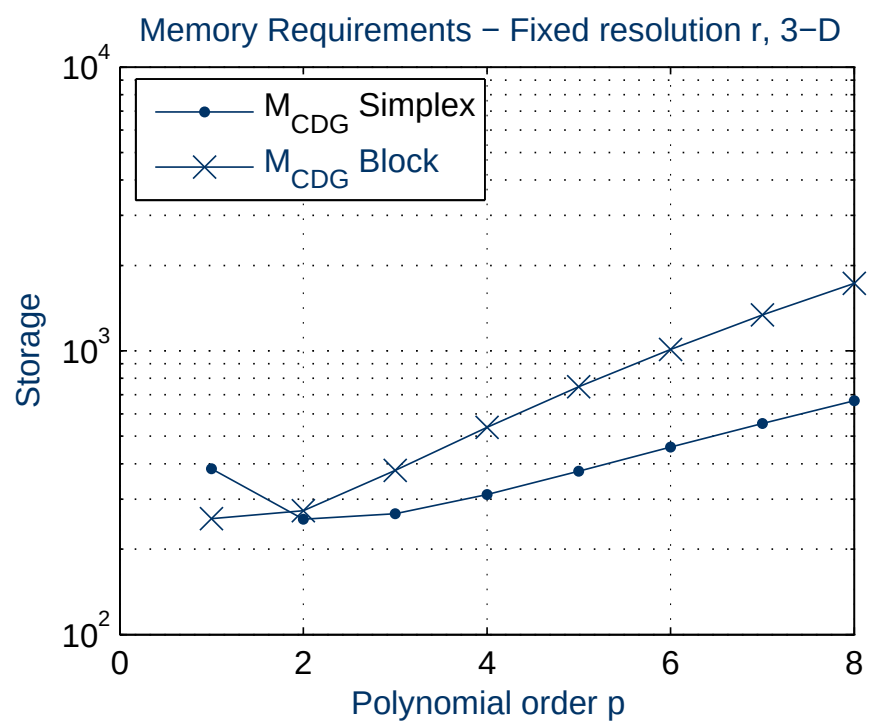
Implicit DG Memory Requirements

- Consider simplex elements, polynomial degree p, mesh resolution r, with N solution components in D dimensions
  - Nodes/element $S = \binom{p+D}{D}$
  - Nodes/edge element $S_e = \binom{p+D-1}{D-1}$
  - Element neighbors $B = D + 1$
  - Elements $T = \frac{D!}{(pr)^D}$
- Storage required for solution and Jacobian:
  - DOFs = SNT
  - Matrix entries (Full) = $(SN)^2T + (SN)^2BT$
  - Matrix entries (CDG) = $(SN)^2T + SS_eN^2BT$

Implicit DG Memory Requirements

- Example: Navier-Stokes + S-A in 3-D, $p = 4$, $10,000$ tetrahedra
  - DOFs = $6 \cdot 35 \cdot 10,000 = 2,100,000$
  - Matrix entries (Full) = $2.2 \cdot 10^9$ elements = $17.6GB$
  - Matrix entries (CDG) = $1.2 \cdot 10^9$ elements = $9.6GB$

- Observations:
  - $p = 2$ and $p = 3$ optimal for tetrahedra
  - For $p \geq 2$, 6 tetrahedra cheaper than 1 hex (!)



Matrix Storage and Operations

- Store matrices in block format
  - Diagonal blocks: SN-by-SN-by-T dense array
  - Off-diagonal blocks (CDG): SN-by-S_eN-by-B-by-T dense array
- Use high-performance dense matrix libraries for operations
  - BLAS3 performance for matrix assembly (matrix-matrix products)
  - BLAS2 performance in Krylov solver (matrix-vector products)
  - Direct solvers: BLAS3 performance by block Gaussian elimination
- Hard to take advantage of block structure with a general format such as compressed column storage

Time Integration

- 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

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

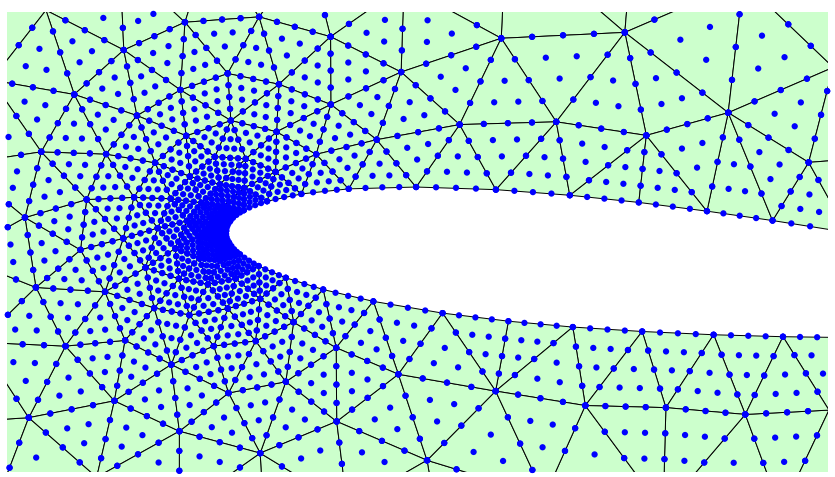
with mass-matrix $\boldsymbol{M}$ and residual $\boldsymbol{R}(\boldsymbol{U}_n)$

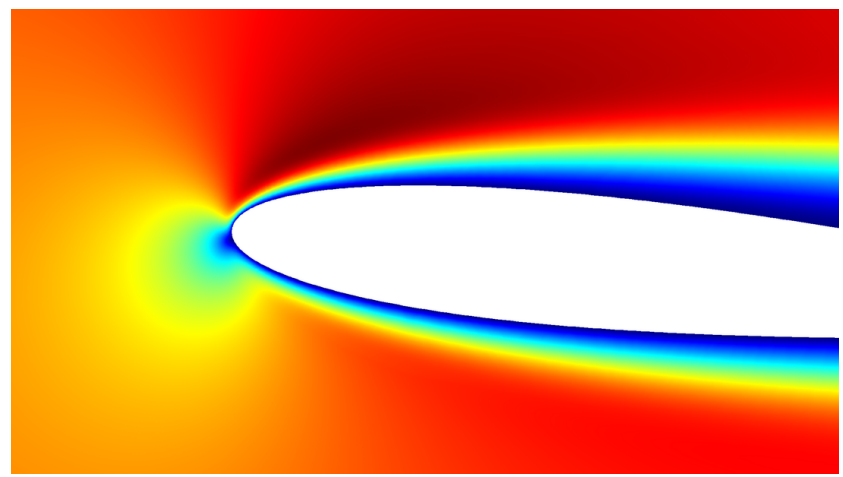
Time Integration

- 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 iterative techniques for $\boldsymbol{A} = \boldsymbol{M} - \Delta t \boldsymbol{K}$
- Use NACA 0012 model problem, Re=1000, $p = 4$



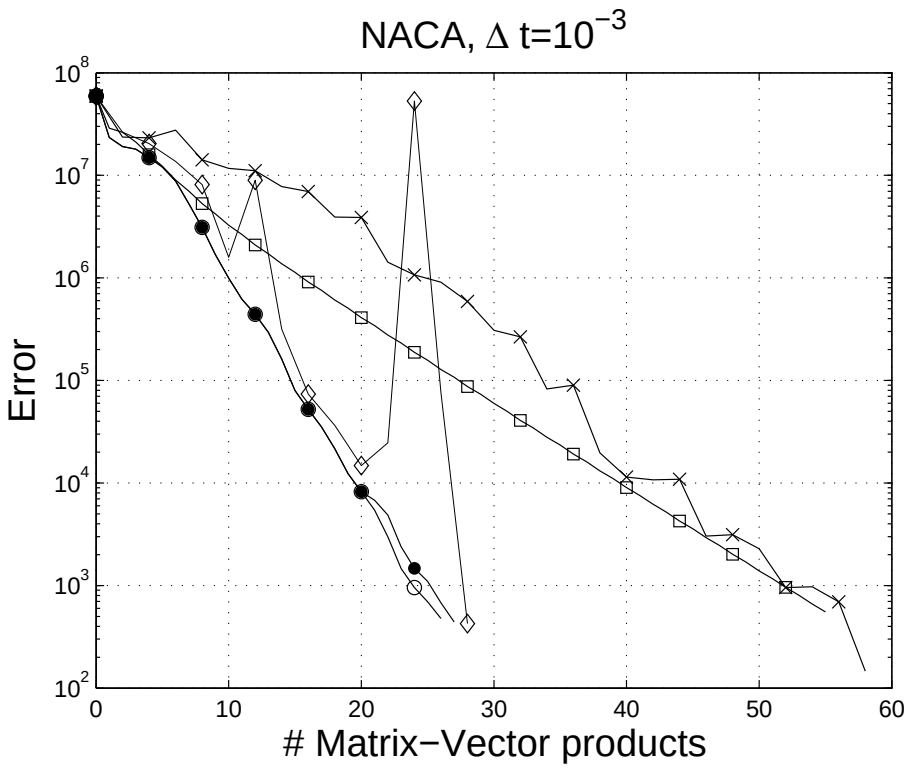


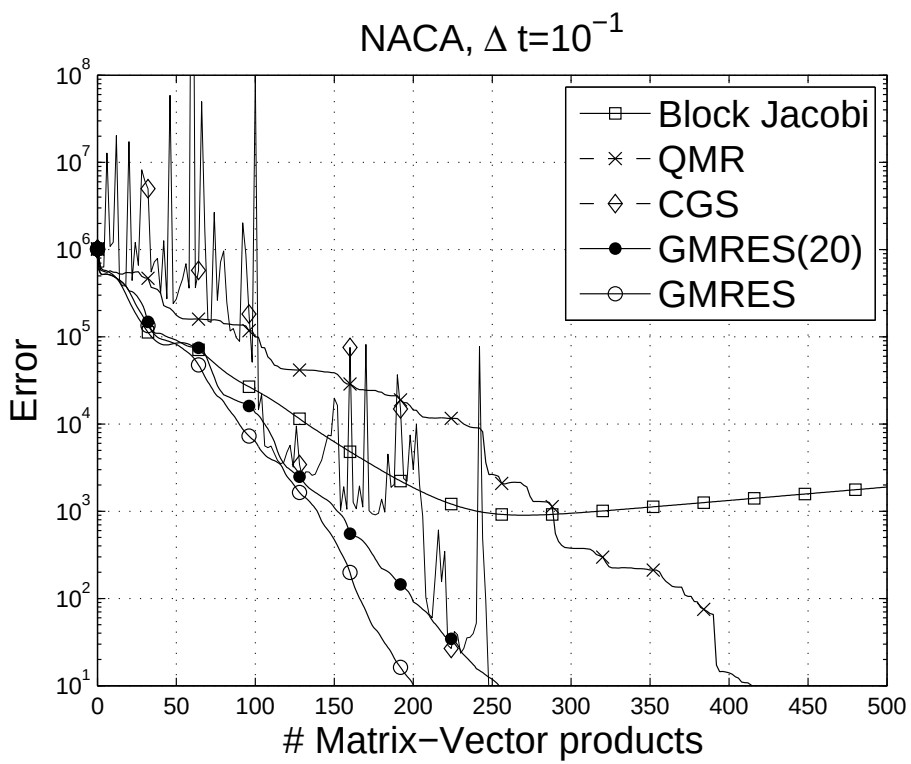
Iterative Solvers for Linear Systems

- 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
- Various standard methods:
  - Block-diagonal: $\boldsymbol{A} \approx \mathrm{blockdiag}(\boldsymbol{A})$
  - Block-Incomplete LU: $\boldsymbol{A} \approx \tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}}$
  - p-multigrid (low-degree) preconditioner
- Highly efficient combination (Persson/Peraire 2006)
  - Use block-ILU(0) as post-smoother for coarse scale correction
  - In particular, convergence is almost independent of Mach number

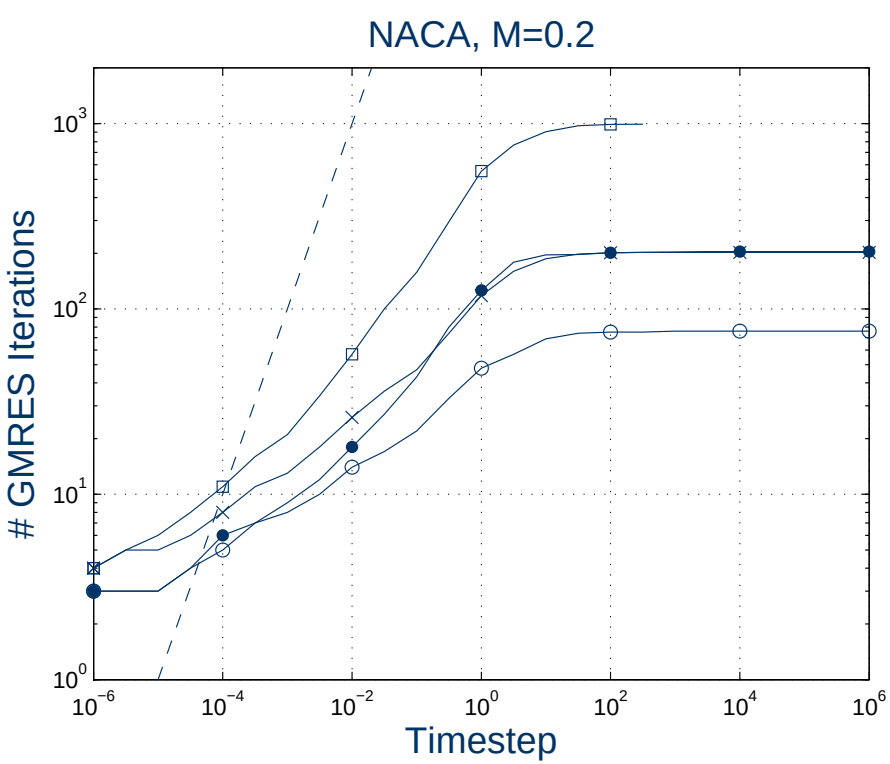
Block-ILU(0)/Coarse Scale Preconditioner

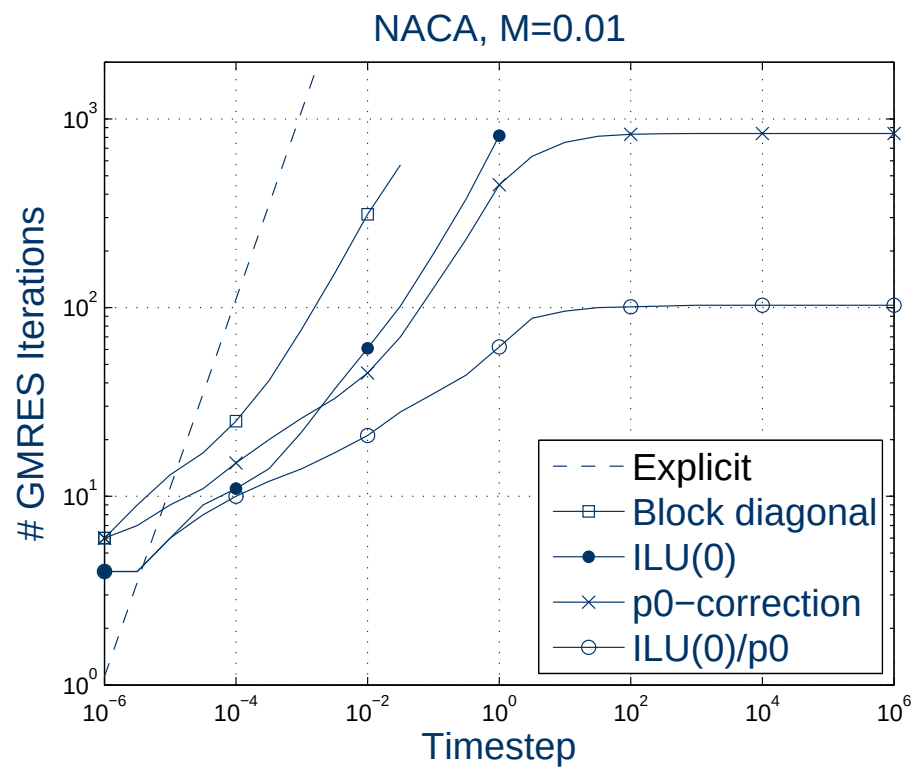
- Pseudo-code for our preconditioner (approximate solution of $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b}$)

  1. Compute coarse projection $\boldsymbol{b}_0$ of $\boldsymbol{b}$ (element-wise averaging)
  2. Solve exactly (e.g. with direct solver) $\boldsymbol{A}_0\boldsymbol{u}_0 = \boldsymbol{b}_0$, with $\boldsymbol{A}_0$ projected from $\boldsymbol{A}$
  3. Compute prolongation $\boldsymbol{u}$ from $\boldsymbol{u}_0$
  4. Compute the residual $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{u}$
  5. Solve $\tilde{L}\tilde{U}\boldsymbol{u}' = \boldsymbol{r}$ with block ILU(0) factorization $\boldsymbol{A} \approx \tilde{L}\tilde{U}$
  6. Add correction $\boldsymbol{u}'$ to $\boldsymbol{u}$

Timestep Dependence

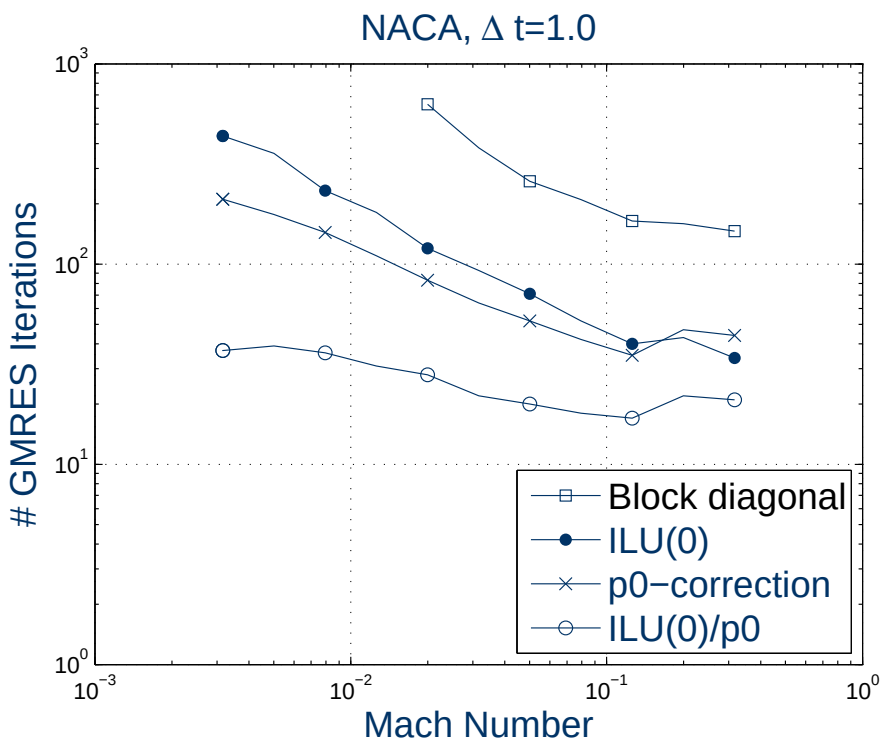
- #GMRES iterations as function of timestep Δt for various preconditioners
- Need large timesteps to beat explicit techniques (dashed line), because of uniform mesh and no viscosity





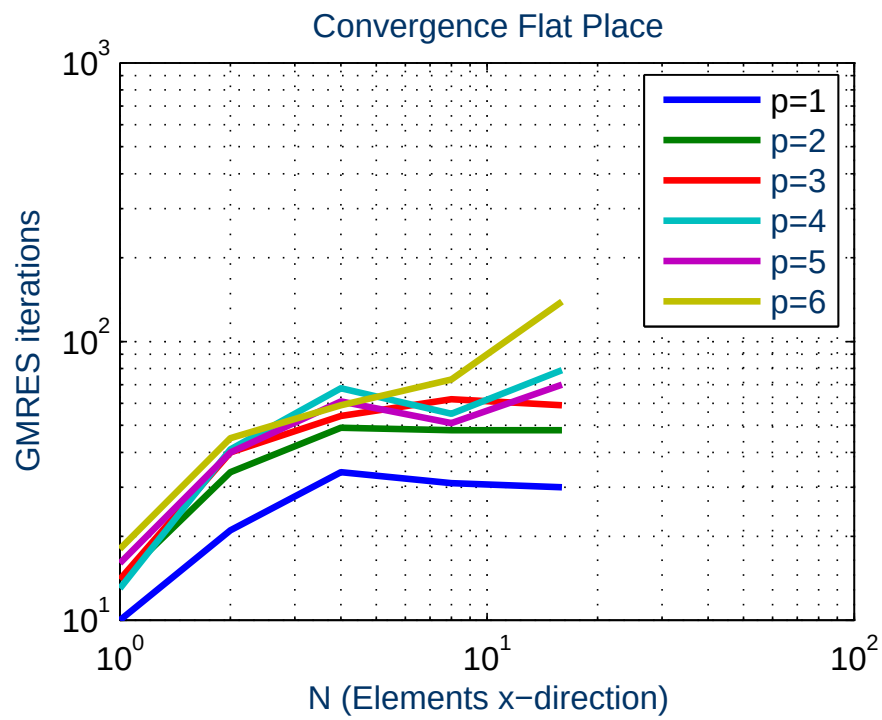
Mach Number Dependence

- The ILU(0)/Coarse scale preconditioner appears to make convergence almost independent of Mach number
- Individually, ILU(0) and coarse corrections perform poorly for low Mach
- Clearly, ILU(0) is an efficient smoother and reduces errors in high frequencies (within *and* between the elements)



ILU(0)/Coarse scale, h and p Dependence

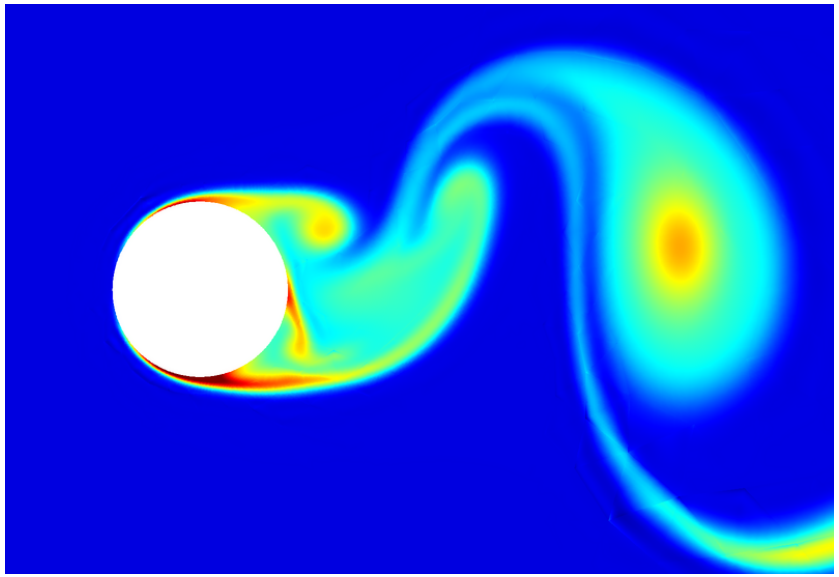
- Flat plate model problem, N-by-N elements
- #GMRES iterations grows approximately linearly with N
  - Expected: Diameter of problem is N (no h-multigrid)
- #GMRES iterations grows with p
  - Expected: No full p-multigrid
- Performance still competitive because of the low cost



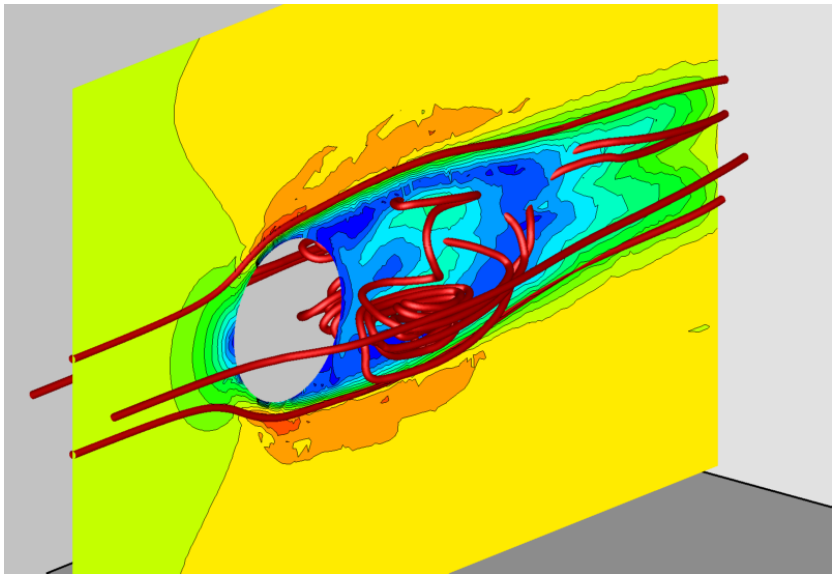
Examples

- We have successfully applied our solver/preconditioner to:
  - Two- and three-dimensional flows
  - Large range of timesteps, including steady-state solutions
  - Highly graded meshes with RANS turbulence modeling
  - Problems with shocks, captured using sensors and artificial viscosity

Cylinder, Re=2000 (Entropy)



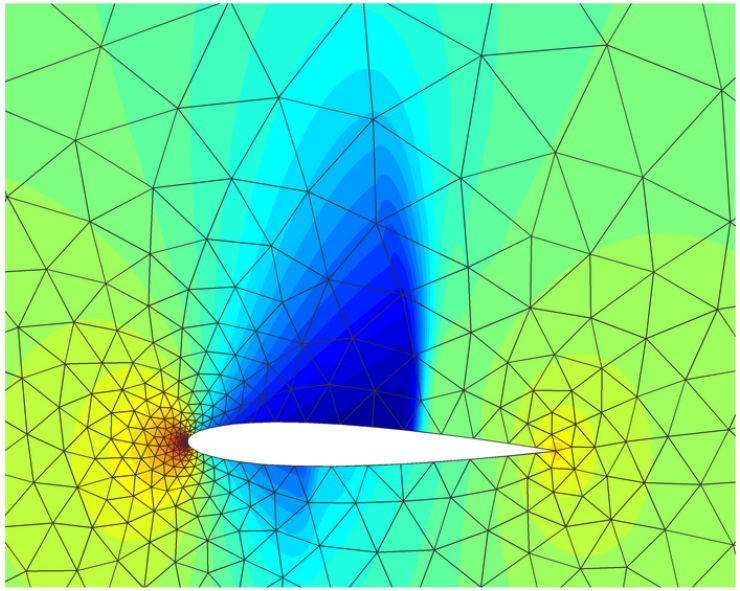
Sphere, Re=1000 (Mach & Velocity)



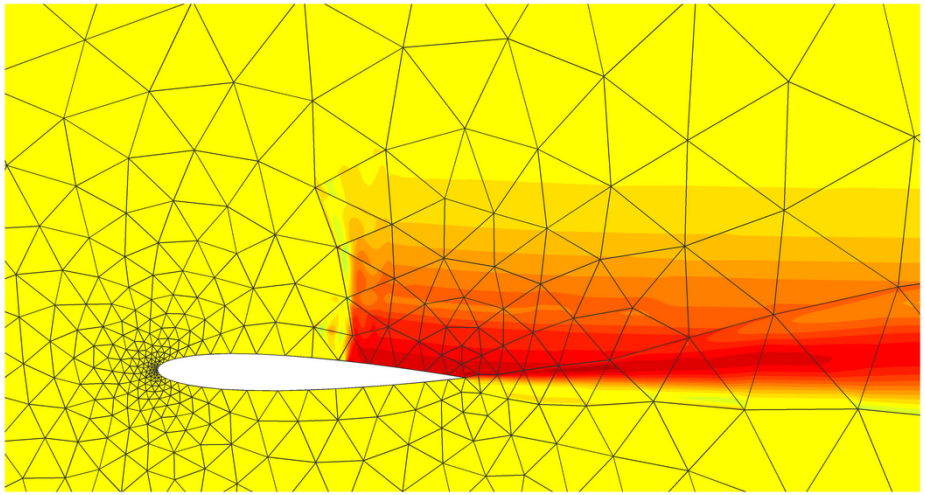
Example: Transonic Flow

- NACA 0012, 1.5° attack angle
- $M = 0.8$ and $p = 4$
- Implicit steady-state solution with Newton solver

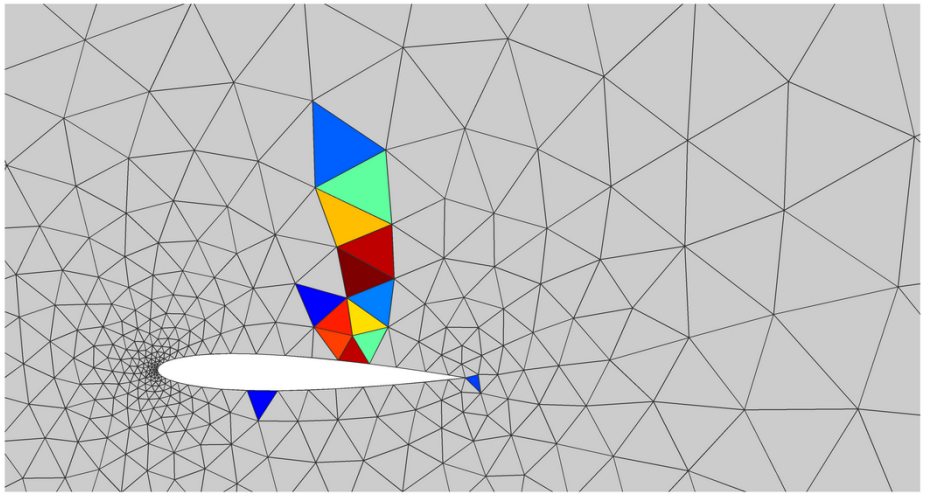
Pressure



Entropy



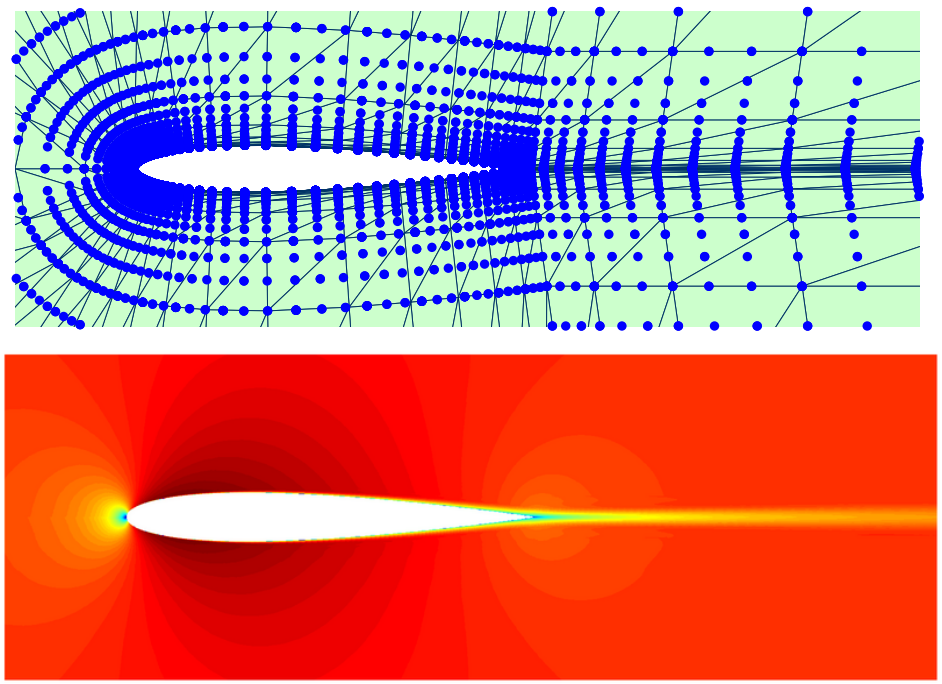
Sensor



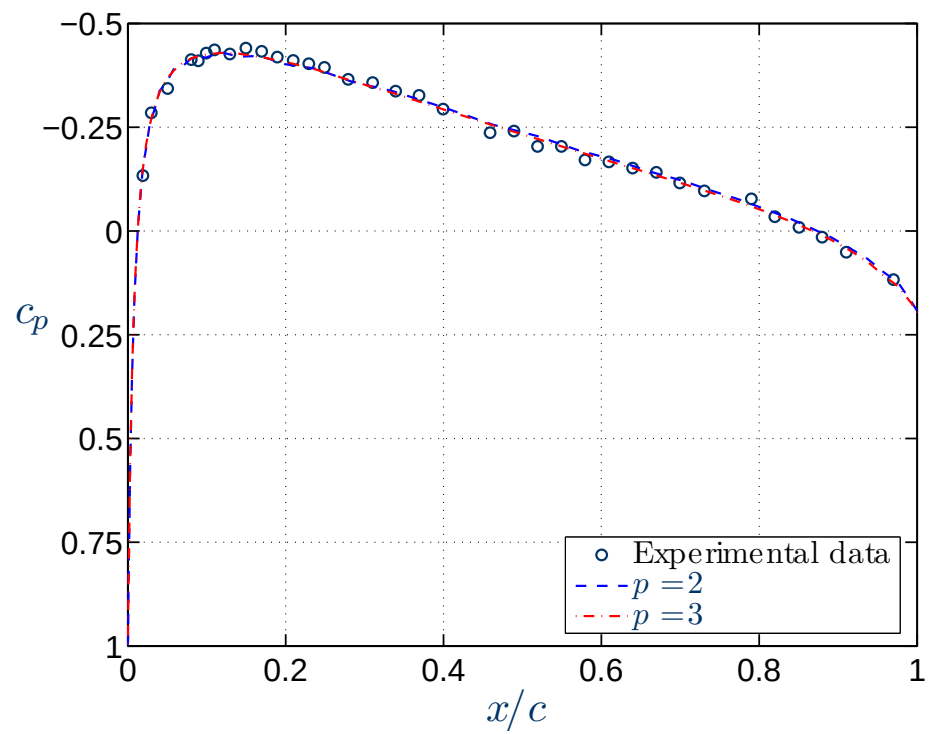
Example: Turbulent Flow over NACA 0012 Airfoil

- Spalart-Allmaras turbulence modeling, $M = 0.3$, $\mathrm{Re} = 1.86 \cdot 10^6$
- p-convergence and good agreement with experimental data

Mesh and Mach



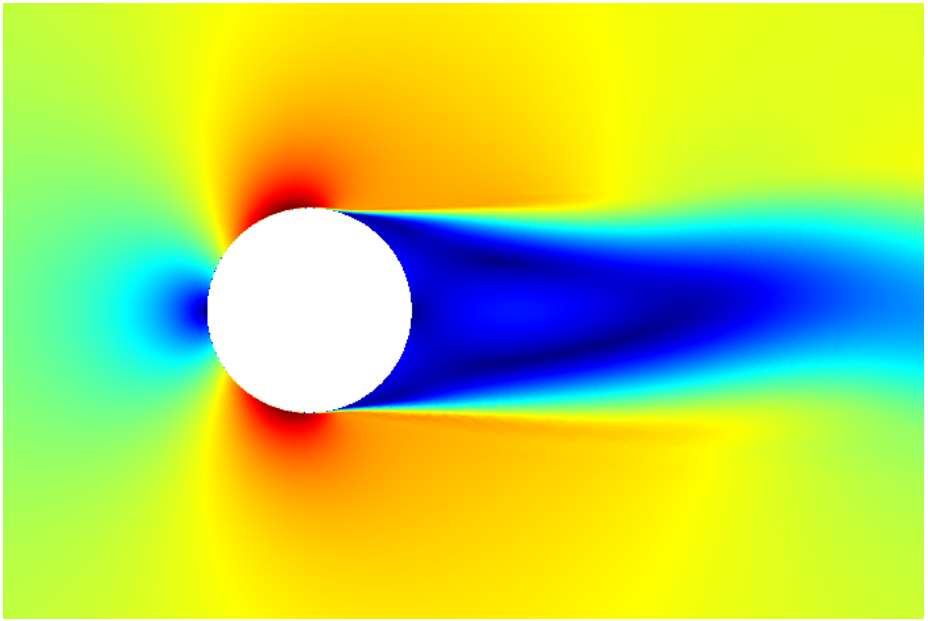
Pressure Coefficient



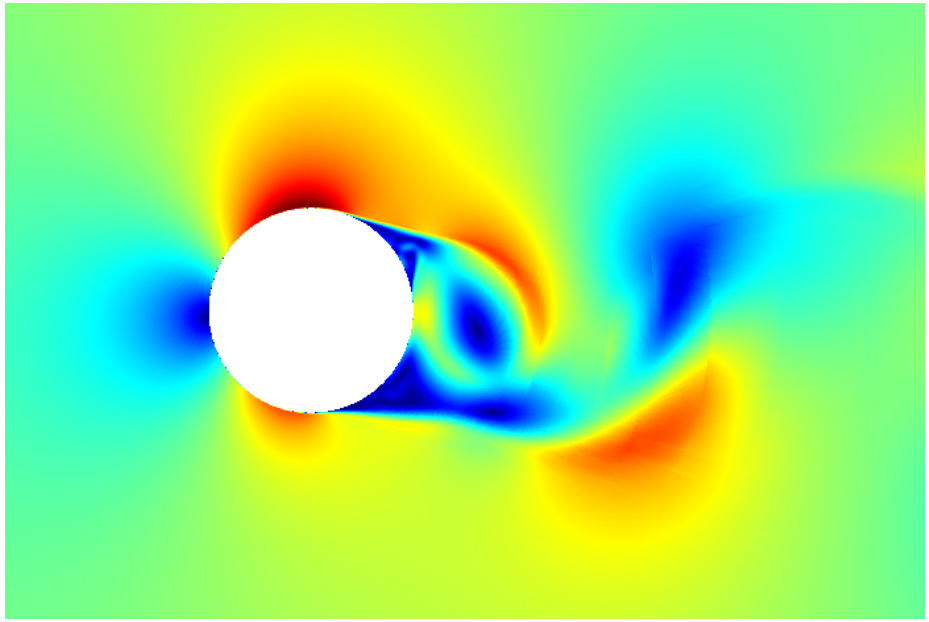
Example: Unsteady Turbulent Flow over Cylinder

- Spalart-Allmaras turbulence modeling, $M = 0.2, \mathrm{Re} = 5 \cdot 10^5, p = 2$
- LES in 2-D not physically meaningful, but comparison shows how URANS smears out complex flow features

Mach, URANS



Mach, DES



Conclusions

- Important steps toward an industrial DG solver:
  - Low-cost compact viscous discretization
  - Simple and effective ILU/coarse scale preconditioner
  - Block matrix storage and operations using high performance libraries
- Current status:
  - Flexible, lightweight, and highly efficient n-D code fully developed
  - Implicit and explicit solvers, various Krylov solvers and precondtitioners
  - Handles large 2-D problems and moderate size 3-D problems in serial
- Current work:
  - Parallelization of code, implementation of LES/DES capabilities, solution of large scale 3-D problems