

The Solver Bottleneck: Past, Present, and Future of Scalable Solvers for High-Order Methods

Per-Olof Persson

Department of Mathematics, University of California, Berkeley
Mathematics Department, Lawrence Berkeley National Laboratory

with Y. Pan, M. Lindsey, A. Narayanan, A. Nayak, A. Shi, M. Zahr, W. Pazner, B. Froehle, L. Wang, J. Peraire

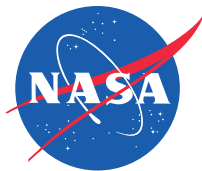
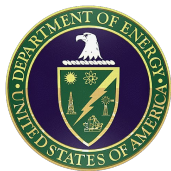


GAMM 2026, Stuttgart, Germany
May 20, 2026



Acknowledgements

This work has been funded by multiple organizations for 15 years.



Thanks for the support from the Stuttgart Center for Simulation Science (SC SimTech) through the Argyris Visiting Professorship 2024.



Outline

- 1 Introduction and Motivation
- 2 DG Jacobian Structures and Sparse Variants
- 3 The Current Standard: ILU-Smoothed Multigrid Solvers
- 4 Recent Results in Matrix-Free, Elimination-Based, and Multigrid Solvers
 - Kronecker-SVD Preconditioners
 - Static Condensation for DG
 - Agglomerated Multigrid
 - Geometric Adaptive Smoothed Aggregation Multigrid

Outline

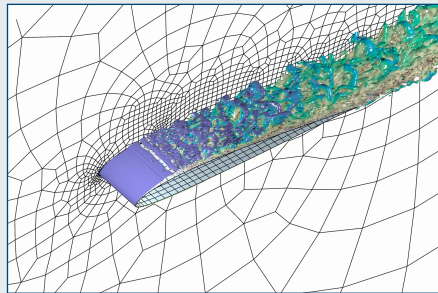
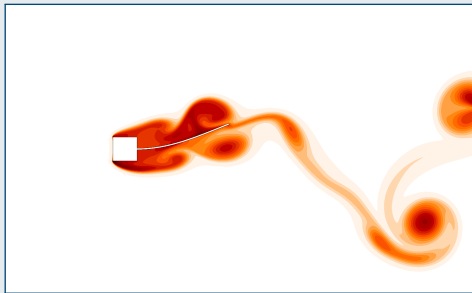
- 1 Introduction and Motivation
- 2 DG Jacobian Structures and Sparse Variants
- 3 The Current Standard: ILU-Smoothed Multigrid Solvers
- 4 Recent Results in Matrix-Free, Elimination-Based, and Multigrid Solvers
 - Kronecker-SVD Preconditioners
 - Static Condensation for DG
 - Agglomerated Multigrid
 - Geometric Adaptive Smoothed Aggregation Multigrid

High-Order Methods for Complex Physics

Why High-Order?

High-order discretizations (DG, SEM) provide **spectral-like resolution** on unstructured grids:

- **Wave-Dominated Physics:** Low dissipation/dispersion for turbulence and acoustics.
- **Precise Structural Dynamics:** Capturing thin-shell bending and complex FSI.
- **Modern Hardware Efficiency:** High arithmetic intensity for GPU acceleration.
- **Optimal Convergence:** Exponential error decay with minimal degrees of freedom.



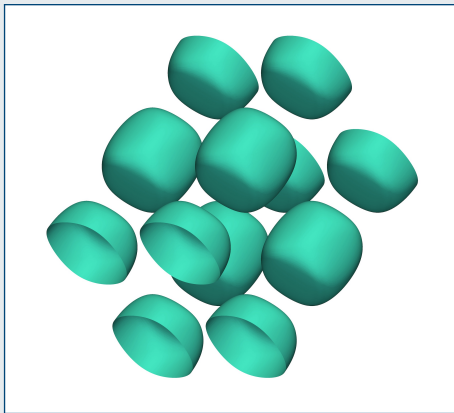
High-Order Methods for Complex Physics

Example: DNS of Compressible Taylor-Green Vortex

- Challenge problem from 3rd High-Order Workshop
- Transitional flow, turbulent decay

Vorticity magnitude contours,
colored by helicity

$$t = 0$$



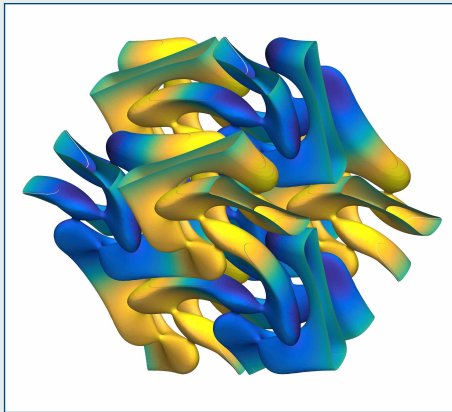
High-Order Methods for Complex Physics

Example: DNS of Compressible Taylor-Green Vortex

- Challenge problem from 3rd High-Order Workshop
- Transitional flow, turbulent decay

Vorticity magnitude contours,
colored by helicity

$$t = 2$$



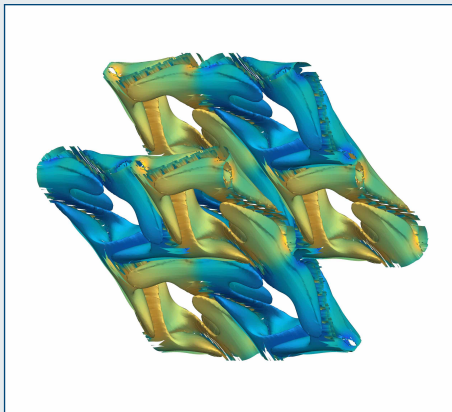
High-Order Methods for Complex Physics

Example: DNS of Compressible Taylor-Green Vortex

- Challenge problem from 3rd High-Order Workshop
- Transitional flow, turbulent decay

Vorticity magnitude contours,
colored by helicity

$$t = 4$$



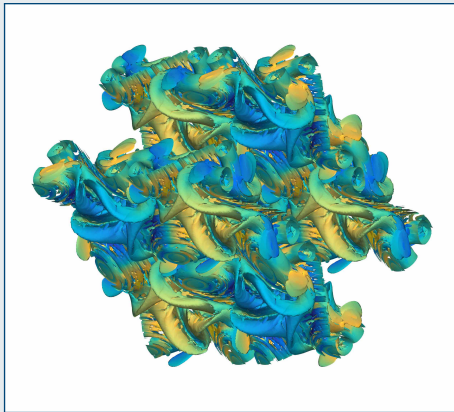
High-Order Methods for Complex Physics

Example: DNS of Compressible Taylor-Green Vortex

- Challenge problem from 3rd High-Order Workshop
- Transitional flow, turbulent decay

Vorticity magnitude contours,
colored by helicity

$$t = 6$$



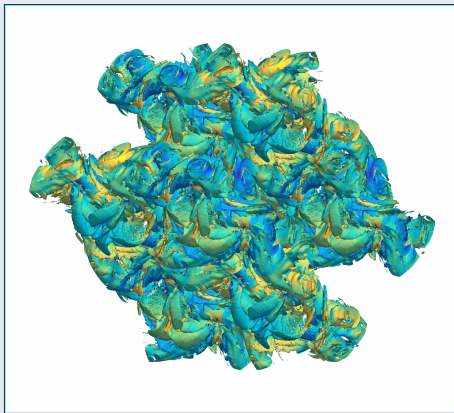
High-Order Methods for Complex Physics

Example: DNS of Compressible Taylor-Green Vortex

- Challenge problem from 3rd High-Order Workshop
- Transitional flow, turbulent decay

Vorticity magnitude contours,
colored by helicity

$$t = 8$$



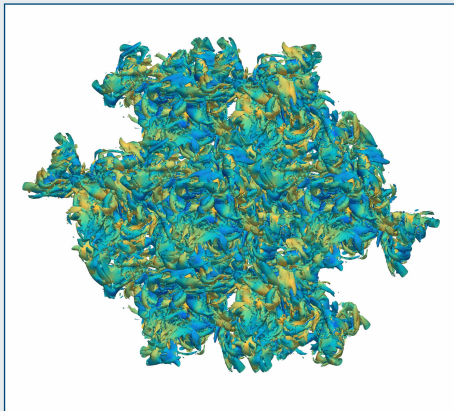
High-Order Methods for Complex Physics

Example: DNS of Compressible Taylor-Green Vortex

- Challenge problem from 3rd High-Order Workshop
- Transitional flow, turbulent decay

Vorticity magnitude contours,
colored by helicity

$$t = 10$$



High-Order Methods for Complex Physics

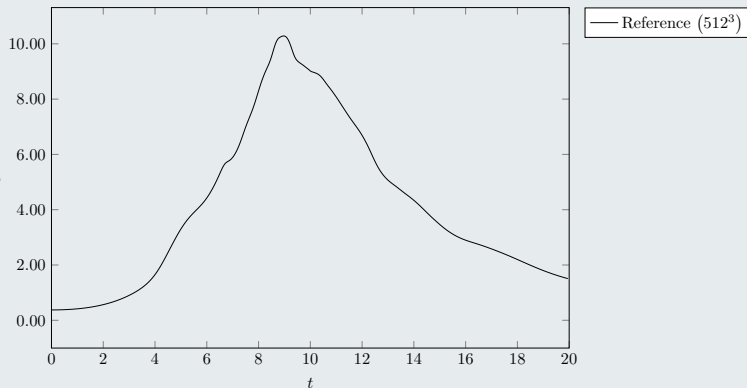
Example: DNS of Compressible Taylor-Green Vortex

- Spectral accuracy as $p \rightarrow \infty$
- Accurate wave propagation
- Low numerical dissipation, long time integration

Enstrophy vs. time

$$\mathcal{E}(t) = \frac{1}{\rho_0 |\Omega|} \int_{\Omega} \rho \frac{\omega \cdot \omega}{2} dx$$

DOFs: 512^3



High-Order Methods for Complex Physics

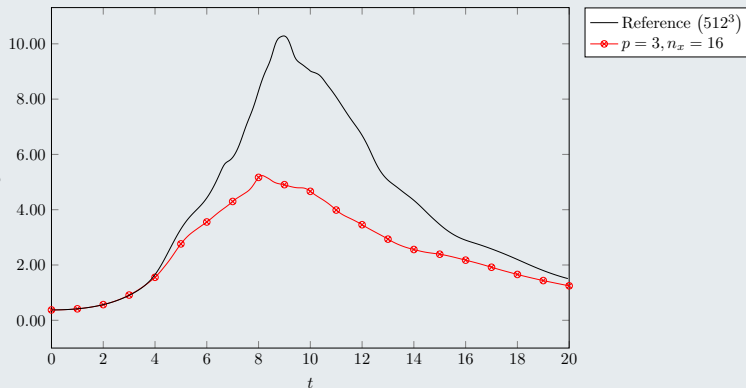
Example: DNS of Compressible Taylor-Green Vortex

- Spectral accuracy as $p \rightarrow \infty$
- Accurate wave propagation
- Low numerical dissipation, long time integration

Enstrophy vs. time

$$\mathcal{E}(t) = \frac{1}{\rho_0 |\Omega|} \int_{\Omega} \rho \frac{\omega \cdot \omega}{2} dx$$

DOFs: 64^3



High-Order Methods for Complex Physics

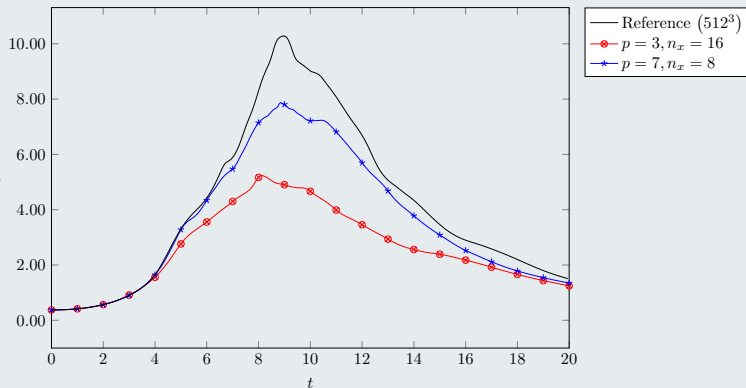
Example: DNS of Compressible Taylor-Green Vortex

- Spectral accuracy as $p \rightarrow \infty$
- Accurate wave propagation
- Low numerical dissipation, long time integration

Enstrophy vs. time

$$\mathcal{E}(t) = \frac{1}{\rho_0 |\Omega|} \int_{\Omega} \rho \frac{\omega \cdot \omega}{2} dx$$

DOFs: 64^3



High-Order Methods for Complex Physics

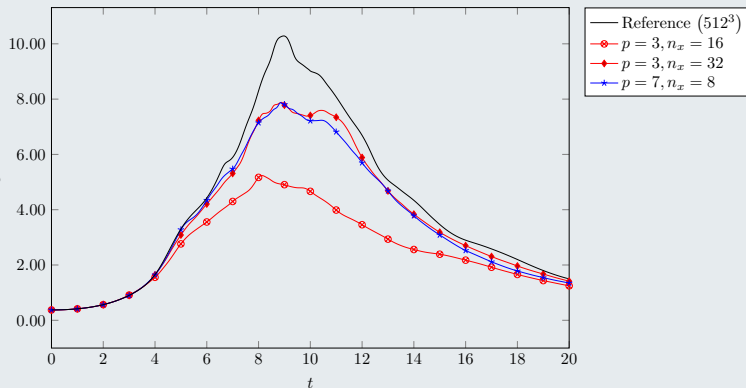
Example: DNS of Compressible Taylor-Green Vortex

- Spectral accuracy as $p \rightarrow \infty$
- Accurate wave propagation
- Low numerical dissipation, long time integration

Enstrophy vs. time

$$\mathcal{E}(t) = \frac{1}{\rho_0 |\Omega|} \int_{\Omega} \rho \frac{\omega \cdot \omega}{2} dx$$

DOFs: 128^3



High-Order Methods for Complex Physics

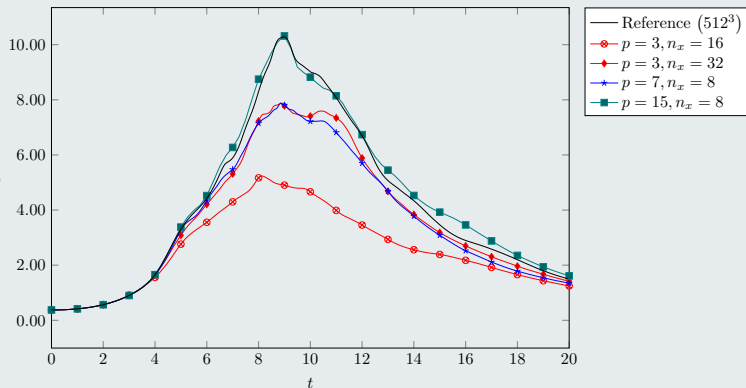
Example: DNS of Compressible Taylor-Green Vortex

- Spectral accuracy as $p \rightarrow \infty$
- Accurate wave propagation
- Low numerical dissipation, long time integration

Enstrophy vs. time

$$\mathcal{E}(t) = \frac{1}{\rho_0 |\Omega|} \int_{\Omega} \rho \frac{\omega \cdot \omega}{2} dx$$

DOFs: 128^3

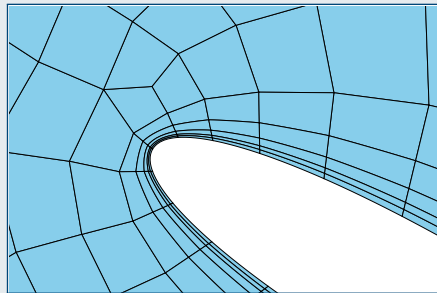
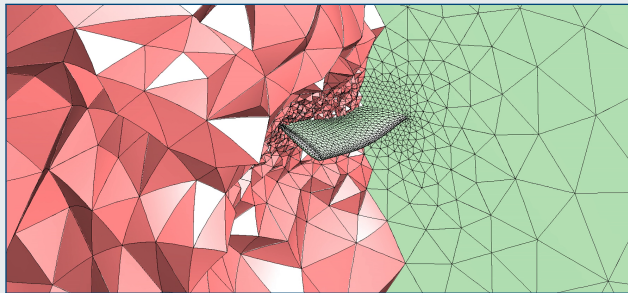


Handling Geometric Complexity

Unstructured & Deforming Meshes

To move from academic test cases to real-world engineering:

- **Complex CAD Integration:** Unstructured grids are essential for industrial geometries.
- **Targeted h -Adaptivity:** Local refinement for sharp features and boundary layers.
- **High-Order Curved Boundaries:** Curvilinear meshes are crucial for high-order accuracy.
- **Dynamic Deforming Domains:** Mapping-based ALE or space-time for large deformations.



High-Order DG for Computational Mechanics

Discontinuous Galerkin (DG) – Fundamental Strengths

- **High Fidelity:** Exceptional accuracy with extremely low numerical dissipation.
- **Geometric Flexibility:** Handles fully unstructured meshes and arbitrary element shapes.
- **Stabilization:** Stabilizes conservation laws without sacrificing the formal order of accuracy.
- **Rigorous Foundation:** Backed by a mature and robust mathematical framework.

Challenges for Real-World Impact

- **Computational Expense:** High cost (per DOF) compared to traditional schemes.
- **Nonlinear Robustness:** Low tolerance for under-resolved features such as shocks.
- **Curved Meshes:** Needs high-quality curved elements, sensitive to “dirty” CAD geometries.

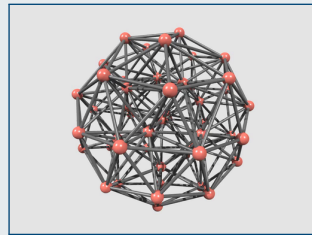
Ultimate Goal

Develop robust, efficient methods to transition high-order DG
from academic research to mainstream engineering tools.

Related work: Mesh Generation

The DistMesh Algorithm

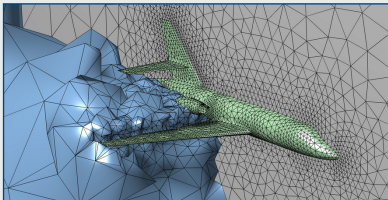
- **Excellent properties:** Simple (one page MATLAB!), implicit geometries (no CAD), high qualities, moving meshes
- **Widely used:** Books, courses, FEM software, C, C++, C#, Python, etc. New Julia version `DistMesh.jl` (2026)!



[1] P.-O. Persson, G. Strang, *SIAM Review*, 2024.

[2] P.-O. Persson, *Ph.D. Thesis, MIT*, 2026.

Curved Mesh Generation



- **Curvilinear mesh generation:** Fully automatic approaches based on nonlinear elasticity [1] and a new FEM formulation of the Winslow equations [2]

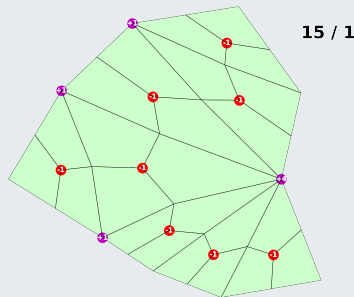
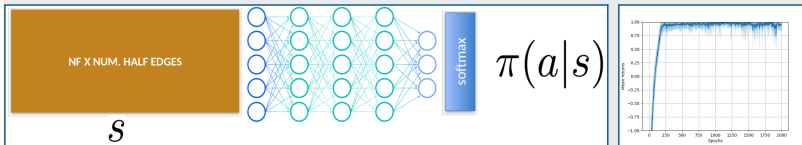
[1] P.-O. Persson, J. Peraire, *AIAA SciTech*, 2009.

[2] M. Fortunato, P.-O. Persson, *J. Comp. Phys.*, 2016.

Related work: Mesh Generation

Deep Reinforcement Learning for Mesh Generation

- Define a “game” for automatic block mesh improvement:
 - “Moves”: Local or global topological operations (e.g. flips)
 - “Score”: Measure of irregularity of the mesh $s = \sum_i |\Delta_i|$
- Use a half-edge mesh structure to define a CNN-type network
- Trained in self-play using Proximal Policy Optimization (PPO)



- [1] Narayanan, Pan, Persson, *Computer-Aided Design*, 2024.
- [2] Narayanan, *Ph.D. thesis*, UC Berkeley, 2024.
- [3] Thacher, Persson, Pan, *Computer-Aided Design*, 2025.

Related work: Moving Domains and Fluid-Structure Interaction

Partitioned FSI using IMEX schemes

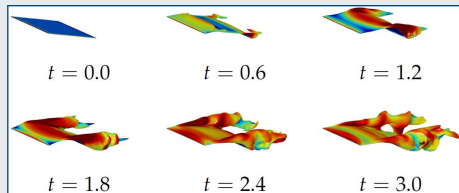
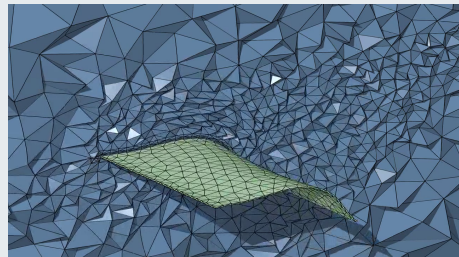
- IMEX schemes to derive FSI partitioning methods
- Treat a **predicted traction** $\tilde{t}$ explicitly and everything else implicitly:

$$r = \begin{bmatrix} r^f(u^f; x(u^s)) \\ r^s(u^s; t(u^f)) \end{bmatrix} = \underbrace{\begin{bmatrix} r^f(u^f; x(u^s)) \\ r^s(u^s; \tilde{t}) \end{bmatrix}}_{\text{implicit}} + \underbrace{\begin{bmatrix} r^{fs}(t(u^f) - \tilde{t}) \end{bmatrix}}_{\text{explicit}}$$

- Use Runge-Kutta based predictor

$$\tilde{t}_{n,i} = \sum_{j=1}^{i-1} \frac{\hat{a}_{ij} - a_{ij}}{a_{ii}} t_{n,j}$$

- DG scheme with mapping-based ALE for fluid
- Volumetric FEM scheme for structures



[1] Froehle, Persson, 2013, 2014.

[2] Huang, Persson, Zahr, 2019.

Related work: Adjoint, PDE-Constrained Optimization

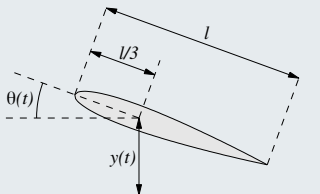
Example: Energetically Optimal Flapping

- Discrete adjoints for sensitivities in time
- Simple derivation, true derivatives
- Enforce periodicity constraints using matrix-free Newton-GMRES
- Example: Maximize energy a fluid exerts on an airfoil during motion

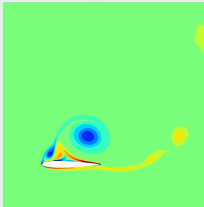
$$\begin{aligned} & \underset{\mu}{\text{minimize}} && - \int_{2T}^{3T} \int_{\Gamma} \mathbf{f} \cdot \dot{\mathbf{x}} \, dS \, dt \\ & \text{subject to} && \int_{2T}^{3T} \int_{\Gamma} \mathbf{f} \cdot \mathbf{e}_1 \, dS \, dt = q \\ & && \mathbf{U}(\mathbf{x}, 0) = \bar{\mathbf{U}}(\mathbf{x}) \\ & && \frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}, \nabla \mathbf{U}) = 0 \end{aligned}$$

[1] Zahr, Persson, 2015.

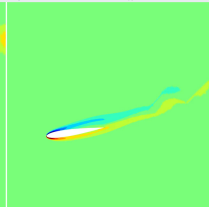
[2] Zahr, Persson, Wilkening, 2015.



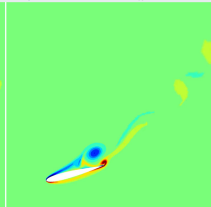
$E = 9.41, I_x = -0.18$



Opt, $E = 0.46, I_x = 0.00$



Opt, $E = 4.95, I_x = -2.50$



Related work: Optimization-Based Shock Tracking

HOIST: High-Order Implicit Shock Tracking

- Optimization approach for aligning curved meshes with discontinuities such as shocks
- Error estimator based on a p -adaptive residual:

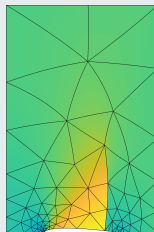
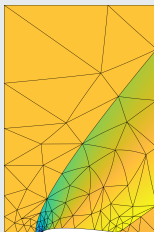
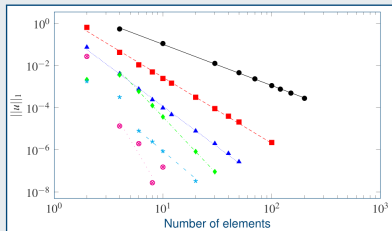
$$r_{h,p'}^K(U_{h,p}) := \int_{\partial K} \psi_{h,p'}^+ \cdot \mathcal{H}(U_{h,p}^+, U_{h,p}^-, n) dS - \int_K F(U_{h,p}) : \nabla \psi_{h,p'} dV$$

- Minimize error function to align mesh to discontinuities: $f_{\text{err}}(\mathbf{u}, \mathbf{x}) = \mathbf{R}(\mathbf{u}, \mathbf{x})^T \mathbf{R}(\mathbf{u}, \mathbf{x})/2$
- High-order **global** convergence rates with shocks

[1] Zahr, Persson, 2018.

[2] Shi, Persson, Zahr, 2020, 2022.

[3] Zahr et al, multiple



The Necessity of New Schemes and Implicit Solvers

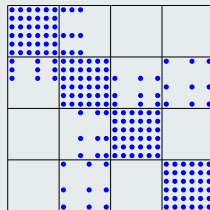
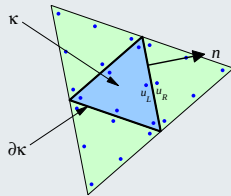
The Case for Implicit Methods

Explicit methods scale well for model problems but fail in the real world.

Robust implicit solvers are required due to:

- **Severe CFL Constraints:** Scales badly with degree ($\Delta t \lesssim Ch/p^2$).
- **Geometric Stiffness:** Low-quality/curved/boundary layer elements
- **Physical Stiffness:** Multiscale/multiphysics couplings, low-Mach flows
- **Steady States:** Non-transient problems (e.g., RANS flows).

This challenge defines the “**Solver Bottleneck**” and motivates the new discretization schemes and iterative methods discussed in this talk.



Outline

- 1 Introduction and Motivation
- 2 DG Jacobian Structures and Sparse Variants**
- 3 The Current Standard: ILU-Smoothed Multigrid Solvers
- 4 Recent Results in Matrix-Free, Elimination-Based, and Multigrid Solvers
 - Kronecker-SVD Preconditioners
 - Static Condensation for DG
 - Agglomerated Multigrid
 - Geometric Adaptive Smoothed Aggregation Multigrid

The Discontinuous Galerkin Method

General systems of conservation laws

- (Reed/Hill 1973, Lesaint/Raviart 1974, Cockburn/Shu 1989-)
- Consider first-order system of conservation laws

$$\mathbf{u}_t + \nabla \cdot \mathbf{F}(\mathbf{u}) = 0$$

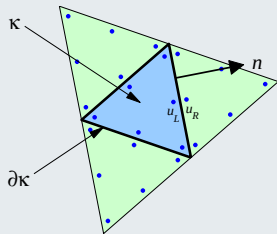
- Triangulate domain Ω into elements $\kappa \in \mathcal{T}_h$
- Seek approximate solution $\mathbf{u}_h$ in the space of element-wise

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

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

$$\int_{\kappa} (\mathbf{u}_h)_t \mathbf{v}_h \, d\mathbf{x} - \int_{\kappa} \mathbf{F}(\mathbf{u}_h) \cdot \nabla \mathbf{v}_h \, d\mathbf{x} + \int_{\partial\kappa} \hat{\mathbf{F}}(\mathbf{u}_h^+, \mathbf{u}_h^-, \hat{\mathbf{n}}) \mathbf{v}_h^+ \, ds = 0$$

- Numerical flux function $\hat{\mathbf{F}}(\mathbf{u}_R, \mathbf{u}_L, \hat{\mathbf{n}})$ for left/right states $\mathbf{u}_L, \mathbf{u}_R$ in direction $\hat{\mathbf{n}}$
- Global problem: Find $\mathbf{u}_h \in \mathbf{V}_h^p$ such that this weighted residual is zero for all $\mathbf{v}_h \in \mathbf{V}_h^p$
- Errors often $\mathcal{O}(h^{p+1})$ for smooth solutions



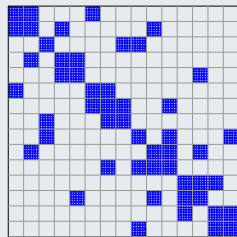
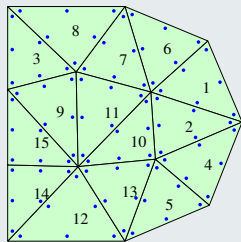
Jacobian Matrix Sparsity Structure

DG Block Structured Matrices

- Dual mesh, each entry a large complete graph A_{ij}
- Assume all blocks dense for simplicity (more later)
- Block-based storage format essential for high BLAS performance
- Blocks often large: $\mathcal{O}(p^D)$ nodes / element
- Example: 3-D Navier-Stokes, $p = 3$, $T = 100,000$, tetrahedra:

Quantity	Size	Example storage
Solution vector	SCT	80 MB
Full block Jacobian	$(D + 2)S^2C^2T$	40 GB

- Many researchers have proposed **matrix-free** DG methods, however, Jacobian typically required for **preconditioning**

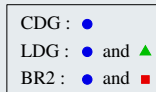
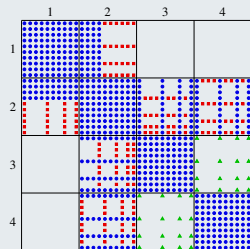


Jacobian Matrix Sparsity Structure

Improving the sparsity – the Compact DG Method

- In the general setting, all nodes within an element are connected
- But with nodal bases, 1st order operators only communicate on faces
- For 2nd order operators, our Compact DG (CDG) scheme [1] is currently the sparsest known scheme

Quantity	Size	Example storage
Solution vector	SCT	80 MB
Inviscid Jacobian	$(S^2 + (D + 1)s^2)C^2T$	16 GB
CDG Jacobian	$(S^2 + (D + 1)Ss)C^2T$	24 GB
BR2/IP Jacobian	$(S^2 + (D + 1)(S + s)s)C^2T$	32 GB
Full block Jacobian	$(D + 2)S^2C^2T$	40 GB



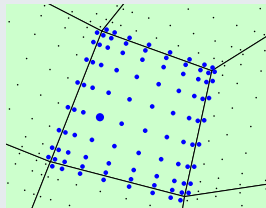
Jacobian Matrix Sparsity Structure

Line-Based Sparsity

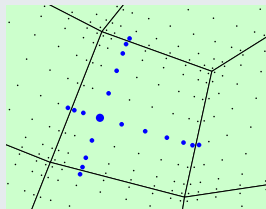
- For **outer product elements** (quads, hexes) and **1st order operators**, the stencil size can be reduced from $\mathcal{O}(p^D)$ to $\mathcal{O}(Dp)$
- For example, the **DG-SEM method** with Gauss-Lobatto nodes. But it requires additional stabilization due to under-integration.
- The **Line-DG method** [1,2] obtains the same sparsity pattern with consistent integration along lines

connectivities / node (block elements)

Polynomial order p		1	2	3	4	5	6	7	8	9	10
2-D	Line-DG connectivities	7	9	11	13	15	17	19	21	23	25
	Nodal-DG connectivities	8	13	20	29	40	53	68	85	104	125
3-D	Line-DG connectivities	10	13	16	19	22	25	28	31	34	37
	Nodal-DG connectivities	20	45	88	155	252	385	560	783	1060	1397



Nodal-DG



Line-DG

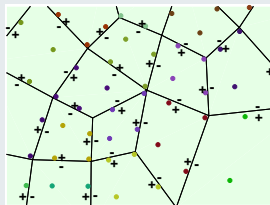
[1] Persson, *J. Comp. Phys.*, 2013.

[2] Pazner, Persson, *J. Sci. Comp.*, 2019.

Jacobian Matrix Sparsity Structure

Other DG-SEM Nodes

- DG-SEM schemes with improved integration accuracy give similar sparsity pattern
- Gauss-Legendre nodes gives 2 additional orders, but significantly larger stencil
- The “perfect compromise”: Gauss-Radau (half-closed DG [1]) gives:
 - One additional integration order
 - Correct mass matrices
 - Linear stability for constant coefficient PDEs
 - Local sparsity pattern that e.g. allows for elimination



Half-Closed DG

[1] Pan, Persson, *J. Comp. Phys.*, 2025.

Summary of Line-Based DG Methods

- **Cheap storage of full Jacobian matrices**, but requires quads/hexes and specialized solvers for 2nd order operators
- However, also requires a **sparsity preserving preconditioner**, which is difficult to construct

Outline

- 1 Introduction and Motivation
- 2 DG Jacobian Structures and Sparse Variants
- 3 The Current Standard: ILU-Smoothed Multigrid Solvers**
- 4 Recent Results in Matrix-Free, Elimination-Based, and Multigrid Solvers
 - Kronecker-SVD Preconditioners
 - Static Condensation for DG
 - Agglomerated Multigrid
 - Geometric Adaptive Smoothed Aggregation Multigrid

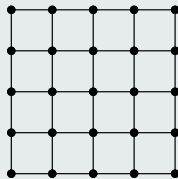
Direct Methods

Scaling of Optimal Direct Sparse Solvers

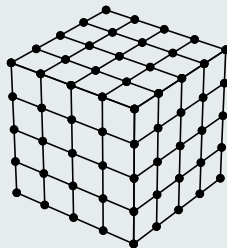
Time and space to solve **any problem** on **any well-shaped mesh** with N nodes:



1-D



2-D



3-D

Space (fill):

$\mathcal{O}(N)$

$\mathcal{O}(N \log N)$

$\mathcal{O}(N^{4/3})$

Time (flops):

$\mathcal{O}(N)$

$\mathcal{O}(N^{3/2})$

$\mathcal{O}(N^2)$

- Not competitive in 3D due to severe fill-in, and difficult to parallelize
- However, can be used for **static condensation** and **incomplete factorizations**

Iterative Methods

Krylov Subspace Methods

Instead, consider **iterative methods**, in particular, **Krylov subspace methods**:

- Conjugate Gradient (CG): For SPD problems
- Generalized Minimum-RESidual (GMRES): For anything else

Iterative methods are:

- In general **cheap, low memory, parallelizable**
- However, typically requires excessive number of iterations for stiff problems
- Sophisticated problem-specific **preconditioning** required

Preconditioners for DG

Standard Preconditioners for DG

- **Block diagonal:** $\mathbf{A} \approx \text{blockdiag}(\mathbf{A})$.
 - Simple and cheap, but in general poor performance.
- **Block Incomplete LU:** $\mathbf{A} \approx \tilde{\mathbf{L}}\tilde{\mathbf{U}}$, Gaussian elimination ignoring fill.
 - Simple and cheap, but difficult to parallelize. Exact for convection with good element ordering.
- **Multigrid (h/p):** Apply inexpensive “smoother”, transfer residual/error between spaces (varying degree p or “mesh” h), apply recursively
 - Needs mesh hierarchy for h -variant, parallelizes well. Very fast convergence for diffusion.
- **Highly efficient combination [1]:**
 - Use block ILU(0) as post-smoother for multigrid
 - Combines advantages of ILU and coarse-scale preconditioners
 - Inexpensive, general purpose, good for wide range of problems

[1] Persson, Peraire, *SIAM J. Sci. Comp.*, 2008.

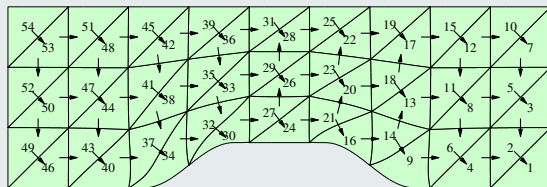
Element Order Dependency

The Minimum Discarded Fill (MDF) Element Ordering

- Properties of Incomplete LU preconditioner highly dependent on element ordering
- Greedy algorithm for pseudo-optimal element ordering [1]: At step j , if j' is chosen next, we would discard the fill

$$\Delta \tilde{U}_{ik}^{(j,j')} = -\tilde{U}_{ij'} \tilde{U}_{j'j'}^{-1} \tilde{U}_{j'k}, \quad \text{for neighbors } i \geq j, k \geq j \text{ of element } j'$$

- Choose the j' that minimizes the norm of the discarded fill: $w^{(j,j')} = \|\Delta \tilde{U}^{(j,j')}\|_F$
- Some simplifications + min-heap data structure $\implies \mathcal{O}(n \log n)$ cost

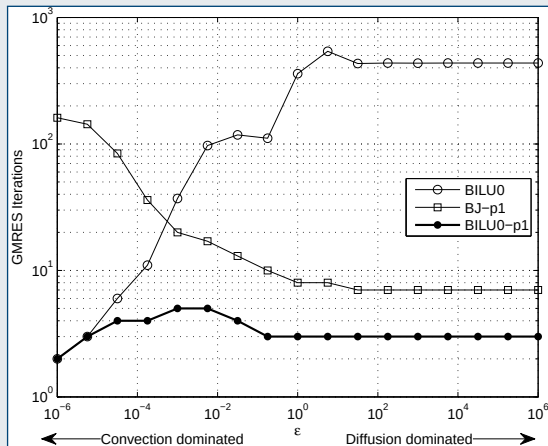


[1] Persson, Peraire, *SIAM J. Sci. Comp.*, 2008.

The ILU-p1 Preconditioner - Convection-Diffusion

GMRES Iterations for Convection-Diffusion Problem

- Block ILU perfect for convection, multigrid perfect for diffusion
- Block ILU-smoothed multigrid (BILU0-p1) almost perfect for any ε



Convergence – Model Navier-Stokes Problem (1/2)

Iteration Counts

× = No convergence after 1,000 iterations.

			Block Jacobi		Block ILU0	
	Δt	M	Pure	MG	Pure	MG
Inviscid	10^{-3}	0.2	24	19	5	4
	10^{-1}	0.2	187	85	12	6
	∞	0.2	840	142	40	9
	10^{-3}	0.01	200	112	15	6
	10^{-1}	0.01	×	×	94	10
	∞	0.01	×	×	374	16
Laminar Re=1,000	10^{-3}	0.2	50	34	4	4
	10^{-1}	0.2	×	597	11	5
	∞	0.2	×	×	37	7
	10^{-3}	0.01	98	66	8	5
	10^{-1}	0.01	×	619	27	9
	∞	0.01	×	×	135	12

Convergence – Model Navier-Stokes Problem (2/2)

Iteration Counts

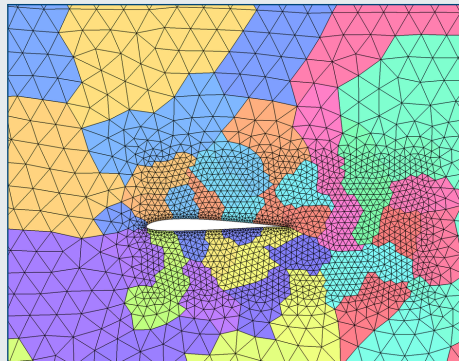
× = No convergence after 1,000 iterations.

			Block Jacobi		Block ILU0	
	Δt	M	Pure	MG	Pure	MG
Laminar Re=20,000	10^{-3}	0.2	26	19	4	4
	10^{-1}	0.2	456	220	16	8
	∞	0.2	×	×	236	20
	10^{-3}	0.01	160	90	12	6
	10^{-1}	0.01	×	×	80	16
	∞	0.01	×	×	×	35
RANS Re= 10^6	10^{-3}	0.2	76	56	8	7
	10^{-1}	0.2	×	×	35	25
	∞	0.2	×	×	70	18
	10^{-3}	0.01	411	231	14	9
	10^{-1}	0.01	×	×	46	16
	∞	0.01	×	×	132	28

ILU Parallelization – Domain Decomposition

Parallel Strategy

- Parallelization of true ILU difficult
- Instead, use partition-wise ILUs with MDF ordering
- Partition using the weights $C_{ij} = \|A_{ii}^{-1}A_{ij}\|_F$
- Essentially a “non-overlapping Schwarz preconditioner with incomplete solutions”
- Approaches Jacobi as $\#$ partitions $\rightarrow \#$ elements



[1] Persson, *AIAA SciTech*, 2009.

Block-ILU and h/p -Multigrid Solvers – Summary

Core Benefits

- **Effective Hybrid:** Combines Block-ILU for convection with multigrid for diffusion.
- **General Purpose:** Simple to construct, excellent convergence across many flow regimes.
- **Optimal Ordering:** Algorithms like MDF maximize iterative solver performance.

Limitations & Future Work

- **Matrix Dependency:** Fully matrix-free implementations inherently difficult.
- **Parallel Scaling:** Domain decomposition degrades performance in strong scaling regime.
- **Hierarchy Required:** h -multigrid strictly relies on a nested mesh hierarchy.
- **Motivation:** These bottlenecks have inspired our continued research on improved solvers.

Outline

- 1 Introduction and Motivation
- 2 DG Jacobian Structures and Sparse Variants
- 3 The Current Standard: ILU-Smoothed Multigrid Solvers
- 4 Recent Results in Matrix-Free, Elimination-Based, and Multigrid Solvers
 - Kronecker-SVD Preconditioners
 - Static Condensation for DG
 - Agglomerated Multigrid
 - Geometric Adaptive Smoothed Aggregation Multigrid

Outline

- 1 Introduction and Motivation
- 2 DG Jacobian Structures and Sparse Variants
- 3 The Current Standard: ILU-Smoothed Multigrid Solvers
- 4 Recent Results in Matrix-Free, Elimination-Based, and Multigrid Solvers
 - Kronecker-SVD Preconditioners
 - Static Condensation for DG
 - Agglomerated Multigrid
 - Geometric Adaptive Smoothed Aggregation Multigrid

Tensor-product preconditioners

Three main goals

- ① **Efficient** to form: $\mathcal{O}(p^{d+1})$
- ② **Efficient** to solve: $\mathcal{O}(p^{d+1})$
- ③ **Provably good** approximations to known preconditioners

Main idea

Form **optimal** approximations to block Jacobi preconditioner with tensor-product structure:

$$\text{2D: } P = A_1 \otimes B_1 + A_2 \otimes B_2$$

$$\text{3D: } P = A_1 \otimes B_1 \otimes C_1 + A_1 \otimes B_2 \otimes C_2$$

Three key features

- ① Efficient “*KSVD*” and Lanczos iterations
- ② Fast inverses via Schur form
- ③ **Optimal** in $\|\cdot\|_F$ and **exact** for certain classes of equations

Nearest Kronecker problem

Concept Definition

- Finding A_1, B_1, A_2, B_2 such that

$$A \approx A_1 \otimes B_1 + A_2 \otimes B_2$$

is known as the **nearest Kronecker problem (NKP)**

- Optimal solution (in Frobenius norm) is given by Kronecker-product singular value decomposition (KSVD)
- Low **tensor-rank** approximation of A is equivalent to low-rank approximation of $\tilde{A}$, where

$$\tilde{A} = \begin{pmatrix} \tilde{A}_1 \\ \tilde{A}_2 \\ \vdots \\ \tilde{A}_{n_1} \end{pmatrix}, \text{ and } \tilde{A}_j \text{ is a block of rows given by } \tilde{A}_j = \begin{pmatrix} (\vec{A}_{1j})^T \\ (\vec{A}_{2j})^T \\ \vdots \\ (\vec{A}_{m_1,j})^T \end{pmatrix}$$

Fast inverses via Schur factorization

Method Overview

- Similar to “matrix diagonalization” (spectral methods)
- Transform the system

$$(A_1 \otimes B_1 + A_2 \otimes B_2)x = b$$

$$(A_2^{-1}A_1 \otimes I + I \otimes B_1^{-1}B_2)x = (A_2^{-1} \otimes B_1^{-1})b$$

$$(Q_1 \otimes Q_2)(T_1 \otimes I + I \otimes T_2)(Q_1^T \otimes Q_2^T)x = (A_2^{-1} \otimes B_1^{-1})b$$

using Schur factorizations

$$A_2^{-1}A_1 = Q_1T_1Q_1^T, \quad B_1^{-1}B_2 = Q_2T_2Q_2^T$$

- (Similar approach in 3D)

Exact representations

Constant coefficient convection diffusion equation

- Important special case for which the KSVD approximation is exact:

- Convection-Diffusion

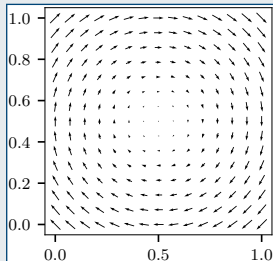
$$u_t + \nabla \cdot (\beta u) - \epsilon \Delta u = 0$$

- Cartesian grids
 - Constant-coefficient diffusion and velocity field
- Tensor representation, with diagonal Gauss weight matrix W , Gauss point evaluation matrix G , differentiation matrix D , and end-point evaluation matrices G_0, G_1 and D_0, D_1

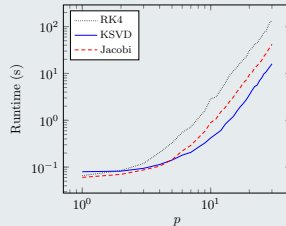
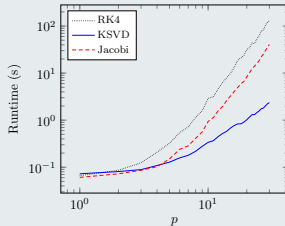
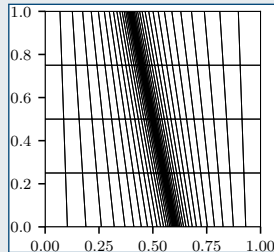
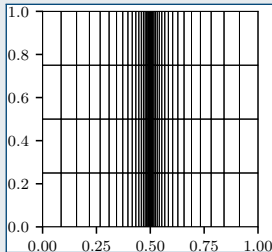
$$\begin{aligned} & G^T W G \otimes (-D^T W D - \eta_e G_0^T G_0 - \eta_e G_1^T G_1 - D_0^T G_0 - G_0^T D_0 + D_1^T G_1 + G_1^T D_1) \\ & + (-D^T W D - \eta_e G_0^T G_0 - \eta_e G_1^T G_1 - D_0^T G_0 - G_0^T D_0 + D_1^T G_1 + G_1^T D_1) \otimes G^T W G \end{aligned}$$

Results: scalar advection equation

Runtime Performance, Anisotropic Skewed Meshes

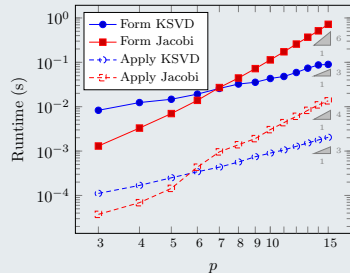


Velocity Field



Results: Kelvin-Helmholtz

Timing

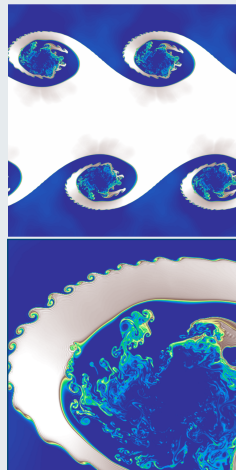


Iterations

GMRES iterations, $p = 3$

Δt	J	K
2.5×10^{-5}	4	4
5.0×10^{-5}	5	5
1.0×10^{-4}	6	7
2.5×10^{-4}	8	10
5.0×10^{-4}	10	14
1.0×10^{-3}	13	22

Flow Visualizations



Kronecker-SVD Preconditioners – Summary

Core Benefits

- **Low-Memory Implicit DG:** Enables matrix-free Kronecker product approximations for elemental operators.
- **Fast Local Inverses:** Uses Schur factorization for highly efficient direct local solves.
- **Asymptotically Optimal:** Faster and requires less storage than standard dense block inversions.

Limitations & Future Work

- **Geometric Restrictions:** Currently limited to outer-product elements like quads and hexes.
- **Setup Costs:** High overhead in KSVD construction phase; exploring cheaper alternatives.
- **Complex Systems:** Needs specialized treatment for highly non-linear equations like Euler.

[1] Pazner, Persson, *SIAM J. Comp. Phys.*, 2018.

[2] Pazner, Persson, *AIAA SciTech*, 2018.

[3] Pazner, Persson, *AIAA CFD Conf.*, 2018. (*Winner, AIAA CFD 2017 Student Paper Competition*)

Outline

- 1 Introduction and Motivation
- 2 DG Jacobian Structures and Sparse Variants
- 3 The Current Standard: ILU-Smoothed Multigrid Solvers
- 4 Recent Results in Matrix-Free, Elimination-Based, and Multigrid Solvers
 - Kronecker-SVD Preconditioners
 - Static Condensation for DG
 - Agglomerated Multigrid
 - Geometric Adaptive Smoothed Aggregation Multigrid

Static Condensation using Block Linear Algebra

Method Overview

- Static condensation widely used for CG, but somewhat unclear how to apply for DG.
- Linear system $Ax = b$, split unknowns in independent/dependent DOFs:

$$\begin{pmatrix} A_{ii} & A_{id} \\ A_{di} & A_{dd} \end{pmatrix} \begin{pmatrix} x_i \\ x_d \end{pmatrix} = \begin{pmatrix} f_i \\ f_d \end{pmatrix}$$

- Eliminate dependent DOFs via Schur complement:

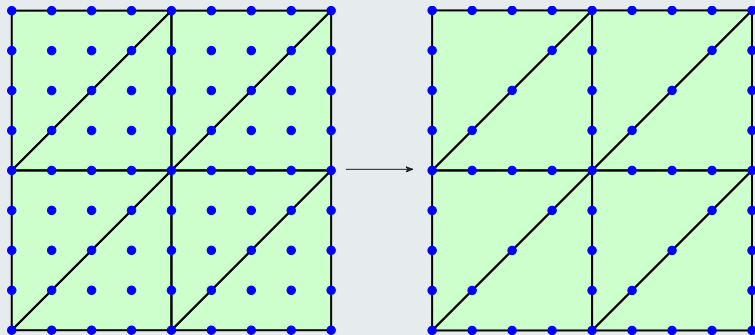
$$\begin{aligned} A_{ii}x_i + A_{id}x_d &= f_i, & A_{di}x_i + A_{dd}x_d &= f_d, & x_d &= A_{dd}^{-1}f_d - A_{dd}^{-1}A_{di}x_i \\ \implies (A_{ii} - A_{id}A_{dd}^{-1}A_{di})x_i &= f_i - A_{id}A_{dd}^{-1}f_d \end{aligned}$$

- **Schur complement steps:** Solve smaller system for independent DOFs ($Sx_i = (A_{ii} - A_{id}A_{dd}^{-1}A_{di})x_i = f_i - A_{id}A_{dd}^{-1}f_d$) and then solve for dependent DOFs ($x_d = A_{dd}^{-1}f_d - A_{dd}^{-1}A_{di}x_i$)
- If A_{dd}^{-1} sparse then S also sparse (in particular if A_{dd} is block-diagonal)
- Pick independent and dependent nodes such that A_{dd} is block-diagonal

Static Condensation for Continuous Galerkin

CG Elimination

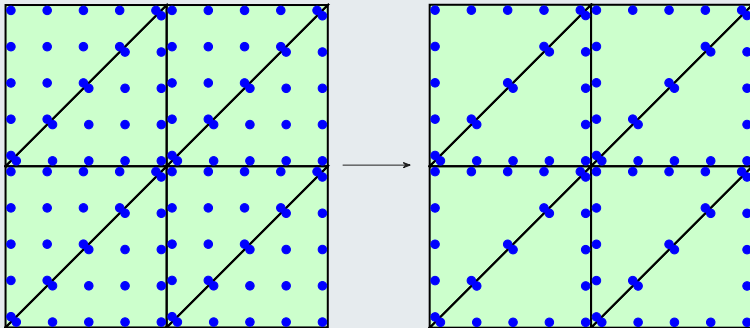
- *Static Condensation*: Eliminate degrees of freedom whose neighbors are already connected $\implies$ no additional fill. Standard procedure for CG.
- Comes automatically with minimum degree ordering! [George, Liu, 1989]
- Can be continued by merging elements $\implies$ essentially nested dissection



Static Condensation for Discontinuous Galerkin (1/2)

Interior Node Elimination

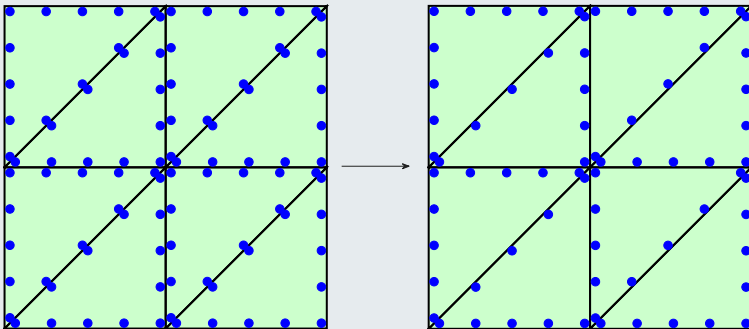
- Static Condensation clearly applicable to all *interior* DG nodes
- DOF reduction (for hexes) $(p-1)^3/(p+1)^3$ [Rueda-Ramírez et al, 2019]
- Not immediately clear how to eliminate the duplicated edge nodes



Static Condensation for Discontinuous Galerkin (2/2)

Edge Node Elimination

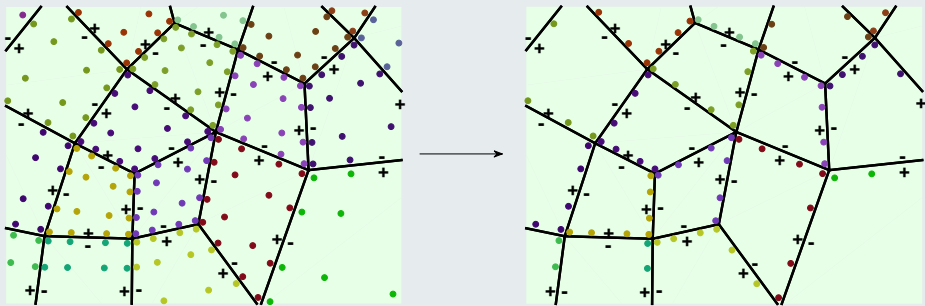
- However, by allowing the condensation to eliminate nodes that propagate their connections to other nodes, we can eliminate all duplicated edge nodes
- This is essentially the DOFs and sparsity of the Hybridizable DG (HDG) method!
- Our work goes further and also eliminates some of the corner nodes



Switch-based Elimination for Discontinuous Galerkin (1/2)

Half-Closed DG Scheme

- For *Half-Closed DG Scheme* [Pan, Persson, 2025], we showed that by introducing a switch function, only edges with e.g. “+” switches need nodes
- All other nodes can be eliminated, a DOF reduction (for hexes) of $p^3/(p+1)^3$
- Same applies for standard DG, and here we consider closed nodes for simplicity



Switch-based Elimination for Discontinuous Galerkin (2/2)

Savings and Application

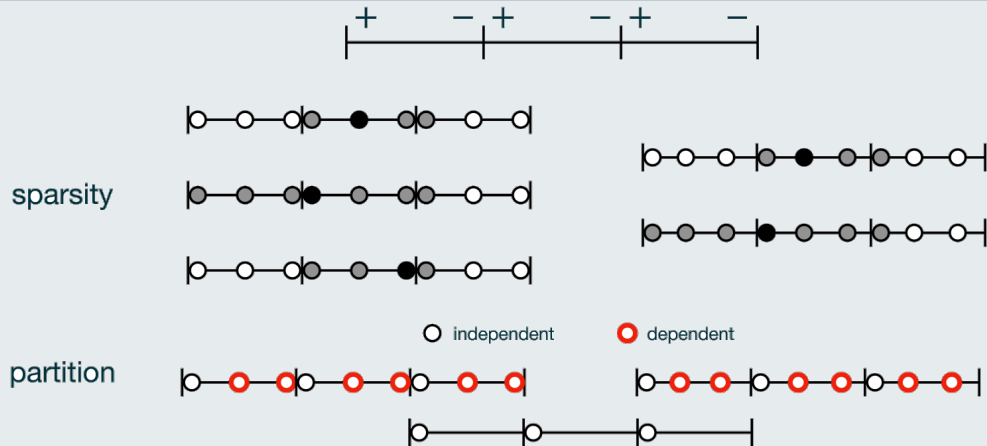
- Significant savings compared to only eliminating the interior DOFs
- Ratio of elements eliminated for hexes using only interior DOFs (top) and our switch-based pattern (bottom):

p	1	2	3	4	5	6	7
$(p-1)^3/(p+1)^3$	0%	4%	12%	22%	30%	36%	42%
$p^3/(p+1)^3$	12%	30%	42%	51%	58%	63%	67%

- Savings in Jacobian entries even higher (more later)
- First, let us show why all nodes not on “+” edges can be eliminated
- Procedure works for fully general 1st order terms + for 2nd order terms using Local DG (LDG) and Compact DG (CDG) discretizations

Static Condensation for DG

LDG Method, Sparsity Pattern and Elimination



- Eliminate all nodes not on a $+$ boundary

Static Condensation for DG

Elimination on Quad Mesh

- Quad elements, polynomial degree $p = 3$
- Eliminate

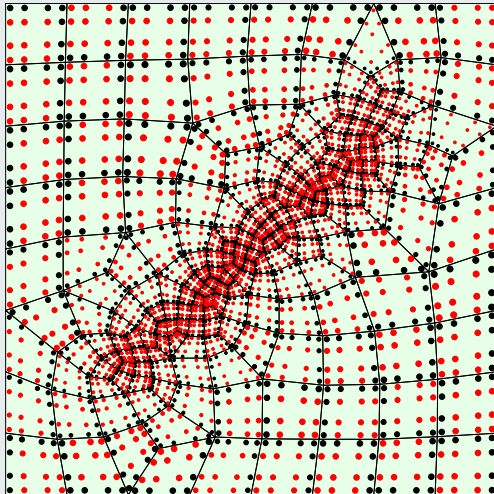
$$p^D = 3^2 = 9$$

out of

$$(p + 1)^D = 4^2 = 16$$

nodes

- Results in a “skeleton” mesh
(similar to HDG)



Static Condensation for DG

Elimination on Quad Mesh

- Quad elements, polynomial degree $p = 3$
- Eliminate

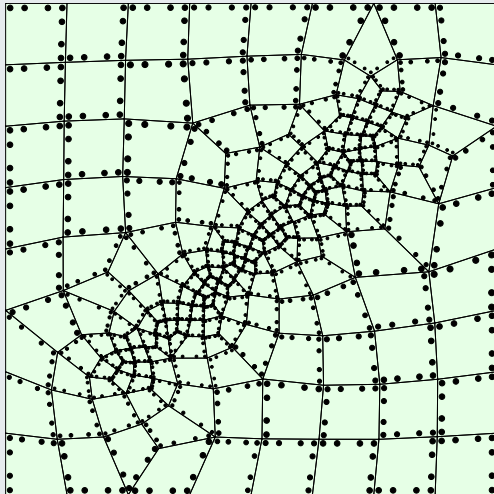
$$p^D = 3^2 = 9$$

out of

$$(p + 1)^D = 4^2 = 16$$

nodes

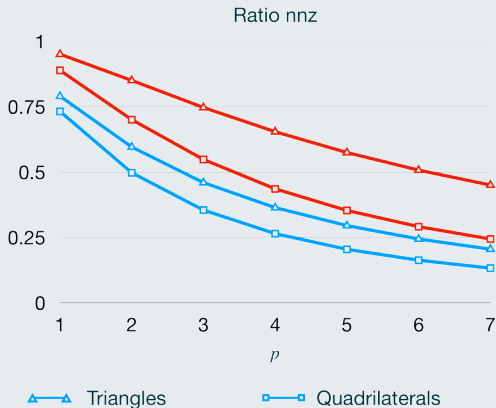
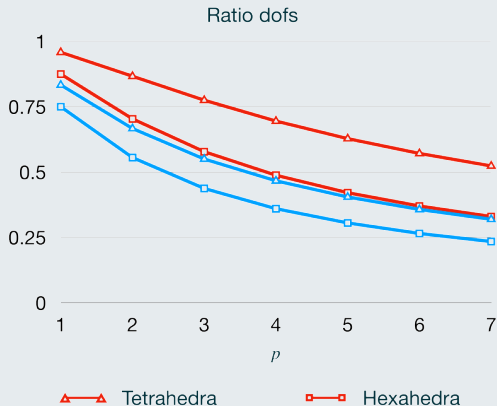
- Results in a “skeleton” mesh
(similar to HDG)



Static Condensation for DG

Degrees of Freedom and Jacobian Entries

- Comparison of degrees of freedom and number of non-zeros in Laplace operator pre/post static condensation



Static Condensation and Iterative Methods

Performance Trade-offs

- The main goal of elimination is to reduce the system size before solving iteratively
- Note that sometimes it might not be competitive for **matrix-free (or explicit) methods** or **very well-conditioned problems** (where elimination is more expensive than mat-vec)
- However, for matrix-based difficult problems, it is likely always going to improve performance in two ways:
 - Smaller systems $\implies$ lower mat-vec cost
 - Elimination improves the spectrum of the operator $\implies$ fewer iterations

Static Condensation + Block-Based Solvers

Smoothing Properties

- Want spectrum $\lambda(I - P^{-1}A)$ close to 0:

original

$$\begin{pmatrix} A_{ii} & A_{id} \\ A_{di} & A_{dd} \end{pmatrix} \begin{pmatrix} x_i \\ x_d \end{pmatrix} = \begin{pmatrix} f_i \\ f_d \end{pmatrix}$$



Iterate on A

$$\begin{pmatrix} I - P_{ii}^{-1}A_{ii} & P_{ii}^{-1}A_{id} \\ P_{dd}^{-1}A_{di} & I - P_{dd}^{-1}A_{dd} \end{pmatrix} \begin{pmatrix} x_i^{n+1} \\ x_d^{n+1} \end{pmatrix} = \begin{pmatrix} x_i^n \\ x_d^n \end{pmatrix}$$

eliminated

$$\begin{pmatrix} A_{ii} & A_{id} \\ A_{di} & A_{dd} \end{pmatrix} \begin{pmatrix} x_i \\ x_d \end{pmatrix} = \begin{pmatrix} f_i \\ f_d \end{pmatrix}$$



1. Eliminate $A \rightarrow S$
2. Iterate on S

$$\begin{pmatrix} I - P_S^{-1}S & 0 \\ C & 0 \end{pmatrix} \begin{pmatrix} x_i^{n+1} \\ x_d^{n+1} \end{pmatrix} = \begin{pmatrix} x_i^n \\ x_d^n \end{pmatrix}$$

→ Improved smoothing properties on eliminated system

Static Condensation + Block-Based Solvers

Convergence, Poisson's Equations

- The improved spectrum after elimination leads to almost 50% fewer iterations
- p -multigrid iterations for Poisson's equation
 - Number of iterations to reach 10^{-8} error
 - $\#$ iterations on eliminated $\approx 0.5 \times \#$ iterations on original

	p=2	p=3	p=5	p=8
Eliminated	8	9	9	8
Original	13	16	16	15

Elimination for DG – Summary

Core Benefits

- **Major Size Reduction:** Shrinks DOFs and Jacobian sizes by a factor of $p^D / (p + 1)^D$.
- **Enhanced Convergence:** Improves spectral properties, accelerating iterative solvers.
- **Simple and General:** Straightforward to apply to any matrix-based DG discretization.

Limitations & Future Work

- **Constraints:** Requires specific viscous formulations (LDG/CDG), loses the benefits of line-based sparsity patterns
- **Performance Overhead:** Elimination has to cost less than the saved mat-vec operations.
- **Future Plans:** Developing optimized parallel version to assess the cost-benefit.

[1] Y. Pan and P.-O. Persson, Half-closed Discontinuous Galerkin discretisations. J. Comp. Phys., Vol. 524, 113731, 2025.

[2] A. Nayak, Y. Pan and P.-O. Persson, Eliminated p-Multigrid for Half-Closed Discontinuous Galerkin Methods. In preparation.

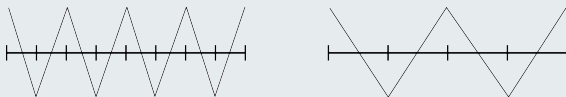
Outline

- 1 Introduction and Motivation
- 2 DG Jacobian Structures and Sparse Variants
- 3 The Current Standard: ILU-Smoothed Multigrid Solvers
- 4 Recent Results in Matrix-Free, Elimination-Based, and Multigrid Solvers
 - Kronecker-SVD Preconditioners
 - Static Condensation for DG
 - Agglomerated Multigrid
 - Geometric Adaptive Smoothed Aggregation Multigrid

Multigrid (MG) methods

Concept

- Originally designed for elliptic equations on structured domains.
- Main idea:
 - Recursively solve linear algebra problem on nested hierarchy.
 - Successive levels resolve distinct high frequencies.
 - Achieve asymptotically optimal linear $\mathcal{O}(N)$ runtime.



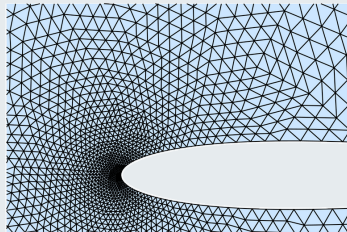
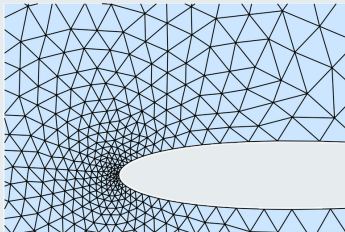
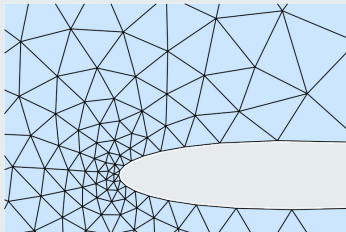
Ingredients in general MG method

- Restriction R_k^{k+1} : Transfer of residual from higher to lower level.
- Smoother S_k : Relax high frequency modes at level k , e.g. Jacobi, Gauss-Seidel, ILU
- Interpolation T_{k+1}^k : Transfer of solution from lower to higher level.

Geometric Multigrid (GMG)

GMG Properties

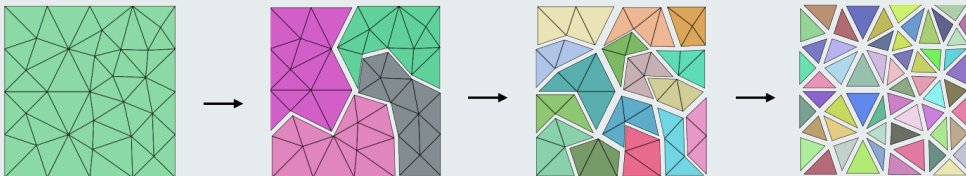
- Hierarchy is formed using mesh discretisation, generally nested mesh constructions are used.
- Direct polynomial injection generally used for interpolation operator, restriction taken to be its adjoint: $R_k^{k+1} = (T_{k+1}^k)^T$.
- Easy to form mesh hierarchy using refinement, but in practice only a fine unstructured mesh is given which is difficult to coarsen



Agglomerated Geometric Multigrid

Agglomeration

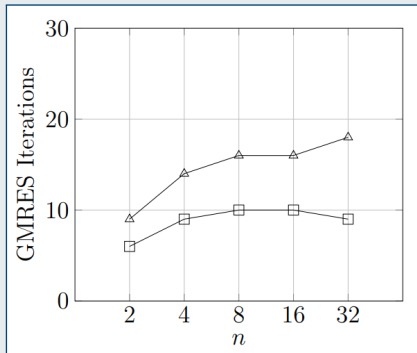
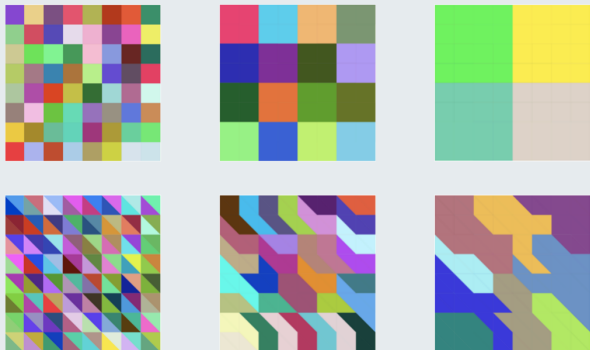
- Main idea:
 - Discontinuous Galerkin discretizations naturally allow for arbitrary (polyhedral) element shapes
 - Create multigrid mesh hierarchies using a new element agglomeration algorithm which extends to arbitrary elements and dimensions
 - Form groups using recursive application of graph partitioning (METIS)
- Result:
 - Full h/p -multigrid solver performance on unstructured meshes of arbitrary elements and approximation orders.
 - Avoids the fundamental difficulty of conforming mesh coarsening.



Agglomerated Geometric Multigrid

Convergence Results

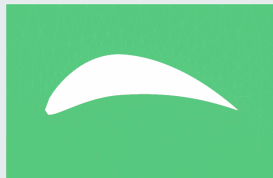
- Convergence for square mesh problem with regular and non-regular mesh hierarchy
- Multigrid is relatively insensitive to the element partitioning



Agglomerated Geometric Multigrid

Airfoil Mesh Agglomeration

- Example: Grouping by a simple local-neighbor method



Agglomerated Geometric Multigrid – Summary

Core Benefits

- **Unstructured Hierarchy:** Multigrid hierarchies on unstructured meshes via graph agglomeration.
- **Geometric Flexibility:** Extends to arbitrary polyhedral elements and approximation orders.
- **Solver Efficiency:** Full h/p -multigrid convergence without conforming mesh coarsening.

Limitations & Future Work

- **Polyhedral Elements:** Requires new data structures to manage the coarse meshes.
- **Next Steps:** Efficient implementations and matrix-free versions

Outline

- 1 Introduction and Motivation
- 2 DG Jacobian Structures and Sparse Variants
- 3 The Current Standard: ILU-Smoothed Multigrid Solvers
- 4 Recent Results in Matrix-Free, Elimination-Based, and Multigrid Solvers
 - Kronecker-SVD Preconditioners
 - Static Condensation for DG
 - Agglomerated Multigrid
 - Geometric Adaptive Smoothed Aggregation Multigrid

Algebraic multigrid (AMG)

AMG Properties

- Introduced as an alternative to geometric multigrid methods.
- Multigrid hierarchy formed using matrix-entries instead of the underlying mesh
- Two main variants: Classical Ruge-Stuben and Smoothed Aggregation AMG.
- However, reliance on the matrix means its entries must be readily accessible.

Smoothed Aggregation Overview [Vanek et al., 1996]

- Fundamentally a method of constructing interpolation/restriction operators for AMG.
- Assume given two partitions B_1, B_2 of the degrees of freedom defining the next MG level.
- Assume given vectors $V = \{v_1, \dots, v_j\}$ representing the column space of the next level.
- Form block diagonal matrix P_1 by splitting coarse modes of V according to partitions
- Define prolongation operator by $T_1 = S_0 P_1$, where smoothing is applied because of high frequency modes introduced by the partitioning

Adaptive smoothed aggregation (α SA)

Adaptive Concept [Brezina et al., 2005]

- Main idea: Adaptively find coarse modes by applying smoother on some random vectors.
- Why? The leftover modes should span the column space of the next level of the hierarchy!

α SA Algorithm Steps

- ➊ Assume given matrix A , and partitions of its degrees of freedom $B_1, \dots, B_m$.
- ➋ Define smoother S , e.g., Jacobi, $S = I - \omega D^{-1}A$, $\omega = 2/3$, $D = \text{diagonal of } A$.
- ➌ **Form random matrix with random Gaussian entries** $(b_1 \dots b_r)$.
- ➍ **Apply S to each random vector p times to obtain coarse space vectors V .**
- ➎ Partition V according to $B_1, \dots, B_m$.
- ➏ **Apply SVD to each diagonal block to filter which modes to keep.**
- ➐ Form prolongation operator as before.

Adaptive smoothed aggregation (α SA)

SVD and Partitions

$$\begin{array}{c}
 B_1 \left\{ \begin{pmatrix} v_{11} & \dots & v_{1j} \\ \vdots & & \vdots \\ v_{k1} & \dots & v_{kj} \\ v_{(k+1)1} & \dots & v_{(k+1)j} \\ \vdots & & \vdots \\ v_{n1} & \dots & v_{nj} \end{pmatrix} \right. \\
 B_2 \left\{ \begin{pmatrix} v_{11} & \dots & v_{1j} \\ \vdots & & \vdots \\ v_{k1} & \dots & v_{kj} \\ v_{(k+1)1} & \dots & v_{(k+1)j} \\ \vdots & & \vdots \\ v_{n1} & \dots & v_{nj} \end{pmatrix} \right. \\
 \end{array}
 \xrightarrow{\quad}
 \begin{array}{c}
 \begin{pmatrix} v_{11} & \dots & v_{1j} \\ \vdots & & \vdots \\ v_{k1} & \dots & v_{kj} \\ v_{(k+1)1} & \dots & v_{(k+1)j} \\ \vdots & & \vdots \\ v_{n1} & \dots & v_{nj} \end{pmatrix} \\
 \begin{pmatrix} v_{11} & \dots & v_{1j} \\ \vdots & & \vdots \\ v_{k1} & \dots & v_{kj} \\ v_{(k+1)1} & \dots & v_{(k+1)j} \\ \vdots & & \vdots \\ v_{n1} & \dots & v_{nj} \end{pmatrix}
 \end{array}
 \xrightarrow{\begin{array}{l} V[B_1, :] = U_1 \Sigma_1 V_1^T \\ V[B_2, :] = U_2 \Sigma_2 V_2^T \end{array}}
 \begin{array}{c}
 \begin{pmatrix} u_{11} & \dots & u_{1l} \\ \vdots & & \vdots \\ u_{k1} & \dots & u_{kl} \\ u_{(k+1)1} & \dots & u_{(k+1)l} \\ \vdots & & \vdots \\ u_{n1} & \dots & u_{nl} \end{pmatrix} \\
 \begin{pmatrix} u_{11} & \dots & u_{1l} \\ \vdots & & \vdots \\ u_{k1} & \dots & u_{kl} \\ u_{(k+1)1} & \dots & u_{(k+1)l} \\ \vdots & & \vdots \\ u_{n1} & \dots & u_{nl} \end{pmatrix}
 \end{array}
 \xrightarrow{\quad}
 \begin{array}{c}
 u_{(k+1)1} \quad \dots \quad u_{(k+1)l} \\
 \vdots \quad \dots \quad \vdots \\
 u_{n1} \quad \dots \quad u_{nl}
 \end{array}$$

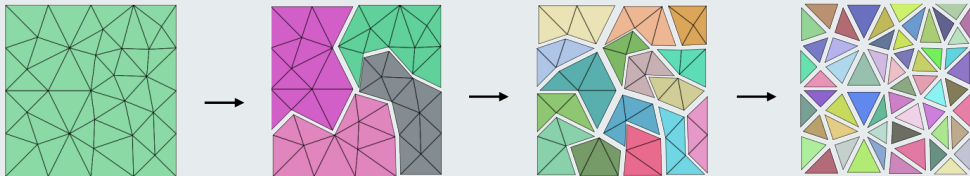
V P_1^0

- Additional SVD step to filter out relevant modes in each block.
- Difference in α SA vs SA lies in how V is formed:
 - **Pre-determined** in SA, **adaptively found via smoothing** random vectors in α SA.

Geometric α SA for DG

Mesh-Based Partitioning

- Main idea: Form mesh partitions $B_1, \dots, B_m$ using mesh information.
- To form each partition block B_i , agglomerate elements together using graph partitioning
- Advantage: Removes explicit dependence on matrix A , α SA construction allows for generalisation to unstructured meshes.
- Combine advantages of GMG and smoothed aggregation AMG.



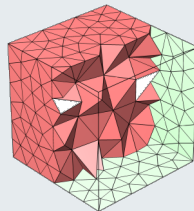
Poisson's equation (h - vs h^* - multigrid)

Setup

- Test problem: $-\Delta u = f$ in 3D, $\Omega = [-1, 1]^3$, fixed mesh, variable p .

p	h^* -	h -	h^* -(pCG)	h -(pCG)
1	8	8	7	6
2	8	9	6	7
3	10	9	7	7

#Iterations

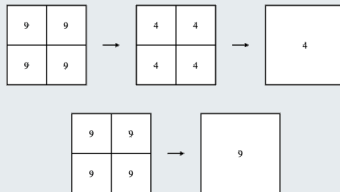


k	dof	nnz
0	110,592	20,901,888
1	56,029	5,411,417
2	9,574	5,387,940
3	3,524	640,246

h^* -MG hierarchy at $p=2$

k	dof	nnz
0	110,592	20,901,888
1	18,500	16,494,040
2	3,104	2,342,854

h -MG hierarchy at $p=2$



Convection-diffusion

Setup

- Test problem: $\mathbf{v} \cdot \nabla u - \mu \Delta u = f$ in 3D, $\Omega = [-1, 1]^3$, $p = 1$, mesh refinement.

#Iterations - Pe = 0

nRef	nDof	Our method	block Jacobi
0	4,188	9	184
1	29,180	10	430
2	214,260	11	1462
3	1,660,556	13	3838

#Iterations - Pe = 100

nRef	nDof	Our method	block Jacobi
0	4,188	9	76
1	29,180	10	181
2	214,260	12	451
3	1,660,556	16	892

#Iterations - Pe = 1000

nRef	nDof	Our method	block Jacobi
0	4,188	10	46
1	29,180	12	83
2	214,260	12	279
3	1,660,556	17	589

#Iterations - Pe = ∞

nRef	nDof	Our method	block Jacobi
0	4,188	9	37
1	29,180	13	63
2	214,260	17	118
3	1,660,556	25	416

Geometric α SA for DG – Summary

Core Benefits

- **Best of Both Worlds:** Merges geometric multigrid for DG with the robust algebraic coarsening of SA-AMG.
- **General Fluxes:** Performs well for various numerical fluxes without specialized treatments.
- **Proven Performance:** Fast convergence for a range of convection-diffusion problems.

Limitations & Future Work

- **Setup Cost:** Smoothing of the random vectors can be expensive.
- **Parameter Sensitivity:** Robustness depends on various parameters and tolerances.
- **Future Directions:** Work in progress – New improvements to algorithm, extensions to systems, etc.

Conclusions

Summary

- **The Implicit Advantage:** To handle difficult real-world problems, we need implicit methods driven by modern iterative solvers and advanced preconditioners.
- **Closing the Gap:** These algorithmic advances can help bring high-order DG into mainstream industrial applications.

Many thanks to:

- The GAMM Organizers
- My Collaborators
- Supporting agencies:
 - DOE ASCR Applied Mathematics
 - NSF Computational Mathematics
 - AFOSR Computational Mathematics
 - NASA NRA

