Preconditioners for Discontinuous Galerkin Discretizations of 3D viscous compressible flows

Philipp Birken
Mark Haas
Gregor Gassner, Claus-Dieter Munz

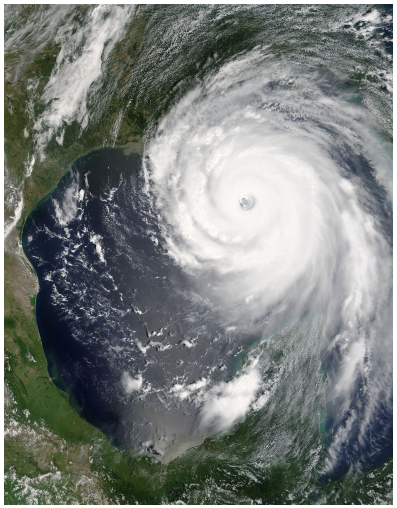
University of Kassel (SFB/TRR 30)
University of Stuttgart (Mark now at Bosch GmbH)

October 9th 2013
Efficient solution of large systems of non-linear PDEs in science

Outline

Outline

Motivation



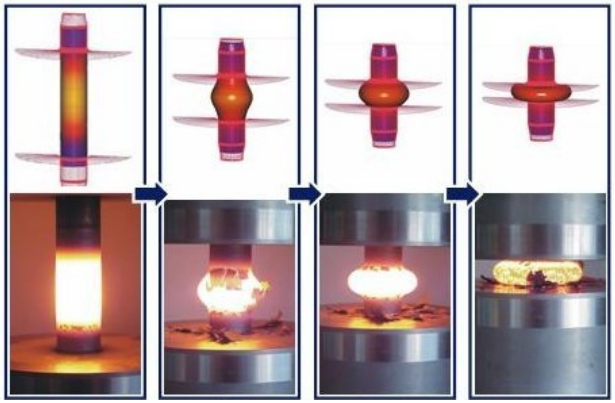
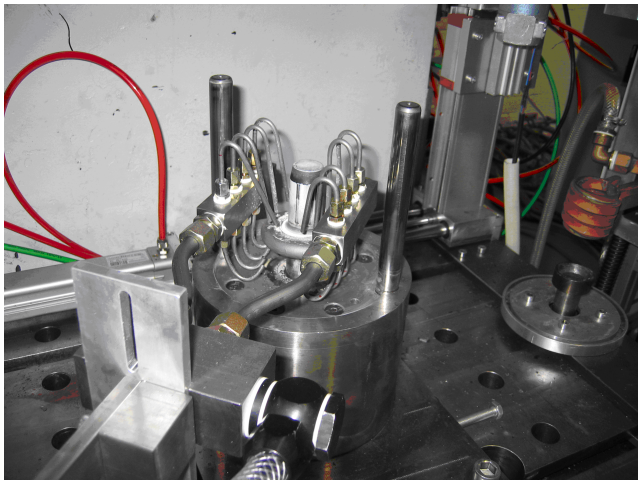



Figure: Hurricane Catrina, NASA, PD (left), Lillgrund Offshore windfarm, Mariusz Padziora, CC-by-sa 3.0 (right); Gas Quenching, Steinhoff

Compressible Navier Stokes equations

Second order system of conservation laws (mass, momentum, energy) modeling viscous compressible flow:

$$
\begin{aligned}
\partial_t \rho + \nabla \cdot \mathbf{m} &= 0, \\
\partial_t m_i + \sum_{j=1}^{3} \partial_{x_j}(m_i v_j + p\delta_{ij}) &= \frac{1}{Re}\sum_{j=1}^{3} \partial_{x_j} S_{ij}, \qquad i = 1,2,3 \\
\partial_t(\rho E) + \nabla \cdot (H\mathbf{m}) &= \frac{1}{Re}\sum_{j=1}^{3} \partial_{x_j}\left(\sum_{i=1}^{3} S_{ij} v_i - \frac{1}{Pr} W_j\right),
\end{aligned}
$$

Equation of state: $p = (\gamma - 1)\rho e$.

$$u_t + \nabla \cdot f(u, \nabla u) = 0.$$

Navier-Stokes-Solvers

- Consider unsteady 3D compressible viscous flow problems
- Important: Turbulence, Boundary Layer, Mach number
- Discretization in space via FV or DG
- Leads to initial value problem in time
- $\rightarrow$ huge number of unknowns, problem is typically stiff.
- $\rightarrow$ implicit time integration necessary.
- Goal: Fast implicit 3D solver in the context of DG!

Overview: B., *Numerical methods for the unsteady compressible Navier-Stokes equations*, Habilitation Thesis, 2012, University of Kassel

Parallel Solvers

Form of equation is independent of precise time integration scheme

$$\mathbf{u} = \alpha \Delta t \mathbf{f}(\mathbf{u}) + \psi.$$

$\rightarrow$ Need for fast low storage parallel scaling solvers Candidates:

- Preconditioned inexact Jacobian-Free Newton-Krylov (JFNK)
- FAS (multigrid) with appropriate smoothers

Figure: Cray Hermit in Stuttgart; Bild: ThE cRaCkEr, CC-by-sa 3.0, via Wikimedia Commons

Why high order methods?

- Standard for Finite Elements, Finite Volume is 2. order
- Higher order for FV methods in 3D problematic, since a lot of neighboring cells necessary (ENO/WENO)
- Popular approach is DG: Localize high order in cell, but use discontinuous ansatz functions for stability for convection
- Alternative: Use FE with high order and stabilization
- For many turbulent flows necessary to resolve eddies
- Direct numerical simulation (DNS): $\mathcal{O}(Re^3)$ unknowns
- Large Eddy Simulation (LES): $\mathcal{O}(Re^{2.5})$ unknowns
- Efficient LES only imaginable using high order methods
- Goal: Efficient DG method for LES

Finite Volume schemes (FVM)

- Standard schemes in computational fluid dynamics.
- Developed in last 50 years.
- In most simple form first order.
- Higher order in space via linear reconstruction and limiter.
- More than second order not practical.
- For explicit Euler scheme stable for $CFL < 1$.
- Implicit methods necessary for large number of problems and part of 3D production solvers.
- Implicit methods magnitudes faster than explicit ones.

Fast implicit Discontinuous Galerkin solvers?

- For first order identical to FVM.
- Much higher orders p possible using piecewise polynomials.
- Huge number of different approaches.
- For explicit scheme stable for $CFL = O(\frac{1}{2p-1})$.
- Implicit methods thus even more necessary.
- In 2D no fast implicit method available.
- In 3D additional difficulties from memory requirements.
- Is it possible to get a fast implicit DG scheme in 3D?
- Determines applicability of DG in industry!

How do different time integration schemes compare for DG?

The story so far

In the implicit context...

- ...quite a number of papers on steady Euler and NS

Less so for the unsteady case:

- Klaij, van der Vegt, van der Wen 06, 2D-NS
- Wang, Mavriplis 07, 2D-Euler
- Kanevsky, Carpenter, Gottlieb, Hesthaven 07, 2D-NS
- Persson, Peraire 08, 2D-NS
- St.-Cyr, Neckels 09, 2D-Euler
- Dolejsi, Holik, Hozman 11, 2D-NS
- Uranga, Persson, Drega, Peraire 11, 3D-NS
- Birken, Gassner, Haas, Munz 13, 3D-NS

Implicit methods for unsteady 3D NS still need some work!

Comparing time integration schemes

In this talk, I'll focus on comparing time integration schemes.

Need to fix

- Specific DG scheme
- Specific solver for systems in implicit time integration
- Find a fair way of comparing schemes

Outline

Eigenvalues

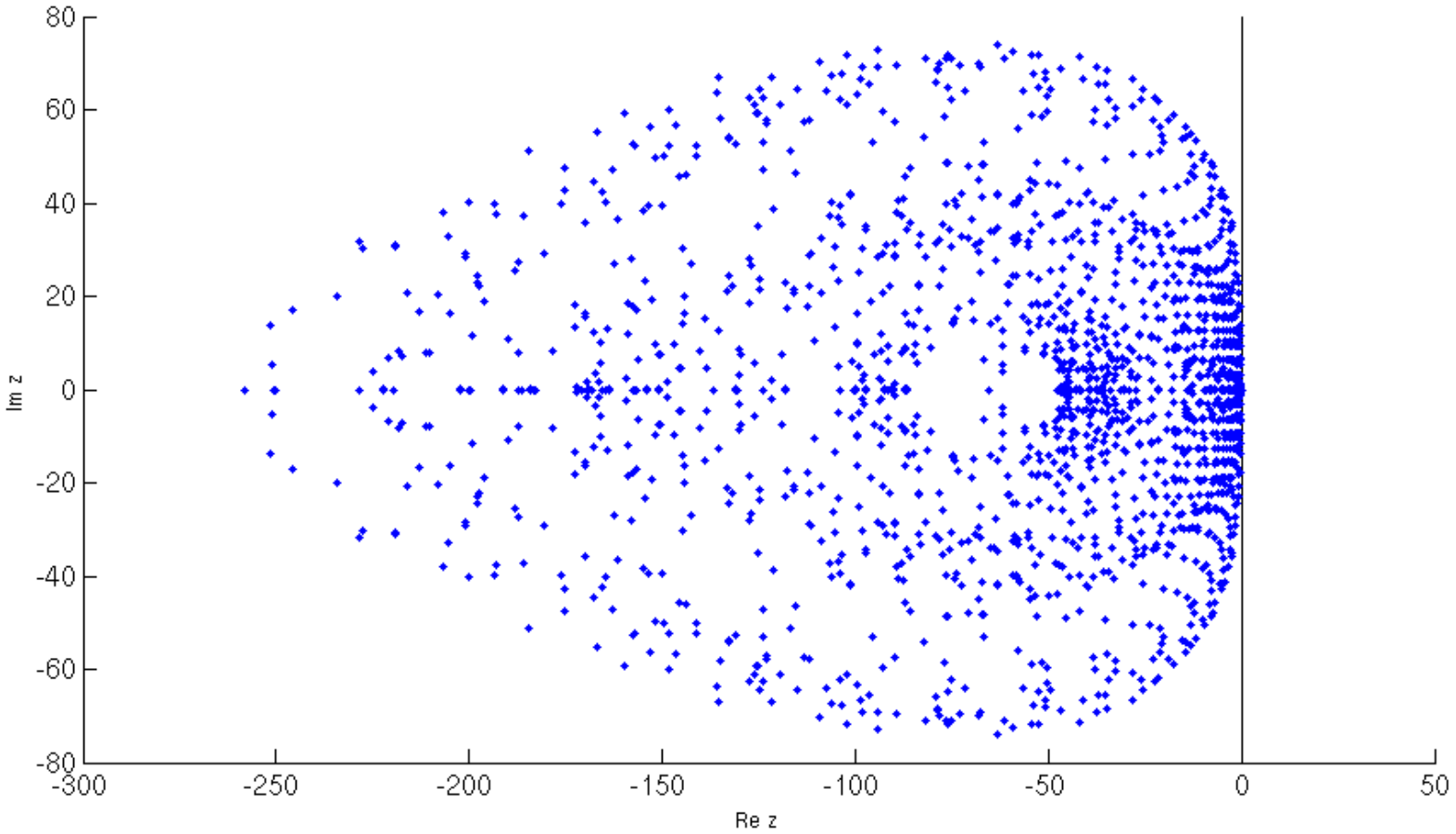


Figure: Re=100, 4th degree polynomial

Implicit time integration

- Large stability region required: A-stable methods!
- Unsteady problem: Higher order and time adaptivity.
- BDF often used, but does not fit the profile.
- Bijl et al (01,02): ESDIRK methods competitive!
- One explicit and subsequent backward Euler steps.
- Time adaptivity easy.

Time integration schemes used

Explicit schemes

- LSERK4: 5-stage explicit RK, 4th order, large stability region (Carpenter, Kenndy)
- RKCK: local time stepping scheme using CERK methods to predict solution locally in each cell, then correct according to a higher order Space-time expansion via the Cauchy-Kovalevskaya procedure (Gassner, Dumbser, Hindenlang, Munz 11).

Implicit schemes

- ESDIRK4, ESDIRK3, SDIRK3
- ROS34PW2, Rosenbrock method, linearized SDIRK3 (Rang, Angermann 05).

Tolerance scaling and time adaptivity

- Given TOL, determine new time step based on

$$d_i = TOL|u_i^n| + TOL$$

$$\Delta t_{new} = \Delta t_n \cdot \|\hat{\mathbf{l}}./\mathbf{d}\|^{-1/(\hat{p}+1)}.$$

- We can prove that $\|e\| \to 0$ for TOL$\to 0$.
- For TOL towards zero, we observe

$$\|\mathbf{e}\| = \tau \cdot TOL^{\alpha}$$

with $\alpha < 1$ method dependent.

- Tolerance scaling (Söderlind, Wang '06): Rescaling via

$$TOL' = TOL_0^{(\alpha - 1/\alpha)} TOL^{1/\alpha}$$

such that for one value $TOL_0 = TOL' = TOL$.

- DIRK schemes: $\alpha = 0.9$, ROS34PW2: $\alpha = 0.8$.

Outline

DG method: Weak form

- Chose grid and element types (teds, quadrilaterals, prisms,...)
- Start with:

$$\tilde{u}_t + \nabla \cdot \tilde{f}(\tilde{u}, \nabla \tilde{u}) = 0.$$

- Determine solution using Galerkin condition

$$(\tilde{u}_t, \phi) + (\nabla \cdot \tilde{f}, \phi) = 0$$

- Integration by parts:

$$(\tilde{u}_t, \phi) + \int_{\partial\Omega} \tilde{f} \cdot n dS - (\tilde{f}, \nabla\phi) = 0$$

- Transform every cell to unit cell E:

$$(u_t, \phi) + \int_{\partial E} f \cdot n dS - (f, \nabla\phi) = 0$$

DG: Polynomial Ansatz

- Use polynomials of degree $p-1$ for trial and solution space:

$$u^P(x,t) = \sum_j u_j(t)\phi_j(x).$$

- Two types of basis for polynomial space possible
  1. Lagrange-type, thus based on nodes (nodal)
  2. Monomial-type, thus based on monomials (modal)
- Solution at cell boundary discontinuous by construction.
- Approximate boundary integrals using numerical fluxes
- Takes ideas from FE and FV world

Computation of the fluxes

- Convective terms using HLLC.
- Diffusive terms more difficult.
  - Standard averaging procedure from FVM unstable for DG.
  - Use dGRP (Gassner, Lörcher, Munz 06).
- Use Gaussian quadrature for scalar products, obtain

$$\mathbf{M}\mathbf{u}' + \sum_{i=1}^{nFaces} \mathbf{M}_i^S \mathbf{g_i} - \sum_{k=1}^{d} \mathbf{S}_k \mathbf{f}_k = \mathbf{0}.$$

- Matrices and vectors depend on basis, quadrature rule, grid and fluxes

Polymorphic Modal-Nodal method

- Here: Polymorphic modal method with nodal integration (Gassner, Lörcher, Munz, Hesthaven 09)
- Unstructured cells with curved boundaries (Tets, Quads,...).
- Hierarchical orthonormalized monomial basis on reference cells

$$x_1^0 x_2^0 x_3^0, \quad x_1^1 x_2^0 x_3^0, \; x_1^0 x_2^1 x_3^0, \; x_1^0 x_2^0 x_3^1, \quad \ldots$$

- Use different (nodal) basis for integration for fast scheme!
- Convective terms using HLLC.
- Diffusive terms with dGRP (Gassner, Lörcher, Munz 06) (small stencil).

Newton and Multigrid

Steady States

- Multigrid fast
- Newton slow

Unsteady problems

- Here we expect significantly faster convergence
- Multigrid marginally, if at all, faster, not fast on unstructured grids
- Newton significantly faster than for steady state

Still: Existing solvers for unsteady problems not fast!

Improving the existing solvers

Multigrid

- Dual time stepping typically implies reusing the steady state algorithm without changes
- Thus less than optimal multigrid convergence
- Redesign multigrid (see B., *Optimizing Runge-Kutta smoothers for unsteady flow problems*, ETNA, 2012)

Newton

- Very good scheme: Jacobian-Free Newton-GMRES with good parallel preconditioner
- Multigrid candidate for preconditioner (see above)

Compare *Solving nonlinear systems inside implicit time integration schemes for unsteady viscous flows*, P. Birken, pp. 57-71 in R. Ansorge, H. Bijl, A. Meister, T. Sonar (editors), Numerics of Nonlinear Hyperbolic Conservation Laws, Notes on Numerical Fluid Mechanics, Springer

Inexact Jacobian-free Newton-Krylov method

Solve nonlinear systems using Newton's method.

- Iterate until relative TOL/5 satisfied.
- Linear equation systems solved using GMRES.
- Tolerances in GMRES by forcing terms of Eisenstat/Walker.
- GMRES does not need Jacobian, only matrix vector products.
- Approximate $\frac{\partial \mathbf{F}}{\partial \mathbf{u}} q$ in GMRES by finite differences:

$$\mathbf{A}\mathbf{q} = \frac{\partial \mathbf{F}(\mathbf{u}^{(k)})}{\partial \mathbf{u}}\mathbf{q} \approx \frac{\mathbf{F}(\mathbf{u}^{(k)} + \epsilon \mathbf{q}) - \mathbf{F}(\mathbf{u}^{(k)})}{\epsilon}.$$

- Immense flexibility
- Method is quadratically convergent in large radius!

Feedback-Loops for DIRK scheme

Given error tolerances τ, initial time t_0 and time step size Δt_0

- For $i = 1, ..., s$
  - For $k = 0, 1, ...$ until termination criterion with tolerance $\tau/5$ is satisfied or MAX_NEWTON_ITER has been reached
    - Determine Eisenstat-Walker relative tolerance
    - Solve linear system using preconditioned GMRES
- If MAX_NEWTON_ITER has been reached, but the tolerance test has not been passed, repeat time step with $\Delta t_n = \Delta t_n/4$
- Estimate local error and compute new time step size Δt_{n+1}
- $t_{n+1} = t_n + \Delta t_n$

Note: Puts additional bound on time step via nonlinear solver

The linear system: 2D is not 3D, FV is not DG!

- System matrix: $\mathbf{A} = \left[\frac{\mathbf{I}}{\Delta t} - \frac{\partial \hat{\mathbf{f}}(\mathbf{u})}{\partial \mathbf{u}} \right] |_{\mathbf{u}^{(k)}}$.
- Sparse, nonnormal, not diagonally dominant and ill conditioned for reasonable Δt.
- Generally unstructured, but symmetric block-sparsity pattern.
- In FVM context: block sizes of 5×5 in 3D.
- Here: Block sizes depend on degree N and dimension:
  - $(d+2) \cdot (N+d)!/(N!d!)$
- This makes the design of efficient implicit DG schemes in 3D so difficult, since blocks are huge! Degree 5: 280×280!
- Worse for DG-SEM case ($(d+2) \cdot (N+1)^d$, degree 5: 1080×1080)

Preconditioning

- Use right preconditioning, since residual is unchanged (inexact JFNK)

$$\mathbf{APy} = \mathbf{b}, \quad \mathbf{y} = \mathbf{P}^{-1}\Delta\mathbf{u}$$

- In a JFNK scheme, we do not compute the matrix a priori.
- Need to compute all parts of matrix needed for preconditioner.
- Interesting: Two-level ILU from Persson, Peraire 08. Uses a coarse scale correction via Jacobi (ILU-CSC)
- Jacobi:

$$\mathbf{P} = \mathbf{D}^{-1}$$

- Parallel, low storage, not very accurate
- Gauß-Seidel:

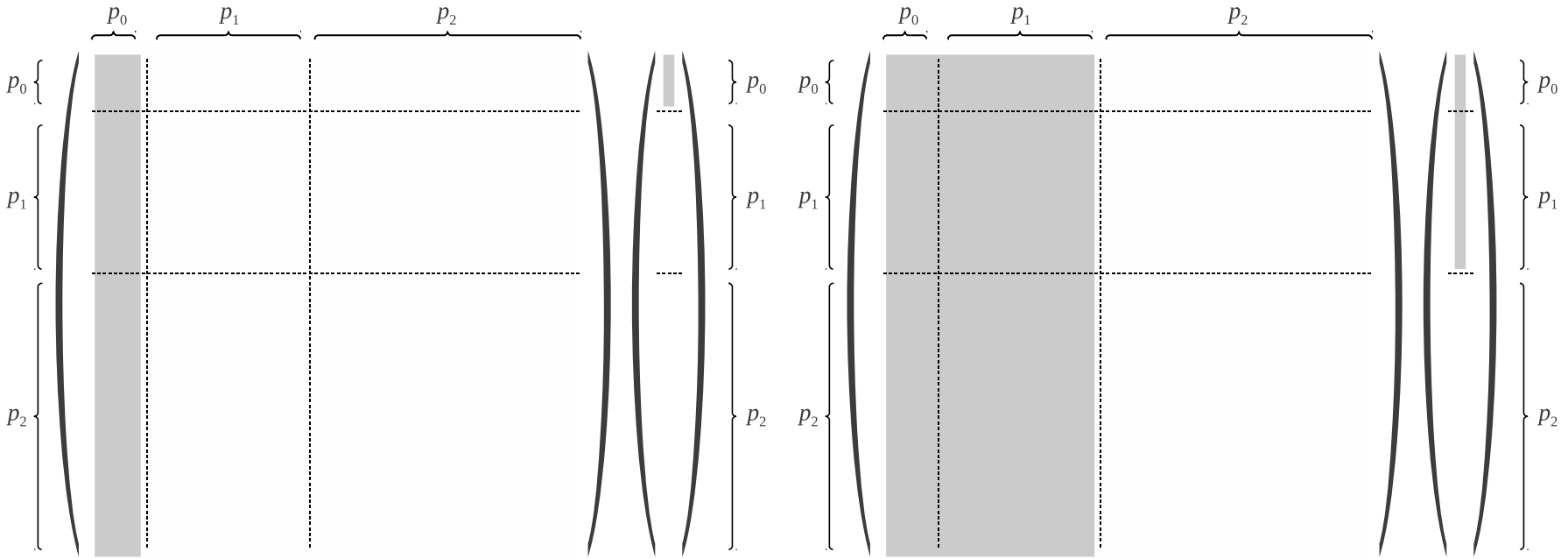
$$\mathbf{P} = (\mathbf{D} + \mathbf{L})\mathbf{D}^{-1}(\mathbf{D} + \mathbf{U})$$

- Sequential, stores full matrix, accurate

Alternative: ROBO-SGS!

- Common in FV: Compute first order matrix only
- Here: Make use of hierarchical basis
- Compute offdiagonal blocks of lower order only in SGS!
- **R**educed **O**ff-diagonal-**B**lock-**O**rder-**SGS**
- Saves setup time and application cost!
- In parallel, no communication between blocks for preconditioner.
- Birken, Gassner, Haas, Munz, JCP 2013
- Note: Idea works for any high order method employing a hierarchical basis

ROBO-SGS: Off-Block-Sparsity patterns



(a) $p = 0$ variant

(b) $p = 1$ variant

Figure: Reduced versions of the off-block Jacobians, $p = 0$ and $p = 1$ variants

Outline

Flow around a cylinder, Ma=0.3, Re=1000

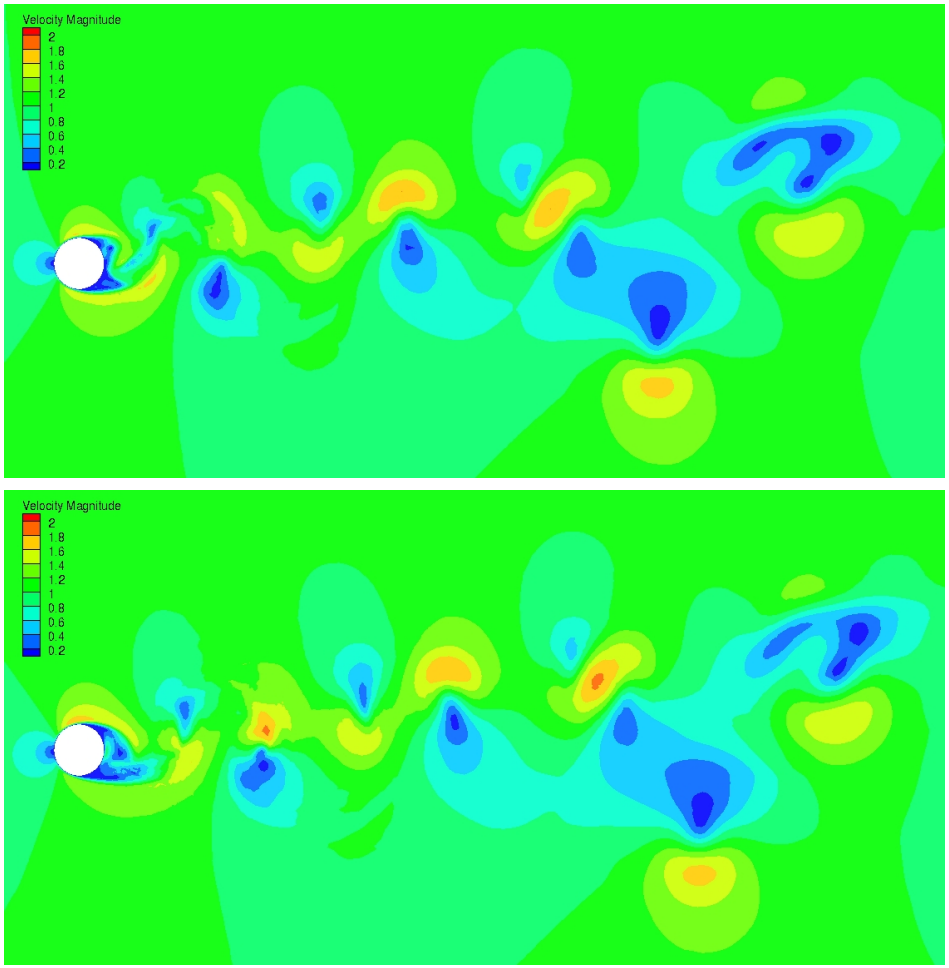


Figure: Initial (top) and final (bottom) velocity magnitude for cylinder problem. $10,400$ hexahedral cells, order 6, $2,912,000$ unknowns.

Comparison of preconditioners

Preconditioner	Iter.	CPU [s]	Comparison to Jacobi [%]
No preconditioner	8,797	2,194	36.0
Jacobi	3,712	1,613	0.0
ROBO-SGS-0	3,338	1,538	-4.6
ROBO-SGS-1	2,824	1,429	-11.4
ROBO-SGS-2	2,656	1,485	-7.9
ROBO-SGS-3	2,641	1,679	4.1
ROBO-SGS-4	2,645	1,989	23.3
ROBO-SGS-5	2,640	2,427	50.5
ILU(0)	2,641	2,467	52.9
ILU(0)-CSC	2,640	2,994	85.6

Table: Computations on 64 cores of the CRAY XE6 cluster *Hermit*.

Parallel scaling

		ROBO-SGS-1			Jacobi		
Cores	Unkn./core	Iter.	Time	Scaling	Iter.	Time	Scaling
64	45,500	2,824	1,429	-	3,712	1,613	-
128	22,750	2,926	750	95%	3,712	833	97%
256	11,375	3,031	395	90%	3,712	432	93%
512	5,688	3,479	230	78%	3,712	231	87%
		ILU			ILU-CSC		
Cores	Unkn./core	Iter.	Time	Scaling	Iter.	Time	Scaling
64	45,500	2,641	2,467	-	2,640	2,994	-
128	22,750	2,641	1,196	103%	2,640	1,477	101%
256	11,375	2,647	576	107%	2,640	728	103%
512	5,688	2,713	287	107%	2,679	383	98%

Table: Parallel scaling for cylinder test case on Cray Hermit.

- Scaling very good even for very small number of unknowns per core
- Iteration number almost constant

Flow around sphere, Re=1000, Ma=0.3

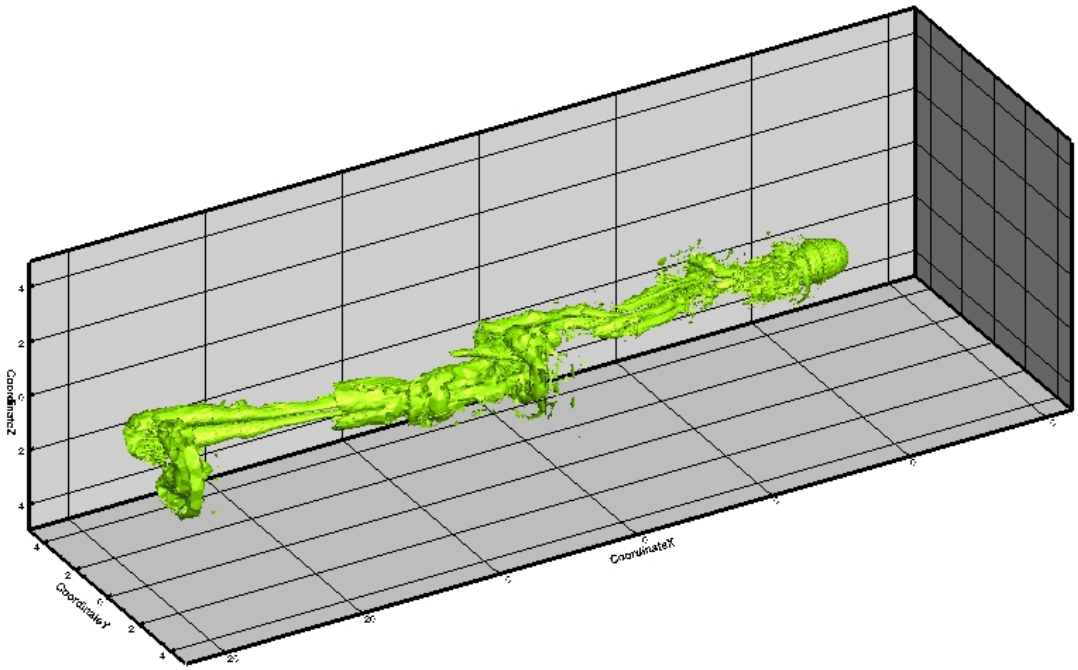


Figure: Isosurfaces of lambda-2$=-10^{-4}$ for $t = 30s$. grid has 21.128 elements, 739,480 unknowns, $TOL = 10e^{-3}$

Comparison of time integration schemes

Scheme	Iter.	CPU in s
LSERK4	-	346,745
RKCK	-	80,889
SDIRK3	22,223	110,844
ESDIRK3	14,724	73,798
ESDIRK4	14,639	66,051
ROS34PW2	60,449	239,869

Table: Efficiency for flow around sphere, ROBO-SGS-1.

Summary and Conclusions

- Implicit modal DG scheme for unsteady 3D viscous flows.
- FV is not DG! 2D is not 3D!
- Solver: JFNK with ROBO-SGS preconditioner.
- ROBO-SGS interesting alternative to Jacobi for moderate degree of parallelism
- Use tolerance scaling to make time integration schemes comparable
- LSERK4 and ROS34PW2 not competitive
- RKCK and ESDIRK schemes winners
- This test case: ESDIRK4 fastest.