High-Order Methods for Turbulent Flow Simulations on Deforming Domains

Per-Olof Persson

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

Joint work with B. Froehle, L. Wang, S. Kanner, M. Zahr, D. J. Willis, J. Peraire

Séminaire du Laboratoire Jacques-Louis Lions
Université Pierre et Marie Curie (Paris VI)

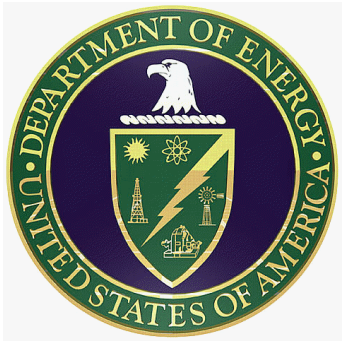
University of California Berkeley

November 13, 2015

Berkeley Lab

Outline

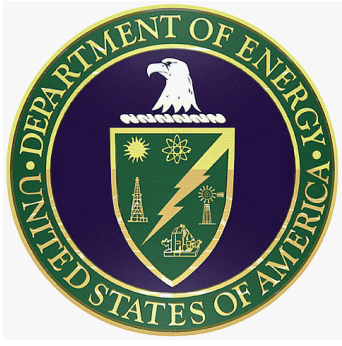
1. Introduction and Motivation
2. Numerical Schemes – Discretization and Solvers
   - The Discontinuous Galerkin Method
   - The Compact Discontinuous Galerkin (CDG) Method
   - Preconditioning for Newton-Krylov Solvers
   - Reducing the cost: The Line-DG Method
3. Methods for Deforming Domains
   - High-Order ALE Formulation
   - Unstructured Mesh Space-Time Methods



Outline

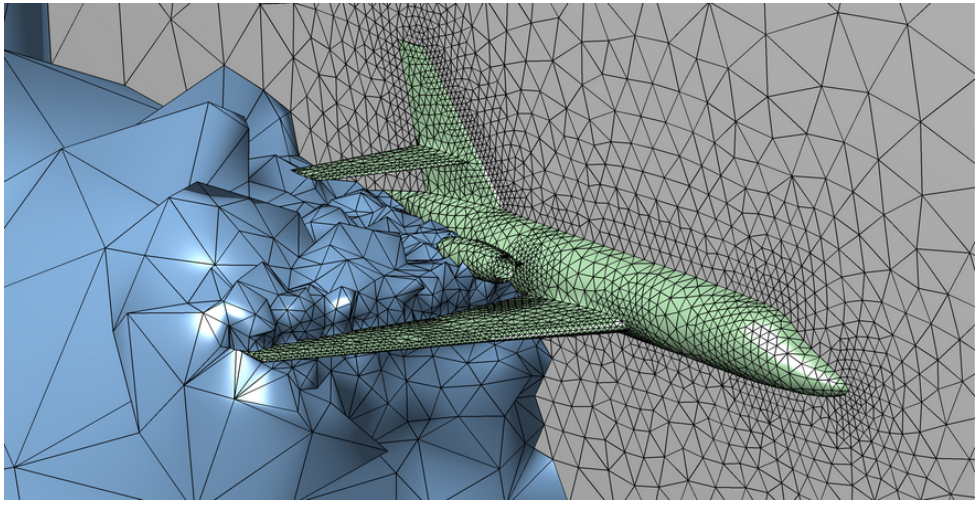
Motivation

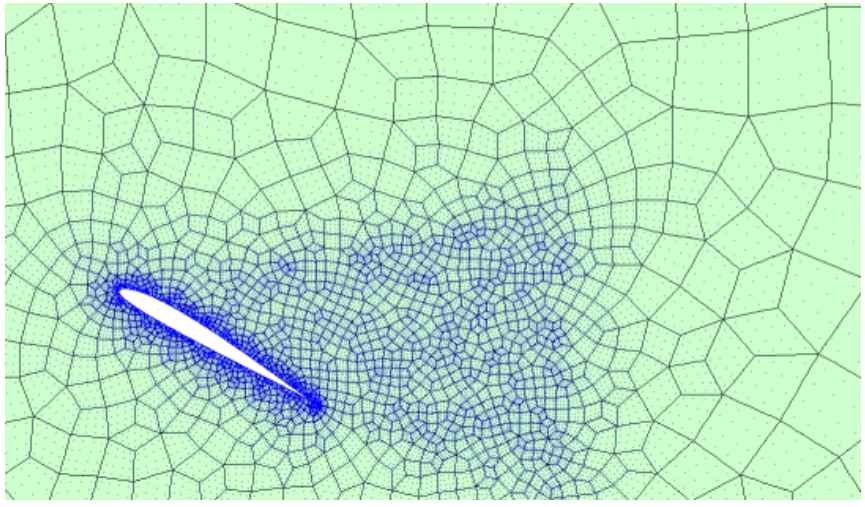
- Need for higher fidelity predictions in computational mechanics
  - Turbulent flows, wave propagation, multiscale phenomena, non-linear interactions
- Many practical applications involve time-varying geometries
  - Fluid/structure interaction, flapping flight, wind turbines, rotor-stator flows
- Goal: Develop *robust*, *efficient*, and *accurate* high-order methods based on fully unstructured meshes



Why Unstructured Meshes?

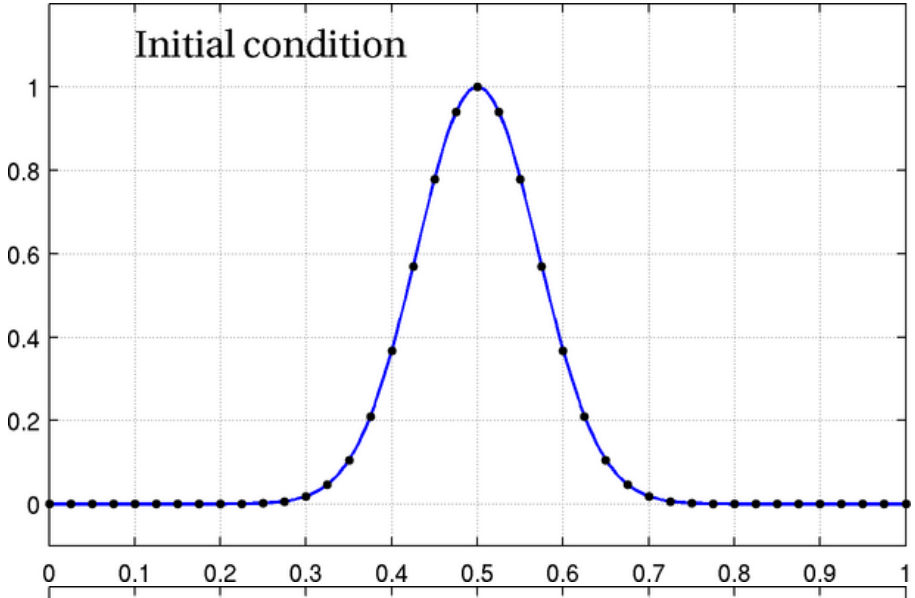
- Complex *geometries* need flexible element topologies
- Complex *solution fields* need spatially variable resolution
- Fully automated mesh generators for CAD geometries are based on unstructured simplex elements
- Real-world simulation software dominated by unstructured mesh discretization schemes

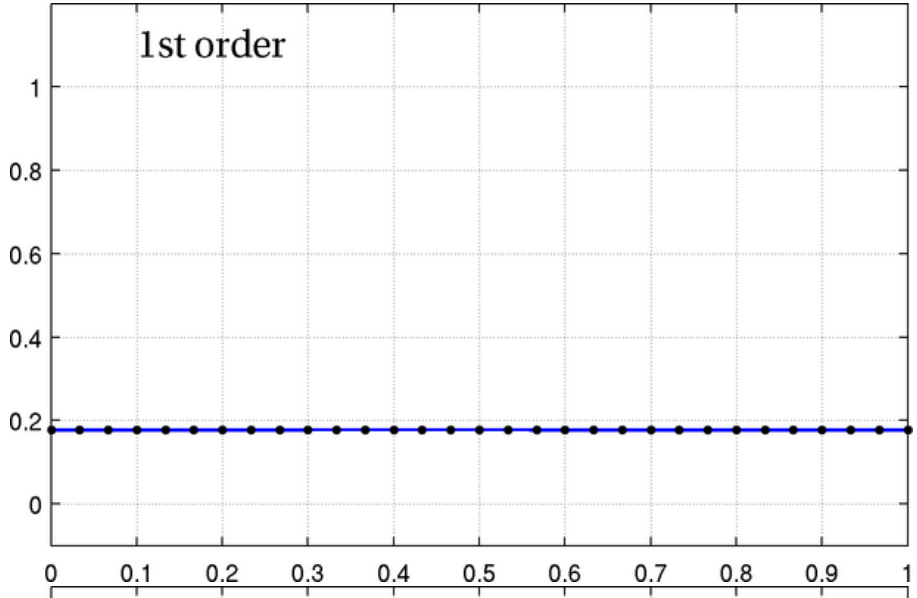


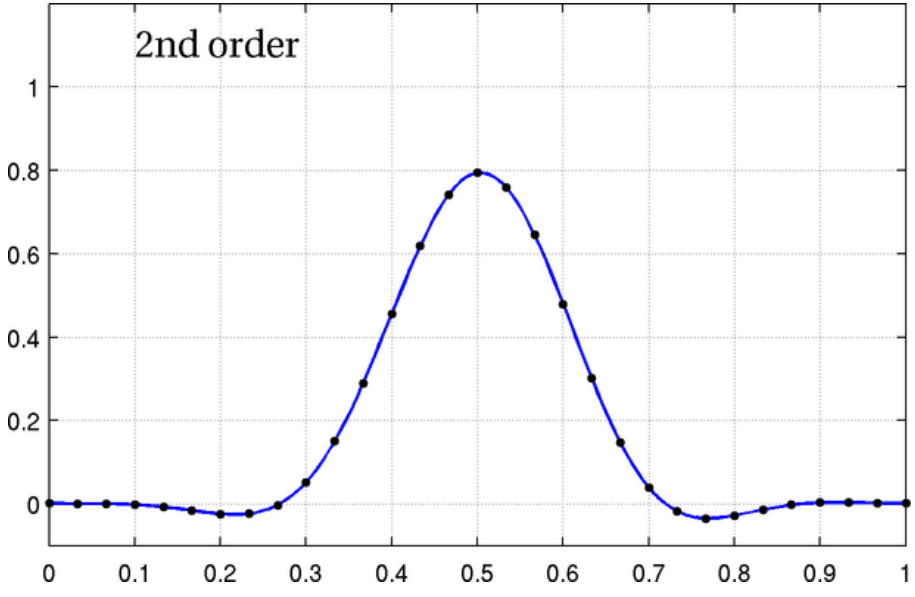


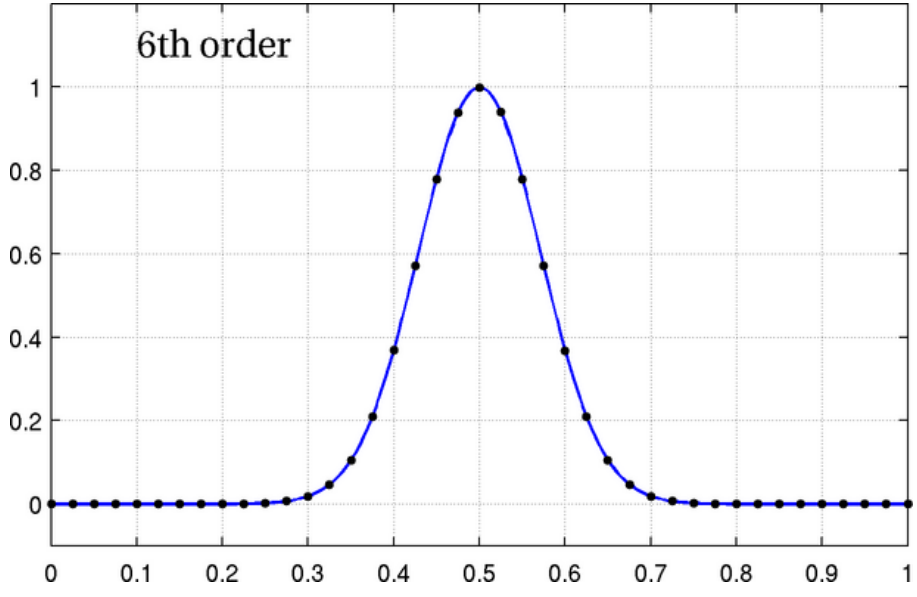
Why high-order accurate methods?

- Scalar convection equation $u_t + u_x = 0$
- High-order gives *superior performance* for *equal resolution*









High-Order Discontinuous Galerkin Simulations

- Discontinuous Galerkin (DG) methods have desirable properties:

	FVM	FDM	FEM	DG
1) High-order/Low dispersion	✗	✓	✓	✓
2) Complex geometries	✓	✗	✓	✓
3) Stabilization for convection	✓	✓	✗	✓

- However, several problems to resolve:
  - High CPU/memory requirements (compared to FVM or H-O FDM)
  - Robustness issues, low tolerance to under-resolved features
  - High-order geometry representation and mesh generation
- *Need to make DG competitive for real-world problems*

Outline

Outline

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:

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

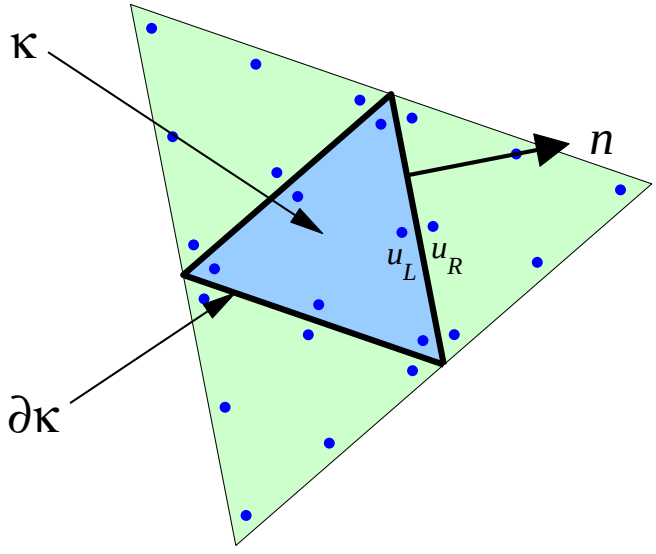
The Discontinuous Galerkin Method

- Integrate by parts:

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

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

- Global problem: 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 solutions



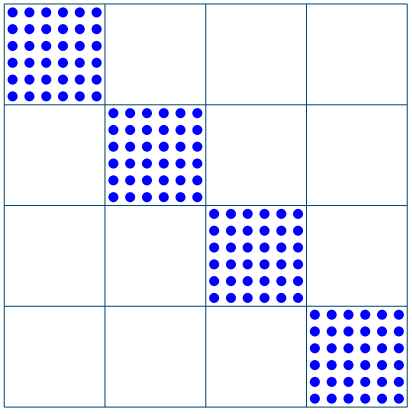
The DG Method – Observations

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

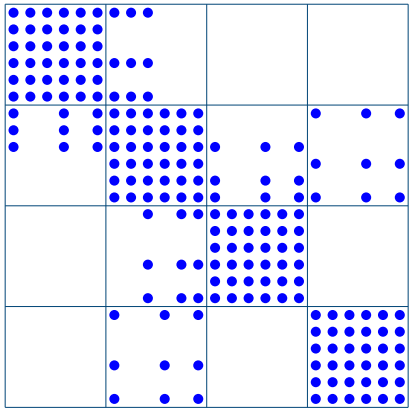
$$(\boldsymbol{u}_h)_t A_\kappa + \int_{\partial\kappa} \hat{\mathcal{F}}_i(\boldsymbol{u}_h^+, \boldsymbol{u}_h^-, \hat{\boldsymbol{n}})\, ds = 0$$

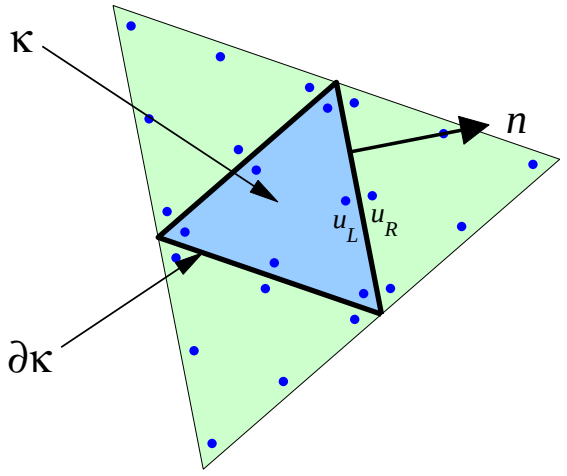
- Boundary conditions enforced naturally for any degree p
- Block-diagonal mass matrix (no overlap between basis functions)
- Block-wise compact stencil – neighboring elements connected

Mass Matrix



Jacobian





Outline

Viscous Discretization

- General approach for second derivatives:
  - Write as system of first order equations [Arnold et al 02]:

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

$$\boldsymbol{\sigma} - \nabla \boldsymbol{u} = 0$$

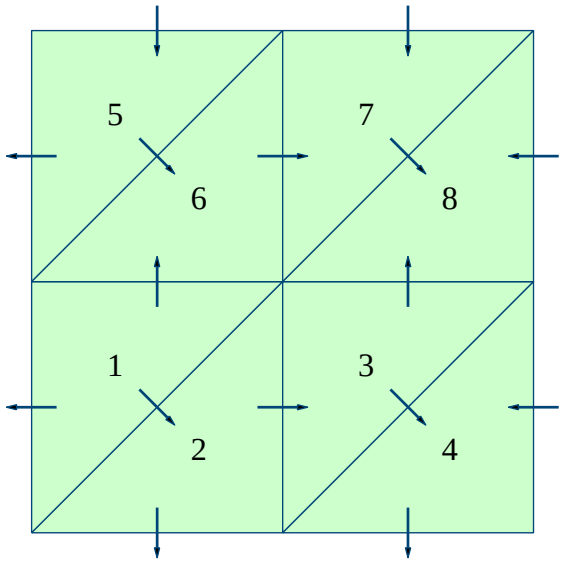
  - Discretize using DG, choose appropriate numerical fluxes $\hat{\boldsymbol{\sigma}}$, $\hat{u}$
- Various schemes have been proposed:
  - *BR2* [Bassi/Rebay 1998]: Different lifting operator for each edge, compact connectivities, similar to Interior Penalty (IP)
  - *LDG* [Cockburn/Shu 1998]: Upwind/Downwind, non-compact
  - *CDG* [Peraire/Persson 2008]:
    Modification of LDG for local dependence – sparse and compact

The Local DG Method

- Consider Poisson problem $-\nabla \cdot (\kappa \nabla u) = f$
- Write as system of first order equations,

$$-\nabla \cdot \boldsymbol{\sigma} = f$$

$$\boldsymbol{\sigma} = \kappa \nabla u$$



- Use numerical inter-element fluxes

$$\hat{\boldsymbol{\sigma}} = \{\boldsymbol{\sigma}_h\} - C_{11}[u_h] + \boldsymbol{C}_{12}[\boldsymbol{\sigma}_h]$$

$$\hat{u} = \{u_h\} - \boldsymbol{C}_{12} \cdot [u_h]$$

where $\{\cdot\}, [\cdot]$ denote averaging and difference

- In particular, choosing $\boldsymbol{C}_{12} = 1$ or -1 depending on a *switch* for each edge, will upwind/downwind $\hat{\boldsymbol{\sigma}}, \hat{u}$

The Local DG Scheme

- Solving for the variables $\boldsymbol{\sigma}_h$ gives

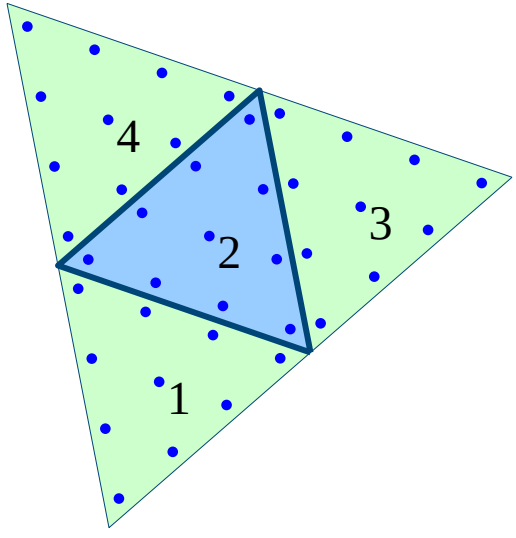
$$\boldsymbol{\sigma}_h = \kappa \nabla_h u_h + \bar{\boldsymbol{\sigma}}_h$$

where

$$\bar{\boldsymbol{\sigma}}_h = \kappa r([u_h]) + \kappa l(\boldsymbol{C}_{12} \cdot [u_h]) + \text{ boundary terms}$$

and $r(\phi)$ and $l(q)$ are *lifting operators* (essentially L_2-projections)

- In general, this introduces non-local couplings since the lifting operators involve all element edges



The Compact DG Scheme

- In the CDG scheme, we split the lifting operators into sums of edge-wise lifting operators $r^e(\phi)$, $l^e(q)$, and set

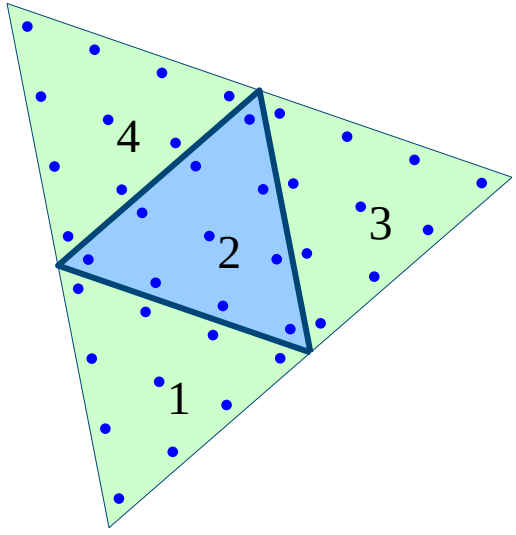
$$\hat{\boldsymbol{\sigma}} = \{\boldsymbol{\sigma}_h^e\} - C_{11}[u_h] + \boldsymbol{C}_{12}[\boldsymbol{\sigma}_h^e]$$

$$\hat{u} = \{u_h\} - \boldsymbol{C}_{12} \cdot [u_h]$$

  where $\boldsymbol{\sigma}_h^e = \kappa \nabla_h u_h + \bar{\boldsymbol{\sigma}}_h^e$, with

$$\bar{\boldsymbol{\sigma}}_h^e = \kappa r^e([u_h]) + \kappa l^e(\boldsymbol{C}_{12} \cdot [u_h]) + \text{ boundary terms}$$

- Since only the lifting operator corresponding to the current edge is used, only neighboring elements are connected



Error Estimates

- In primal form, the LDG scheme becomes (ignoring bnd terms):

$$\int_\Omega \kappa(r([u]) + l(\boldsymbol{C}_{12} \cdot [u])) \cdot (r([v]) + l(\boldsymbol{C}_{12} \cdot [v]))\, dx =$$
$$\sum_{e \in \mathcal{E}_i} \sum_{f \in \mathcal{E}_i} \int_\Omega \kappa(r^e([u]) + l^e(\boldsymbol{C}_{12} \cdot [u])) \cdot (r^f([v]) + l^f(\boldsymbol{C}_{12} \cdot [v]))\, dx$$

- The CDG scheme excludes some terms that are indefinite:

$$\sum_{e \in \mathcal{E}_i} \int_\Omega \kappa(r^e([u]) + l^e(\boldsymbol{C}_{12} \cdot [u])) \cdot (r^e([v]) + l^e(\boldsymbol{C}_{12} \cdot [v]))\, dx =$$
$$\sum_{e \in \mathcal{E}_i} \sum_{f \in \mathcal{E}_i} \delta_{ef} \int_\Omega \kappa(r^e([u]) + l^e(\boldsymbol{C}_{12} \cdot [u])) \cdot (r^f([v]) + l^f(\boldsymbol{C}_{12} \cdot [v]))\, dx$$

- Non-compact terms are eliminated but the scheme remains stable

Error Estimates

- Coercivity and boundedness for the CDG scheme same as for LDG, leading to a-*priori* estimates:

$$|||u - u_h||| \leq Ch^p |u|_{p+1,\Omega}$$

and

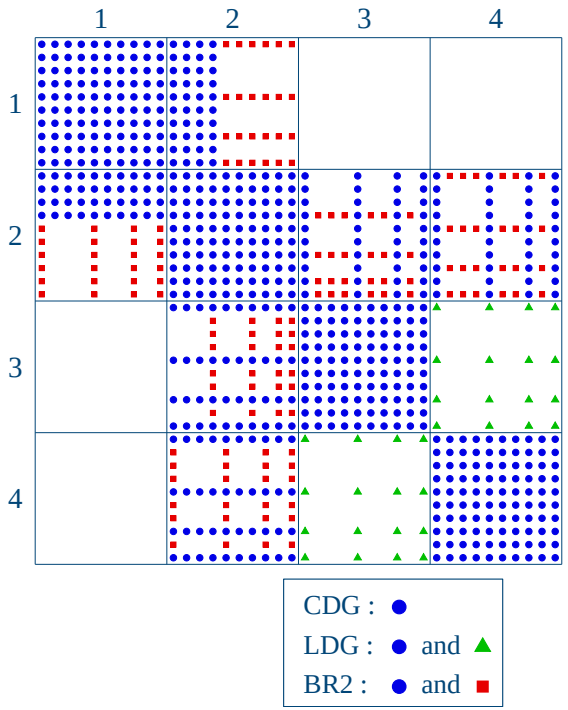
$$||u - u_h||_{0,\Omega} \leq Ch^{p+1} |u|_{p+1,\Omega}$$

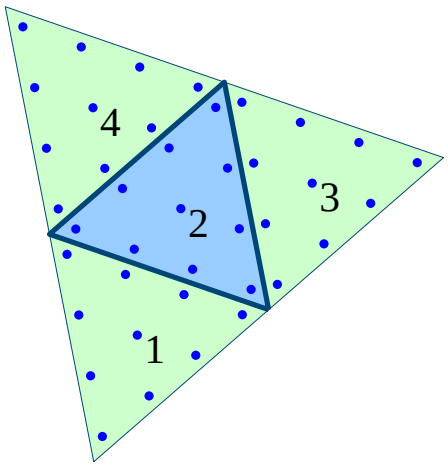
with the norm

$$|||v|||^2 = \sum_{K \in \mathcal{T}_h} |v|^2_{1,K} + \sum_{e \in \mathcal{E}_i} ||r_e([v])||^2_{0,\Omega} + \sum_{e \in \partial\Omega_D} ||r_D(v)||^2_{0,\Omega}$$

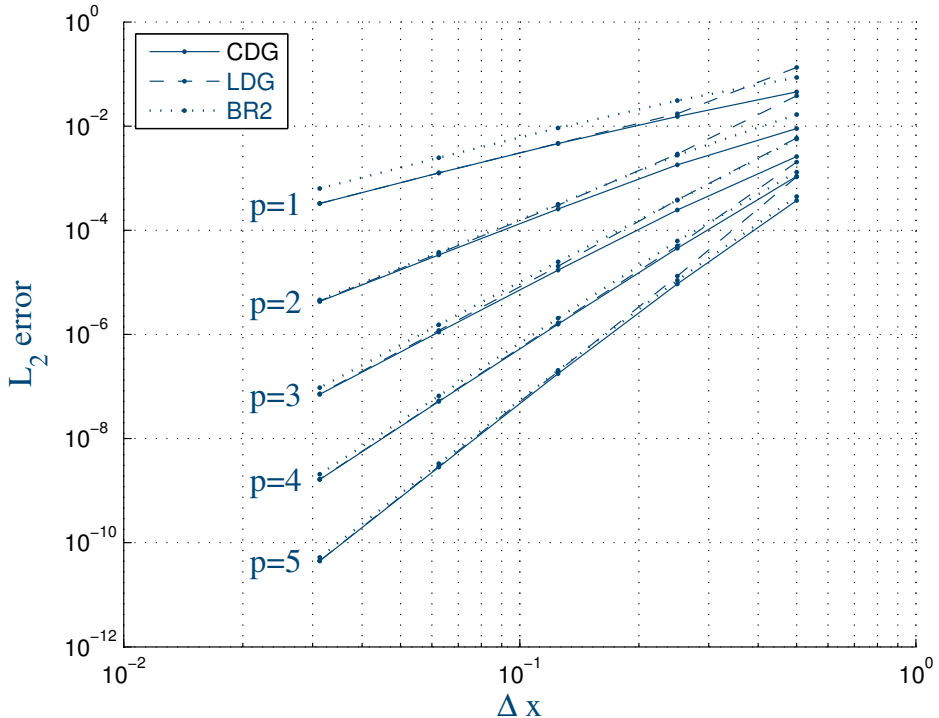
- Assumes $C_{11} = \mathcal{O}(h^{-1})$, but is observed numerically for $C_{11} = 0$

The CDG Method – Summary





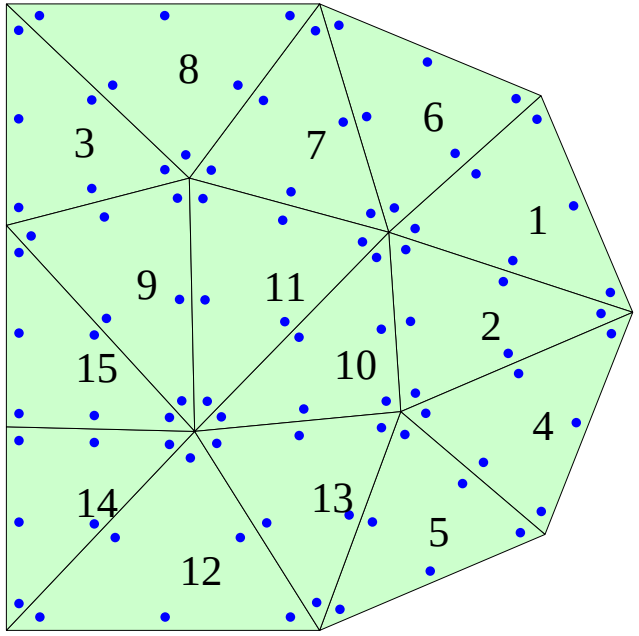
- Modification of numerical fluxes in LDG scheme
- Excludes non-compact and indefinite terms
- Provably optimal accuracy $\mathcal{O}(h^{p+1})$
- Higher stability/accuracy than LDG/BR2
- Sparser than LDG/BR2/IP

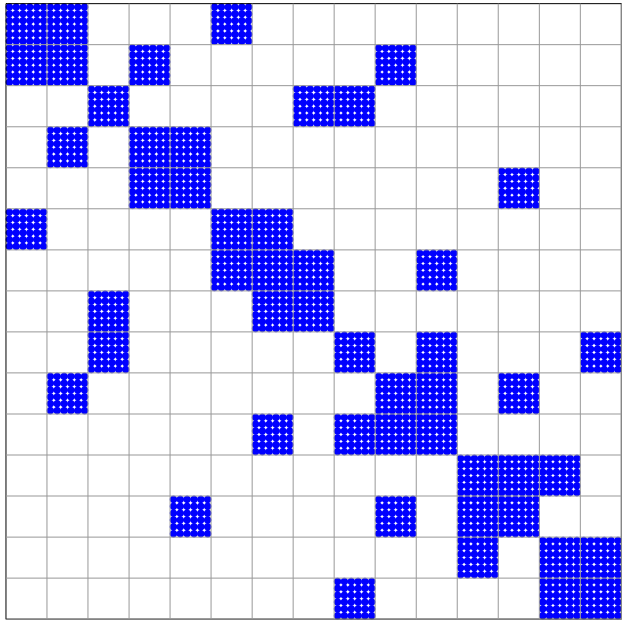


Outline

Jacobian Matrix Sparsity Structure

- Dual mesh connectivity, with each entry a large complete graph $\boldsymbol{A}_{ij}$
- Off-diagonal blocks actually sparser with CDG, but assume dense for simplicity
- Size N of submatrices $\boldsymbol{A}_{ij}$ is often > 100
- Block-based storage format essential for high performance using BLAS routines
- For other structures (e.g. elimination matrices), use block-wise compressed column





Preconditioners for Krylov Methods

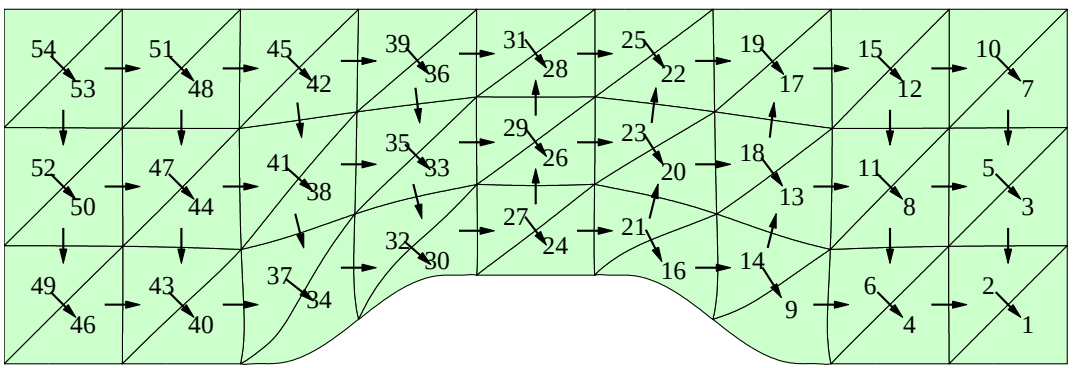
- Preconditioning required for fast convergence in Krylov methods
- Standard point-wise Jacobi, ILU, etc, ineffective for DG
- Block Jacobi and Gauss Seidel are generally poor:

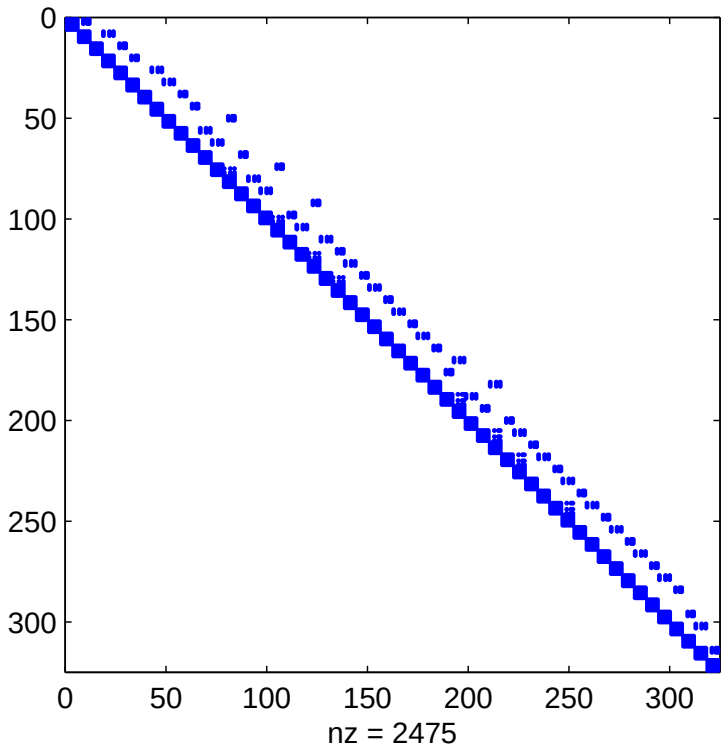
$$\tilde{\boldsymbol{A}}_{ij}^{\mathrm{J}} = \begin{cases} \boldsymbol{A}_{ij} & \text{if } i = j, \\ \boldsymbol{0} & \text{if } i \neq j, \end{cases} \qquad \text{and} \qquad \tilde{\boldsymbol{A}}_{ij}^{\mathrm{GS}} = \begin{cases} \boldsymbol{A}_{ij} & \text{if } i \leq j, \\ \boldsymbol{0} & \text{if } i > j. \end{cases}$$

- Block-ILU(0) algorithm $\tilde{\boldsymbol{A}}^{\mathrm{ILU}} = \tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}}$ effective for good orderings
- Block-ILU(0) postsmoothing for coarse scale correction [Persson, Peraire 2008], cheap, general purpose preconditioner

Element Order Dependency

- Properties of Gauss Seidel and ILU preconditioners highly dependent on the ordering of the elements
- For *upwinded scalar convection*, “ordering by lines” gives an optimal upper triangular matrix
- But for viscous or multivariate problems, best ordering not clear
- Matrix-approach: Minimize error in the approximations, rather than using physical observations





Minimum Discarded Fill Element Ordering

- Greedy algorithm for element ordering [Persson, Peraire 2008]:
  At step j, if j' is chosen next, we would discard the fill

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

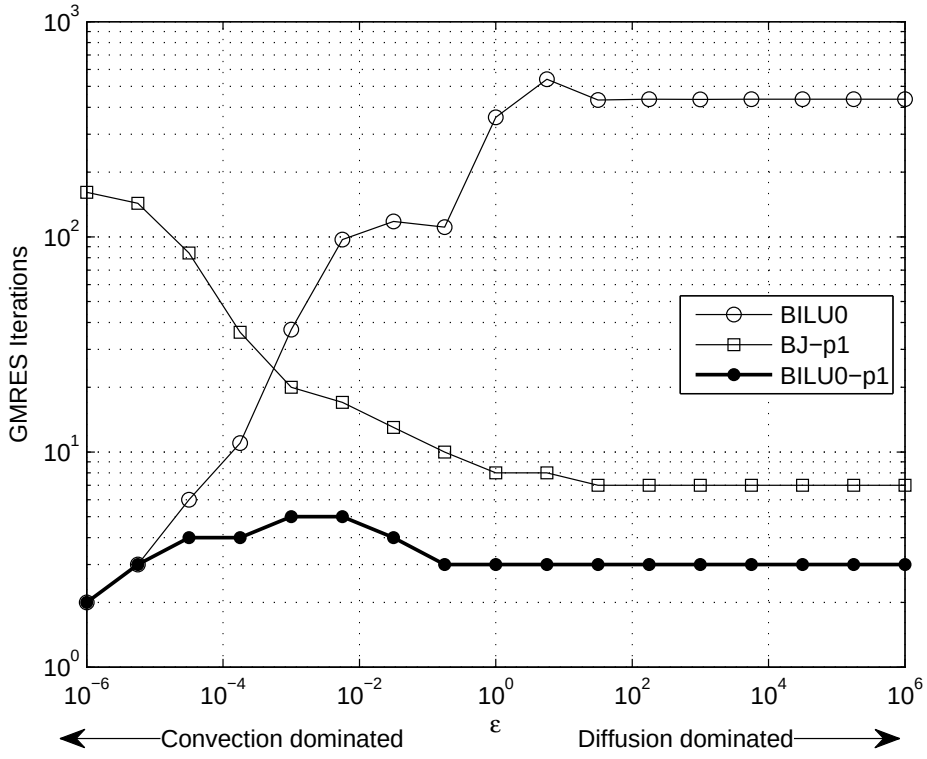
- Choose the j' that minimizes the norm of the discarded fill

$$w^{(j,j')} = \|\Delta\tilde{\boldsymbol{U}}^{(j,j')}\|_{\mathrm{F}}$$

- Some simplifications, min-heap data structure $\Longrightarrow \mathcal{O}(n \log n)$ cost
- Increased locality: Consider only neighbors for j'

Effect of Ordering on ILU

- MDF ordering makes block-ILU0 with coarse grid correction almost perfect for convection-diffusion
- Good element ordering critical for Navier-Stokes as well:



Problem	Element Ordering		
	Random	RCM	MDF
Inviscid	51	27	12
Laminar, Re=1,000	200	135	12
Laminar, Re=20,000	197	139	27
RANS, Re=10^6	98	99	18

Convergence – Model Navier-Stokes Problem

$\times$ = No convergence after 1,000 iterations

Problem	Parameters		Preconditioner/Iterations					
			Block Jacobi		Block G-S		Block ILU0	
	Δt	M	BJ	BJ-p1	BGS	BGS-p1	BILU0	BILU0-p1
Inviscid	10^{-3}	0.2	24	19	14	11	5	4
	10^{-1}	0.2	187	85	73	49	12	6
	∞	0.2	840	142	456	72	40	9
	10^{-3}	0.01	200	112	111	67	15	6
	10^{-1}	0.01	$\times$	$\times$	$\times$	532	94	10
	∞	0.01	$\times$	$\times$	$\times$	$\times$	374	16
Laminar	10^{-3}	0.2	50	34	25	19	4	4
Re=1,000	10^{-1}	0.2	$\times$	597	477	225	11	5
	∞	0.2	$\times$	$\times$	$\times$	$\times$	37	7
	10^{-3}	0.01	98	66	51	33	8	5
	10^{-1}	0.01	$\times$	619	$\times$	207	27	9
	∞	0.01	$\times$	$\times$	$\times$	748	135	12

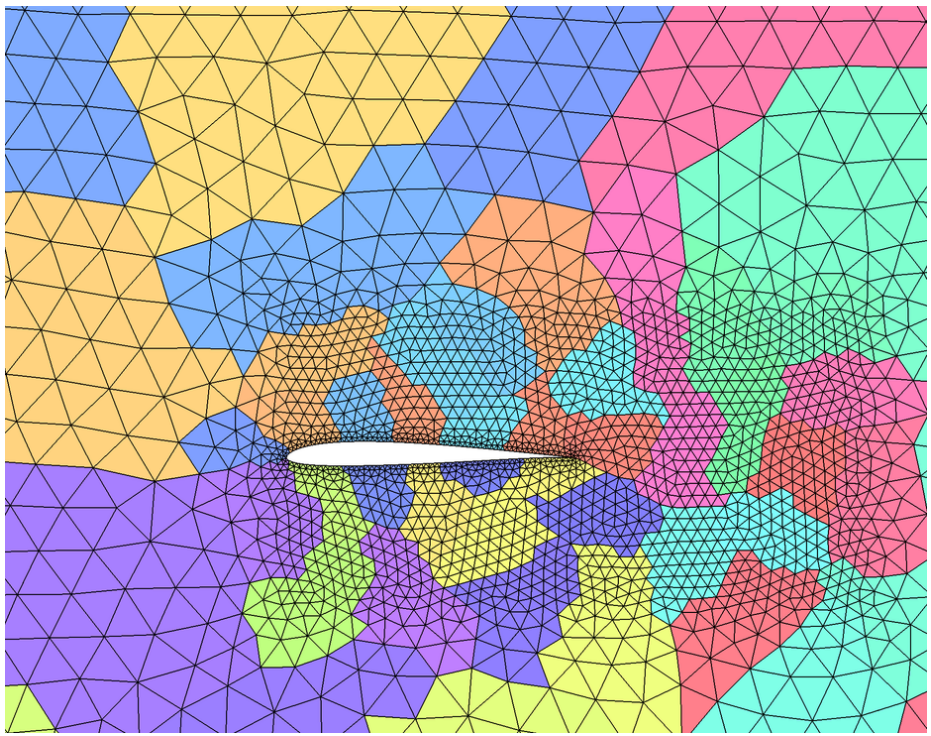
Convergence – Model Navier-Stokes Problem

$\times$ = No convergence after 1,000 iterations

Problem	Parameters		Preconditioner/Iterations					
			Block Jacobi		Block G-S		Block ILU0	
	Δt	M	BJ	BJ-p1	BGS	BGS-p1	BILU0	BILU0-p1
Laminar	10^{-3}	0.2	26	19	14	11	4	4
Re=20,000	10^{-1}	0.2	456	220	219	113	16	8
	∞	0.2	$\times$	$\times$	$\times$	$\times$	236	20
	10^{-3}	0.01	160	90	61	38	12	6
	10^{-1}	0.01	$\times$	$\times$	$\times$	735	80	16
	∞	0.01	$\times$	$\times$	$\times$	$\times$	$\times$	35
RANS	10^{-3}	0.2	76	56	33	28	8	7
Re=10^6	10^{-1}	0.2	$\times$	$\times$	$\times$	$\times$	35	25
	∞	0.2	$\times$	$\times$	$\times$	$\times$	70	18
	10^{-3}	0.01	411	231	174	110	14	9
	10^{-1}	0.01	$\times$	$\times$	$\times$	$\times$	46	16
	∞	0.01	$\times$	$\times$	$\times$	$\times$	132	28

ILU Parallelization – Domain Decomposition

- In parallel, use partition-wise ILUs with MDF ordering
- Partition using the weights $C_{ij} = \|\boldsymbol{A}_{ii}^{-1}\boldsymbol{A}_{ij}\|_F$
- Essentially a "non-overlapping Schwartz preconditioner with incomplete solutions"
- Approaches Jacobi as # partitions $\rightarrow$ # elements
- Good option for many problems – GMRES iterations cheap compared to matrix creation

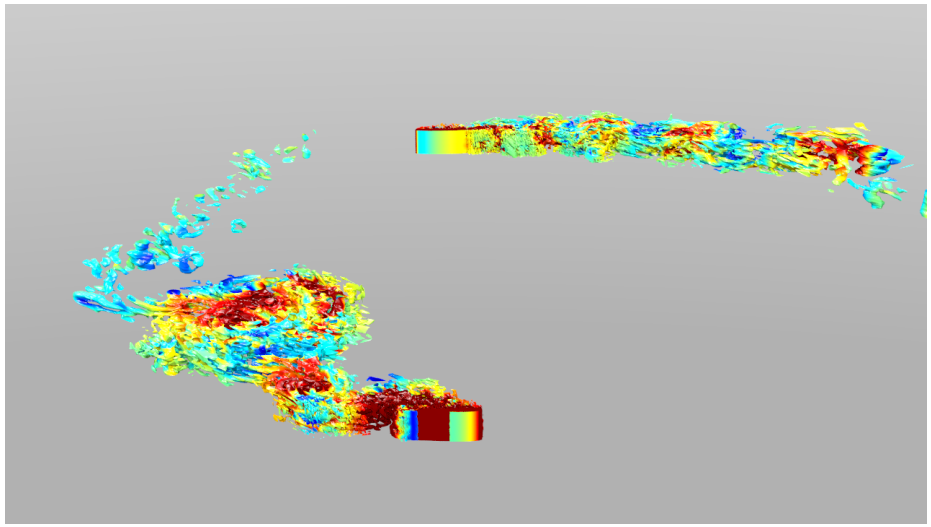


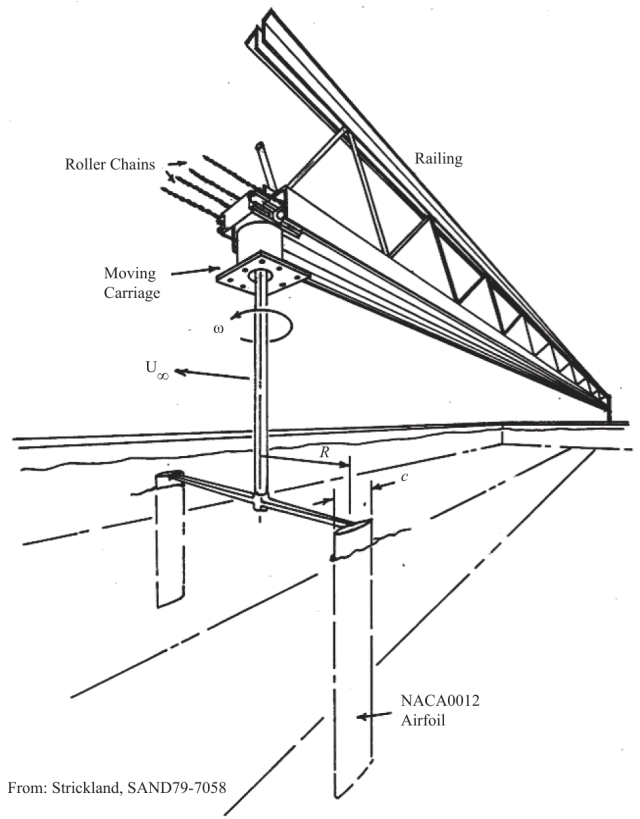
Sandia National Laboratories' Low Re VAWT

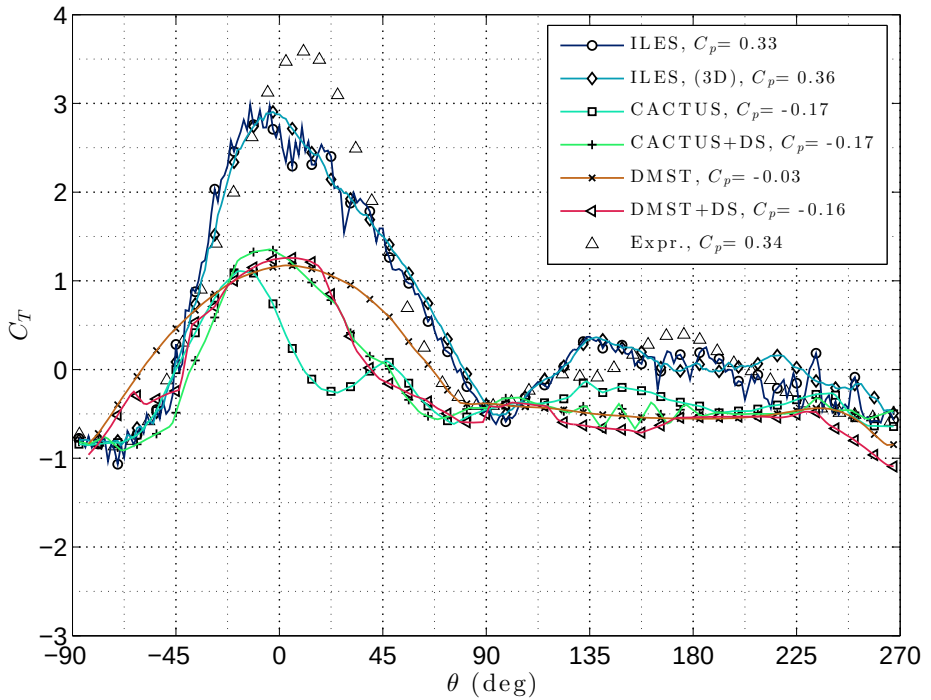
Sandia National Labs tow tank experiment from 1979

- ILES simulation of vertical axis wind turbines (VAWT)

  [Kanner/Persson, AIAA J., 2015]

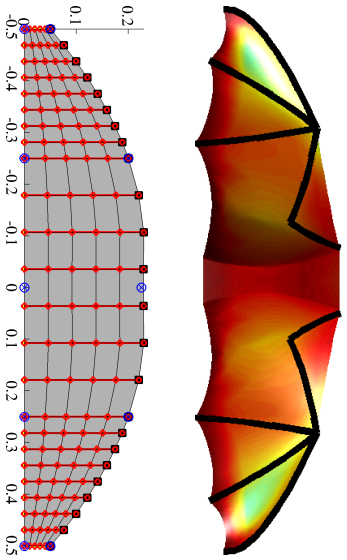


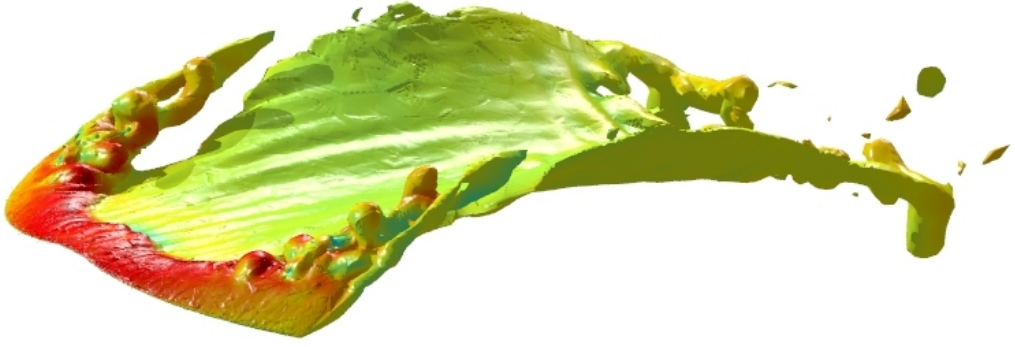


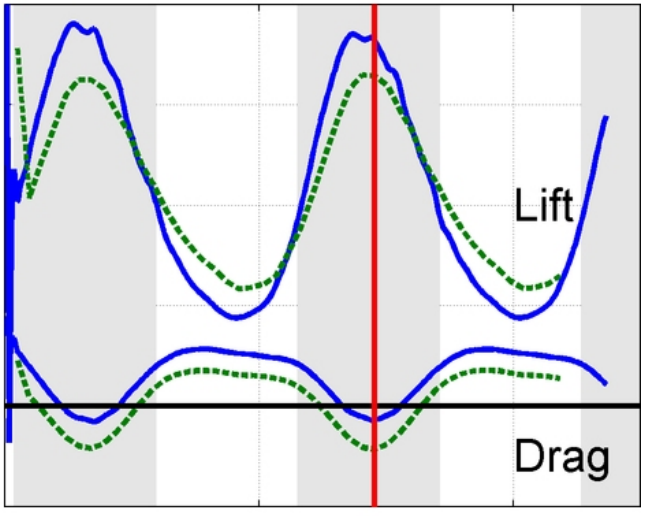


Optimal Flapping Kinematics

- **Goal:** To design and analyze an effective flapping wing shape for cruising flight
- A *multi-fidelity approach* with a range of simulation tools [Willis & Persson 2011, 2014]

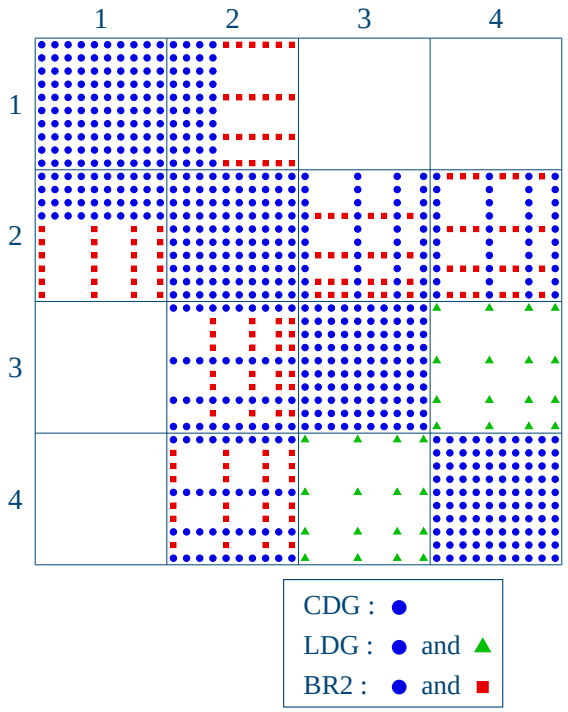






Outline

Computational Cost – DG Jacobian Matrices

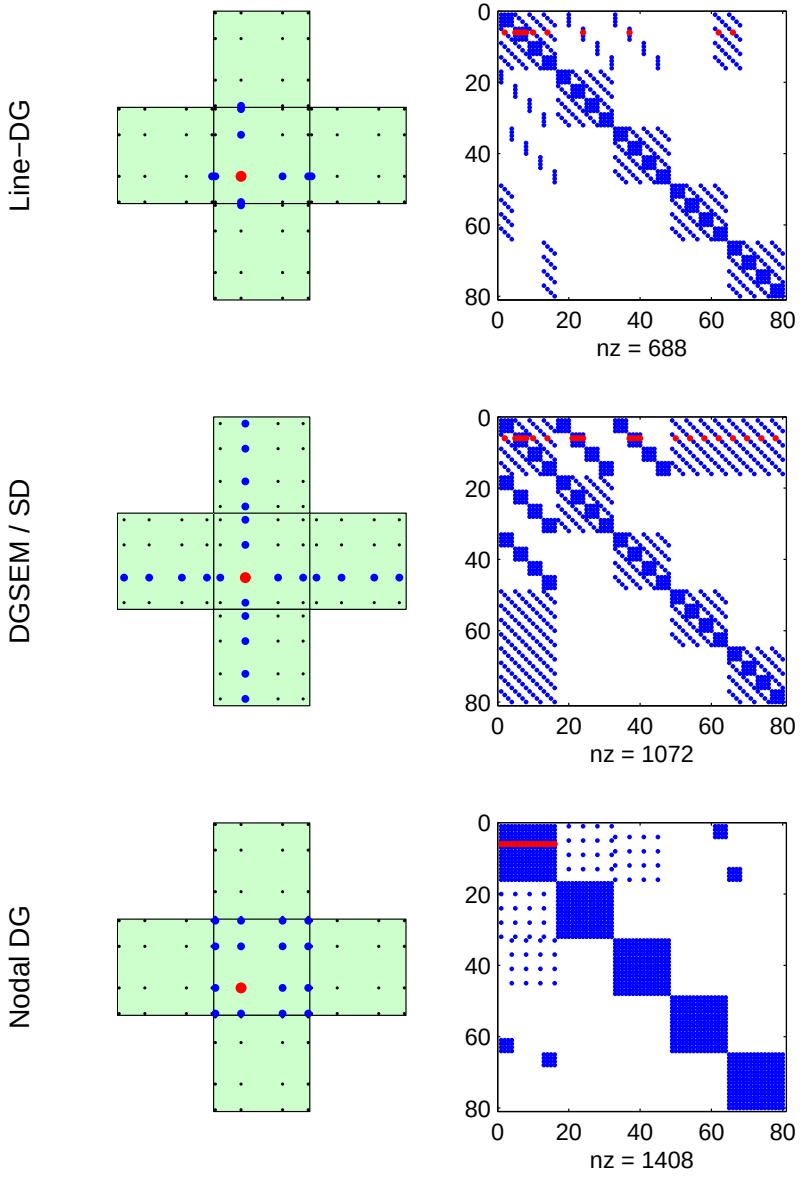


- Number of non-zeros in Jacobian matrix for nodal DG (C solution components, T simplex elements, degree p, dimension D)
- Nodes per element $S = \binom{p+D}{p} = \mathcal{O}(p^D)$, nodes per face $s = \binom{p+D-1}{D-1} = \mathcal{O}(p^{D-1})$
- Example: 3-D Navier-Stokes, $p = 3$, $T = 100,000$:

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

DG Methods – Stencil comparison

- Nodal DG couples all nodes inside element $\Longrightarrow$ stencil size $\mathcal{O}(p^D)$
- Schemes with finite-difference type coupling have stencil size $\mathcal{O}(Dp)$
- Line-DG [Persson, JCP 2012]:
  Apply 1-D DG for each coordinate
- Plots show first-order operator
- Assumptions:
  - SD/DGSEM based on Gauss-Legendre solution nodes
  - Nodal-DG consistently integrated
- Line-DG maximizes sparsity:
  - Line-based stencils
  - Solution nodes on edges

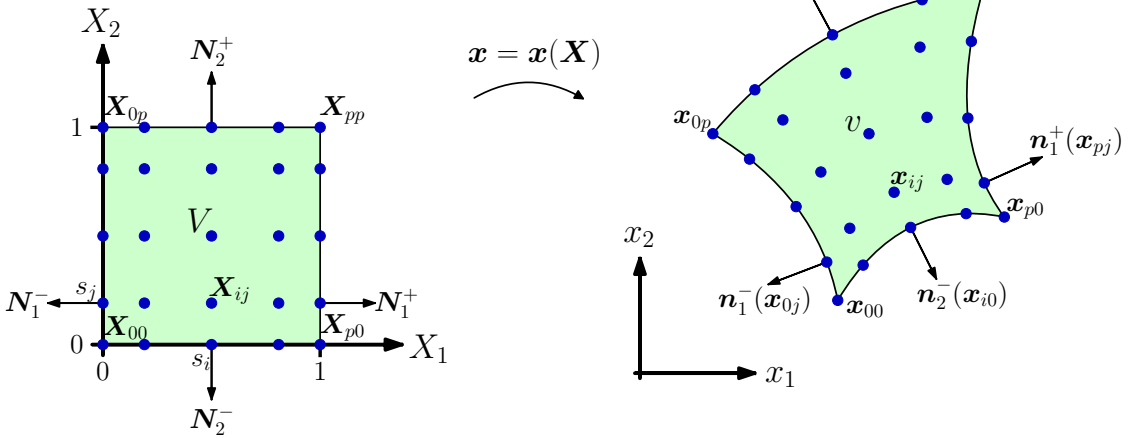


Line-based Discontinuous Galerkin

- Map system of conservation law from v to reference element V:

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

$$J\frac{\partial \boldsymbol{u}}{\partial t} + \nabla_X \cdot \widetilde{\boldsymbol{F}}(\boldsymbol{u}) = \boldsymbol{0}$$



where $\widetilde{\boldsymbol{F}} = (\widetilde{\boldsymbol{f}}_1, \widetilde{\boldsymbol{f}}_2, \widetilde{\boldsymbol{f}}_3) = J\boldsymbol{G}^{-1}\boldsymbol{F}$, with $\boldsymbol{G} = \nabla_X \boldsymbol{x}$ and $J = \det(\boldsymbol{G})$

- Consider curve $\boldsymbol{x}_{jk}(\xi) = \boldsymbol{x}(\xi, X_j, X_k)$, find $\boldsymbol{r}_{jk}(X_1) \approx \partial \widetilde{\boldsymbol{f}}_1 / \partial X_1$ by a 1-D DG procedure: Find $\boldsymbol{r}_{jk}(\xi) \in \mathcal{P}_p([0,1])^m$ such that

$$\int_0^1 \boldsymbol{r}_{jk}(\xi) \cdot \boldsymbol{v}(\xi)\, d\xi = \int_0^1 \frac{d\widetilde{\boldsymbol{f}}_1(\boldsymbol{u}_{jk}(\xi))}{d\xi} \cdot \boldsymbol{v}(\xi)\, d\xi$$

$$= \widehat{\widetilde{\boldsymbol{f}}}_1(\boldsymbol{u}_{jk}^+(1), \boldsymbol{u}_{jk}(1)) \cdot \boldsymbol{v}(1) - \widehat{\widetilde{\boldsymbol{f}}}_1(\boldsymbol{u}_{jk}(0), \boldsymbol{u}_{jk}^-(0)) \cdot \boldsymbol{v}(0) - \int_0^1 \widetilde{\boldsymbol{f}}_1(\boldsymbol{u}_{jk}(\xi)) \cdot \frac{d\boldsymbol{v}}{d\xi}\, d\xi$$

Line-based Discontinuous Galerkin

- Use existing approximate Riemann solver as-is
- Find $\boldsymbol{r}_{jk}(\xi)$ by standard finite element procedure

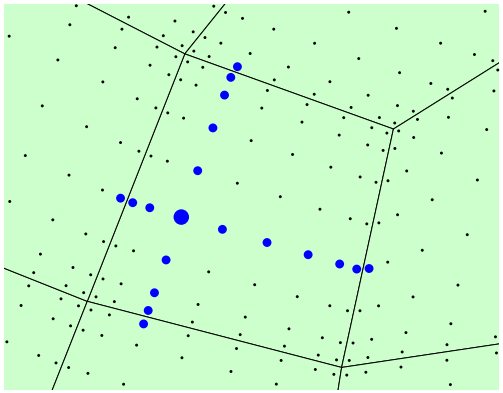
$$\boldsymbol{u}_{jk}(\xi) = \sum_{i=0}^{p} \boldsymbol{u}_{ijk}\phi_i(\xi), \quad \boldsymbol{r}_{jk}(\xi) = \sum_{i=0}^{p} \boldsymbol{r}_{ijk}\phi_i(\xi)$$

- Discrete form $\boldsymbol{M}\boldsymbol{r}_{jk} = \boldsymbol{b}$, find $\boldsymbol{r}_{jk}$ by solving m linear systems with $(p+1)$-by-$(p+1)$ mass matrix $\boldsymbol{M}$
- Repeat along each direction to obtain semi-discrete formulation:

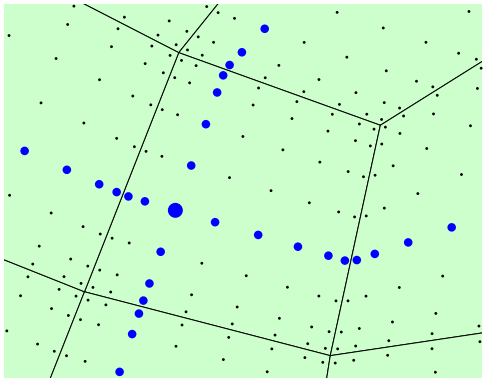
$$J_{ijk}\frac{d\boldsymbol{u}_{ijk}}{dt} + \sum_{n=1}^{3} \boldsymbol{r}_{ijk}^{(n)} = \boldsymbol{0}$$

- Observations:
  - All integrals are one-dimensional
  - No statement about integration/flux points: Integrals assumed exact
  - Numerical fluxes only evaluated point-wise

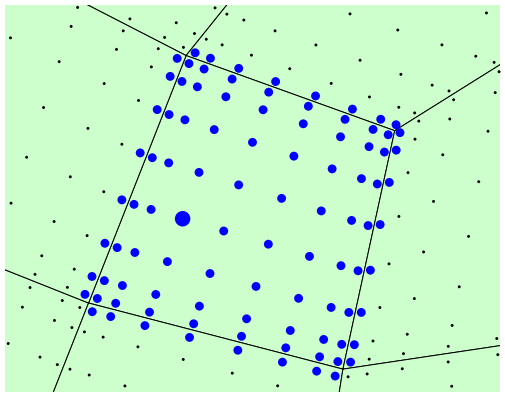
Sparsity patterns



Line-DG



SD / DGSEM



Standard Nodal DG

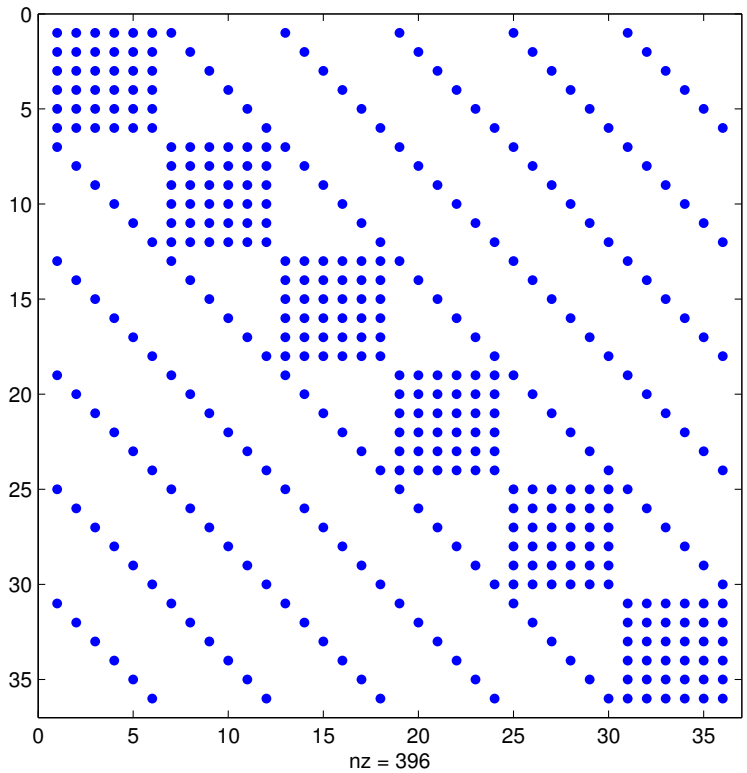
connectivities / node (3-D hexahedrals)

	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
	DGSEM/SD connectivities	11	17	23	29	35	41	47	53	59	65
	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
	DGSEM/SD connectivities	16	25	34	43	52	61	70	79	88	97
	Nodal-DG connectivities	20	45	88	155	252	385	560	783	1060	1397

- For $p = 3$ the Line-DG method is 5.5 times sparser than nodal DG, and for $p = 10$ it is almost 40 times sparser

Line-DG – Efficient Block Solver

- Block-Jacobi, block-ILU, etc, need efficient block solvers that take advantage of the sparsity pattern
- Direct solvers
  - No good separators, fill
- Point-wise Jacobi, ILU, etc
  - As before, very poor convergence
- FFT
  - Only for certain operators / discretizations
- Alternating Direction Implicit (ADI)
  - Highly problem dependent
- Tensor product methods
  - Not exactly tensor products



Tensor Product Matrix Diagonalization

[Lynch, Rice, Thomas, 1964]

- Suppose the matrix has the form

$$K = I \otimes A + B \otimes I$$

- System $Kx = f$ with $x = \mathrm{vec}(X)$ and $f = \mathrm{vec}(F)$ can then be written

$$AX + XB^T = F$$

- Find eigenvalue decompositions of A and B

$$A = V_A \Lambda_A V_A^{-1}, \qquad B = V_B \Lambda_B V_B^{-1}$$

- Plug in, left-multiply by V_A^{-1}, right-multiply by V_B^T

$$\Lambda_A (V_A^{-1} X V_B^T) + (V_A^{-1} X V_B^T) \Lambda_B = V_A^{-1} F V_B^T$$

or $\Lambda_A W + W \Lambda_B = G$, with solution

$$W_{ij} = \frac{1}{\lambda_{A,i} + \lambda_{B,j}} G_{ij}$$

Tensor Product Matrix Diagonalization

[Lynch, Rice, Thomas, 1964]

- More generally, if

$$K = A \otimes B + C \otimes D,$$

multiply by $A^{-1} \otimes D^{-1}$,

$$(A^{-1} \otimes D^{-1})K = I \otimes (D^{-1}B) + (A^{-1}C) \otimes I$$

and apply the same technique as before

Tensor Product Approximation

- Problem: In general the matrix is not in tensor product form
- However, it might be close enough to form a preconditioner
- [Van Loan, 2000] Find approximation

$$K \approx A \otimes B + C \otimes D$$

using the Kronecker product SVD:

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1N} \\ \vdots & \ddots & \vdots \\ A_{M1} & \cdots & A_{MN} \end{bmatrix} = \sum_{k=1}^{MN} \sigma_k B_k \otimes C_k$$

- Keep first 2 terms to minimize

$$\|K - A \otimes B - C \otimes D\|_F$$

- Compute efficiently using block power iterations / Arnoldi

Numerical Results – Single Block

- Consider convection problem

$$\nabla \cdot \begin{pmatrix} a(x,y) \\ b(x,y) \end{pmatrix} = 0$$

  on a single square block with $p = 5$

GMRES iterations for Jacobi:

Problem	Tensor product	Full block
$a(x,y) = \text{const}, b(x,y) = \text{const}$	1	1
$a(x,y) = a(x), b(x,y) = b(y)$	1	1
$a(x,y) = 1 + y, b(x,y) = 1 + x$	7	1
$a(x,y) = (1 + x^2)(1 + y) + 1, b(x,y) = 4x + y + 2$	7	1

Numerical Results – Multiple Elements

- Consider convection problem

$$\nabla \cdot \begin{pmatrix} a(x,y) \\ b(x,y) \end{pmatrix} = 0$$

  on a 10-by-10 grid of elements with $p = 5$

GMRES iterations for Jacobi:

Problem	Tensor product	Full block
$a(x,y) = \text{const}, b(x,y) = \text{const}$	19	19
$a(x,y) = a(x), b(x,y) = b(y)$	19	19
$a(x,y) = 1 + y, b(x,y) = 1 + x$	21	20
$a(x,y) = (1 + x^2)(1 + y) + 1, b(x,y) = 4x + y + 2$	21	20

Numerical Results – Convection-Diffusion

- Consider convection-diffusion problem

$$\nabla \cdot \begin{pmatrix} a(x,y) \\ b(x,y) \end{pmatrix} - \nabla \cdot (\varepsilon \nabla u) = 0$$

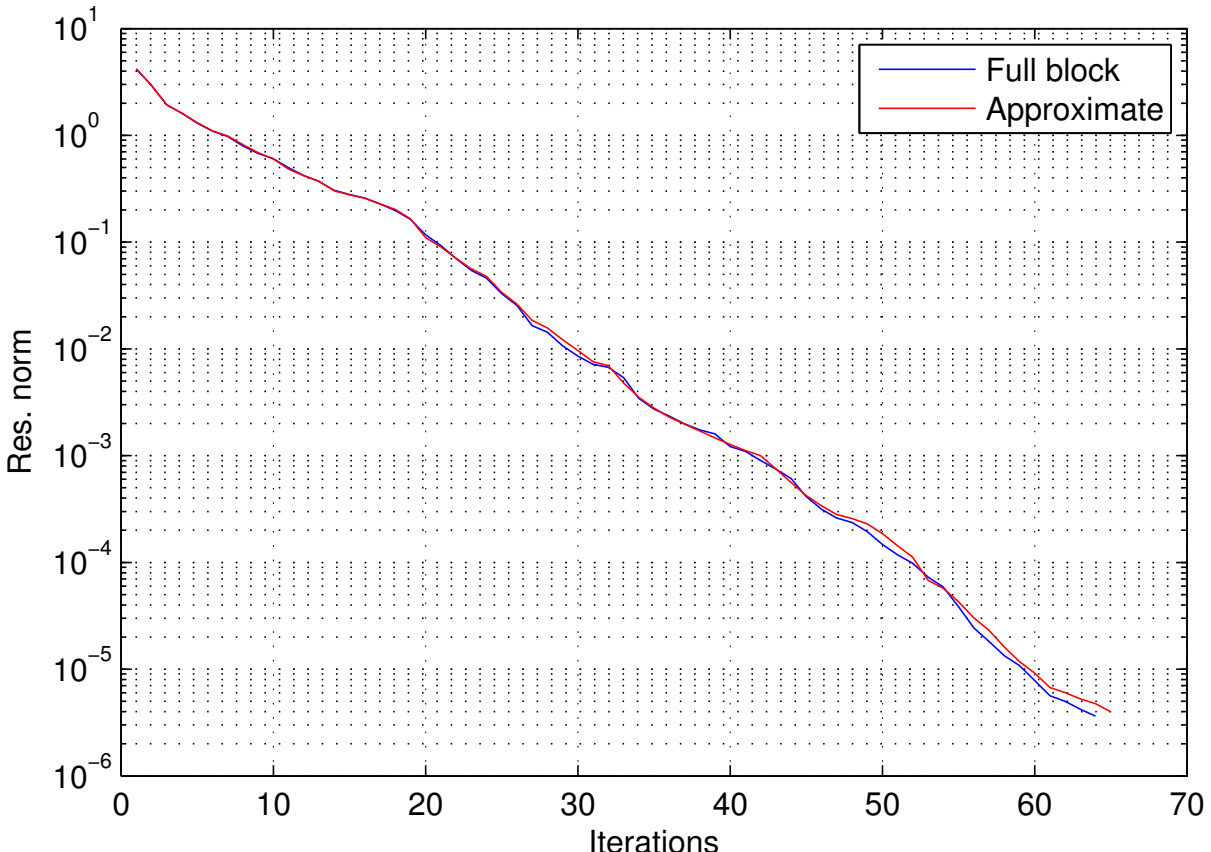
  on a 10-by-10 grid of elements with $p = 5$

GMRES iterations for Jacobi:

Problem	Tensor product	Full block
$a(x,y) = \text{general}, b(x,y) = \text{general}, \varepsilon = 10^{-3}$	28	28
$a(x,y) = \text{general}, b(x,y) = \text{general}, \varepsilon = 10^{-1}$	137	137

Numerical Results – Euler equations

- Consider the Euler equations (compressible gas dynamics)
- System of 4 coupled variables – diagonalize each component separately and solve 4-by-4 system
- Non-linear but well approximated by tensor products



Extensions

- 3-D
  - Unclear how to approximate

$$K \approx A_1 \otimes B_1 \otimes C_1 + A_2 \otimes B_2 \otimes C_2 + A_3 \otimes B_3 \otimes C_3$$

    using KP-SVD
  - Unclear how to solve this using matrix diagonalization (in the general case)
- Incomplete LU
  - Additional terms from the LU updates of the diagonal blocks

Outline

Methods for Deforming Domains

- Arbitrary Lagrangian-Eulerian (ALE) methods:
  - Body-fitted mesh motion based on a time-varying mapping
  - Remeshing techniques needed for large deformations, which often reduce the accuracy
  - Special treatment for satisfaction of the geometric conservation law
- Space-time methods:
  - Fully consistent discretization in both space and time
  - Allow for arbitrary domain deformations and topology changes
  - Need for robust and efficient 3D/4D mesh generators
- Here we present a number of solutions for large deformations, all based on a mesh moving technique with entirely localized operations: [Wang/Persson 2013, 2015; Wang Ph.D. thesis 2015]
  1. A space-time DG discretization for any order and space dimension
  2. A fully unstructured space-time mesh generator for 2D/3D + time
  3. A combined implicit ALE / L_2-projection approach

Outline

ALE Formulation for Deforming Domains

- Use mapping-based ALE formulation for moving domains

  [Visbal,Gaitonde 2002], [Persson,Bonet,Peraire 2009]

- Map from reference domain V to physical deformable domain $v(t)$
- Introduce the *mapping deformation gradient* $\boldsymbol{G} = \boldsymbol{\nabla}_X \boldsymbol{\mathcal{G}}$ and the *mapping velocity* $\boldsymbol{v}_X = \frac{\partial \boldsymbol{\mathcal{G}}}{\partial t}\big|_X$, and set $g = \det(\boldsymbol{G})$
- For numerically computed grid motions, compute *stage consistent* velocities by imposing

$$\boldsymbol{x}_i = \boldsymbol{x}_0 + \Delta t \sum_{j=1}^{s} a_{ij} \boldsymbol{\nu}_j \quad \Longrightarrow \quad \boldsymbol{\nu}_i = \sum_{j=1}^{s} (A^{-1})_{ij} \frac{\boldsymbol{x}_j - \boldsymbol{x_0}}{\Delta t}, \quad i = 1, \ldots, s,$$

  where A is the implicit Runge-Kutta Butcher tableaux

  [Froehle & Persson, 2014]

- Transform equations to account for the motion

Transformed Equations

- The system of conservation laws in the physical domain $v(t)$

$$\left.\frac{\partial \boldsymbol{U}_x}{\partial t}\right|_x + \boldsymbol{\nabla}_x \cdot \boldsymbol{F}_x(\boldsymbol{U}_x, \boldsymbol{\nabla}_x \boldsymbol{U}_x) = 0$$

  can be written in the reference configuration V as

$$\left.\frac{\partial \boldsymbol{U}_X}{\partial t}\right|_X + \boldsymbol{\nabla}_X \cdot \boldsymbol{F}_X(\boldsymbol{U}_X, \boldsymbol{\nabla}_X \boldsymbol{U}_X) = 0$$

  where

$$\boldsymbol{U}_X = g\boldsymbol{U}_x \,, \qquad \boldsymbol{F}_X = g\boldsymbol{G}^{-1}\boldsymbol{F}_x - \boldsymbol{U}_X\boldsymbol{G}^{-1}\boldsymbol{v}_X$$

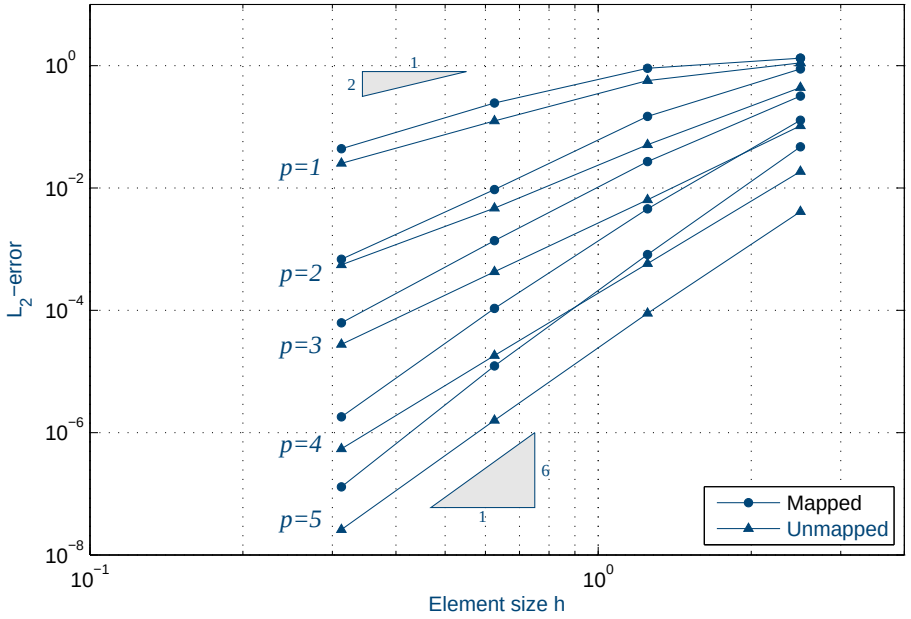
  and

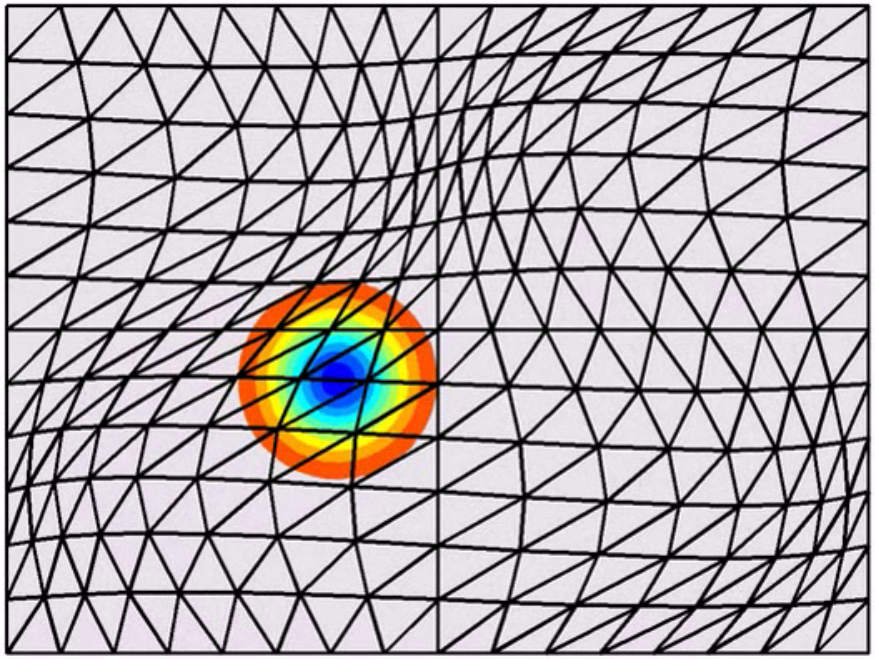
$$\boldsymbol{\nabla}_x \boldsymbol{U}_x = \boldsymbol{\nabla}_X(g^{-1}\boldsymbol{U}_X)\boldsymbol{G}^{-T} = (g^{-1}\boldsymbol{\nabla}_X\boldsymbol{U}_X - \boldsymbol{U}_X\boldsymbol{\nabla}_X(g^{-1}))\boldsymbol{G}^{-T}$$

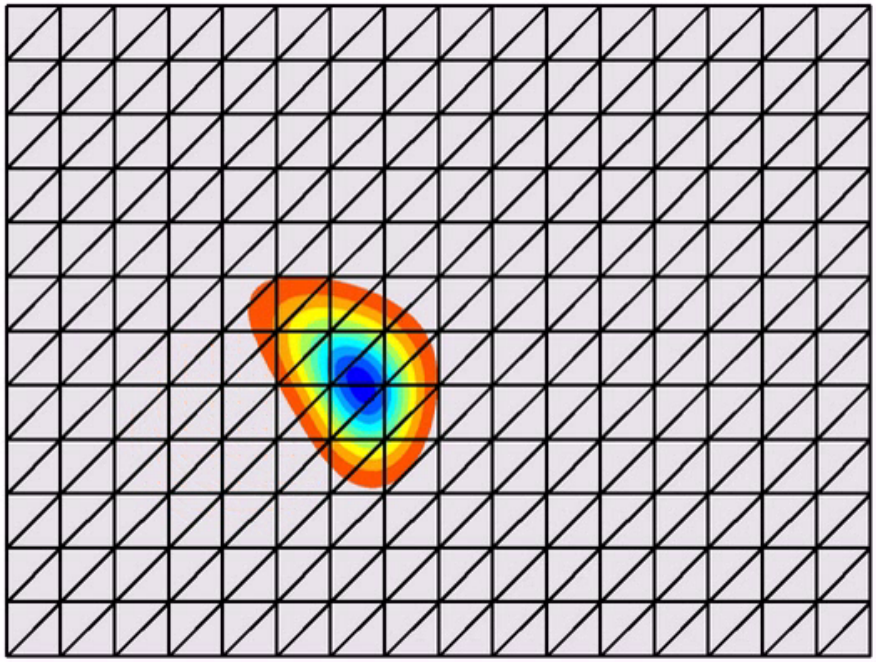
- Details in [Persson,Bonet,Peraire 2009], including how to satisfy the so-called Geometric Conservation Law (GCL)

ALE Formulation for Deforming Domains

- Mapping-based formulation gives arbitrarily high-order accuracy in space and time

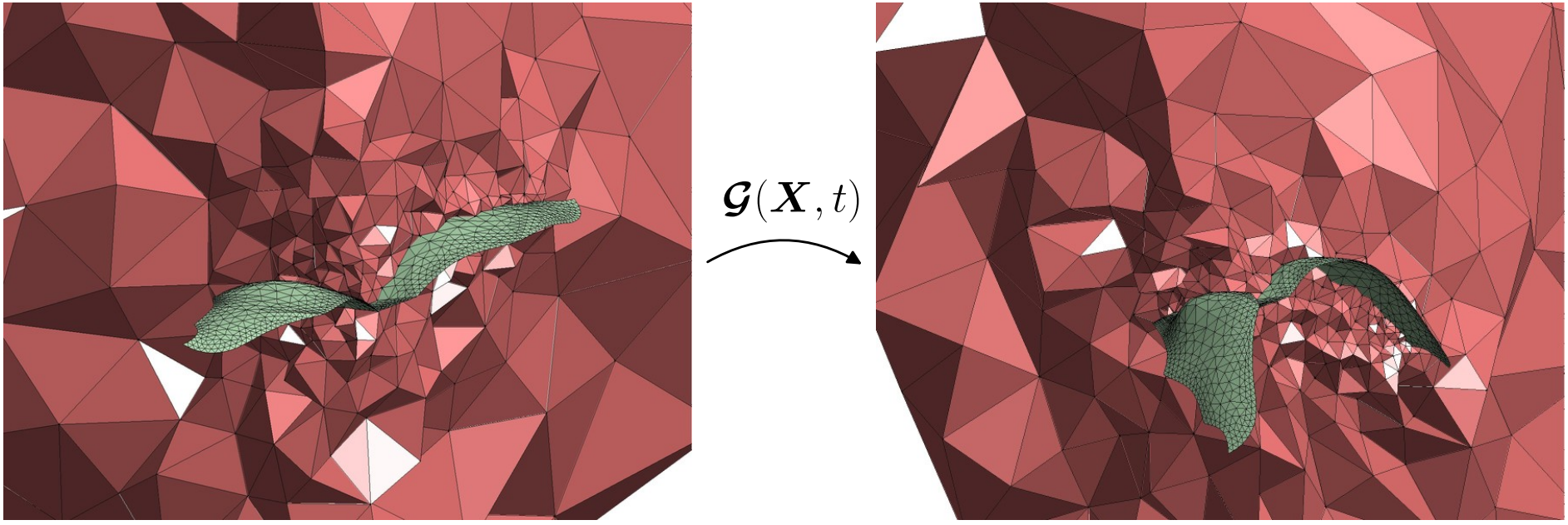






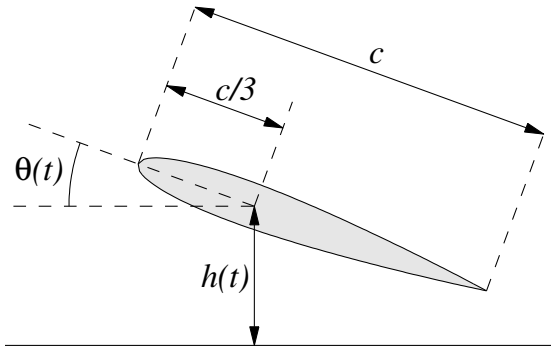
Nonlinear Elasticity for Deforming Domains

- *Non-linear solid mechanics approach*: [Persson & Peraire 2009]
  - An initial reference mesh corresponds to an undeformed solid
  - External forces come from the true moving boundary constraints
  - Solving for a force equilibrium gives the deformed (curved) boundary conforming mesh
- High-order ALE methods require a smooth mapping $\mathcal{G}(\boldsymbol{X}, t)$ such that the elements are aligned with the moving boundaries



Example: HO Workshop Moving Domain Problem

- Deforming domain problem of a NACA 0012 airfoil undergoing a flapping-type motion
- Freestream Mach = 0.2, Re = 1000
- Steady-state solution as initial condition



- Maximize energy the fluids exerts on the airfoil during the motion:

$$\underset{h(t),\theta(t)}{\text{maximize}} \quad \int_0^T \int_{\Gamma} \boldsymbol{f} \cdot \boldsymbol{v} \, dS \, dt$$

$$\text{subject to} \quad h(0) = h'(0) = h'(T) = 0, \; h(T) = 1$$

$$\theta(0) = \theta'(0) = \theta(T) = \theta'(T) = 0$$

$$\frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{F}(\boldsymbol{U}, \nabla \boldsymbol{U}) = 0$$

- $h(t)$, $\theta(t)$ discretized via *clamped cubic splines*

Fully discrete time-adjoints for DIRK-DG schemes

- Discrete adjoints for sensitivities of a time-integrated system
- Simple derivation, true derivatives (incl. discretization errors)
- Consider a diagonally implicit Runge-Kutta (DIRK) scheme:

$$\boldsymbol{u}^{(n)} = \boldsymbol{u}^{(n-1)} + \sum_{i=1}^{s} b_i \boldsymbol{k}_i^{(n)}, \quad \mathbb{M}\boldsymbol{k}_i^{(n)} = \Delta t_n \boldsymbol{r}\left(\boldsymbol{u}^{(n-1)} + \sum_{j=1}^{i} a_{ij}\boldsymbol{k}_j^{(n)},\ \boldsymbol{\mu},\ t_{n-1} + c_i \Delta t_n\right)$$

- We form the fully-discrete, time-dependent PDE-constrained optimization problem:

$$\underset{\boldsymbol{u}^{(i)} \in \mathbb{R}^{N_{\boldsymbol{u}}}, \boldsymbol{k}_1^{(1)} \in \mathbb{R}^{N_{\boldsymbol{u}}}, \boldsymbol{\mu} \in \mathbb{R}^{N_{\boldsymbol{\mu}}}}{\text{minimize}} \quad J(\boldsymbol{u}^{(0)}, \ldots, \boldsymbol{u}^{(N_t)}, \boldsymbol{k}_1^{(1)}, \ldots, \boldsymbol{k}_s^{(N_t)}, \boldsymbol{\mu})$$

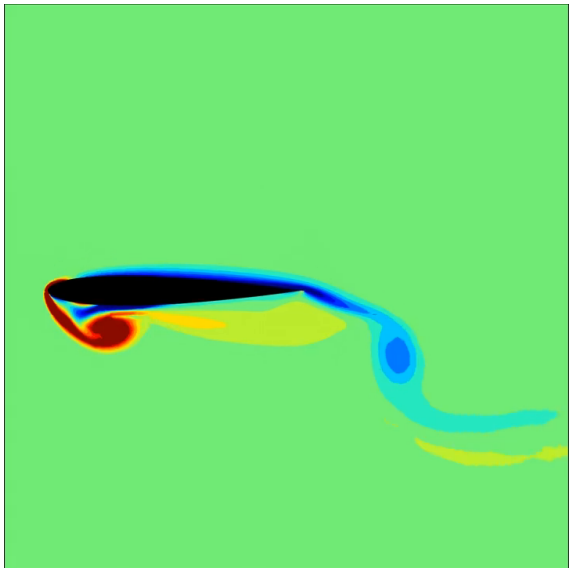
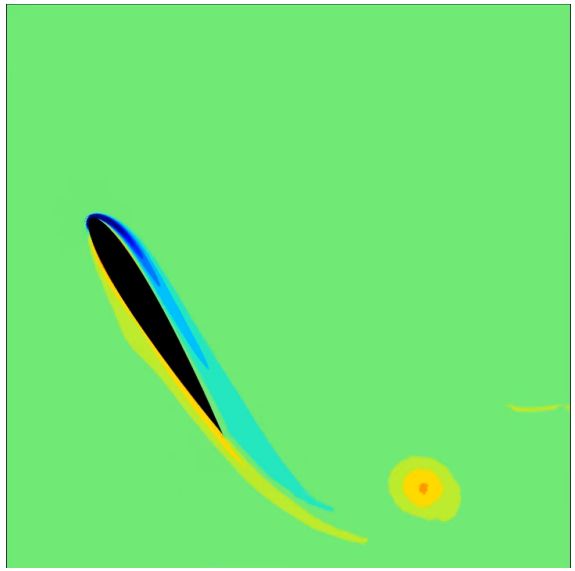
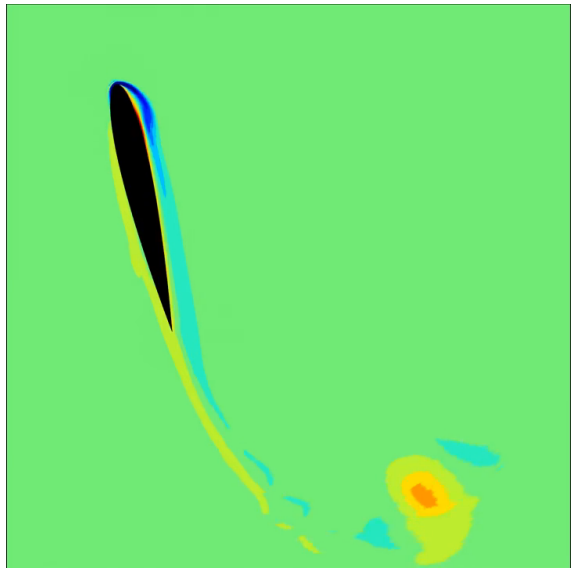
$$\text{subject to} \quad \boldsymbol{u}^{(0)} = \boldsymbol{u}_0(\boldsymbol{\mu}), \quad \boldsymbol{u}^{(n)} = \boldsymbol{u}^{(n-1)} + \sum_{i=1}^{s} b_i \boldsymbol{k}_i^{(n)}$$

$$\mathbb{M}\boldsymbol{k}_i^{(n)} = \Delta t_n \boldsymbol{r}\left(\boldsymbol{u}^{(n-1)} + \sum_{j=1}^{i} a_{ij}\boldsymbol{k}_j^{(n)},\ \boldsymbol{\mu},\ t_{n-1} + c_i \Delta t_n\right)$$

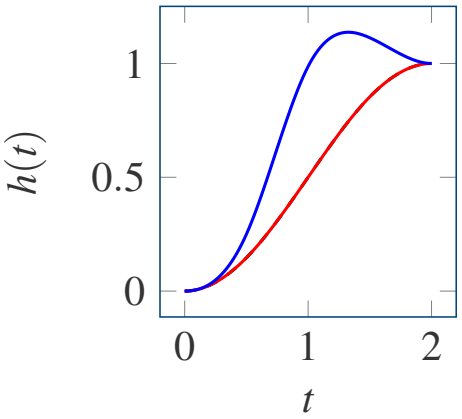
Fully discrete time-adjoints for DIRK-DG schemes

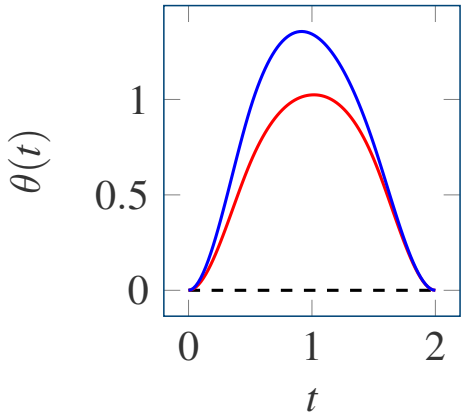
- The corresponding first-order optimality conditions give the fully-discrete adjoint evolution equations
- These have the natural form of an evolution *backwards* in time, and the stages are solved in reverse order
- Using the primal and dual solutions, we can obtain the gradient $dJ/d\boldsymbol{\mu}$ of the discretized output functional with respect to the parameters using a straight-forward application of the chain rule
- Also apply for gradient of general PDE-dependent constraints
- Optimize using the L-BFGS-B algorithm

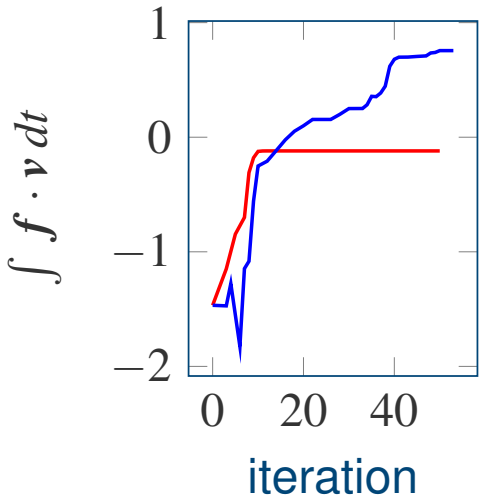
Optimization Results

- Initial guess (- - -): $h_0(t) = (1 - \cos(\pi t/T))/2$, $\theta_0(t) = 0$
- Optimization 1 (——): $h_0(t) = (1 - \cos(\pi t/T))/2$, $\theta(t)$ parametrized
- Optimization 2 (——): $h(t)$, $\theta(t)$ parametrized







Partitioned FSI using IMEX schemes

- [Froehle & Persson 2013, 2014]
- IMEX schemes can be used to derive accurate partitioning methods for fully coupled FSI problems
- Treat a *predicted* traction $\tilde{t}$ explicitly and everything else implicitly:

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

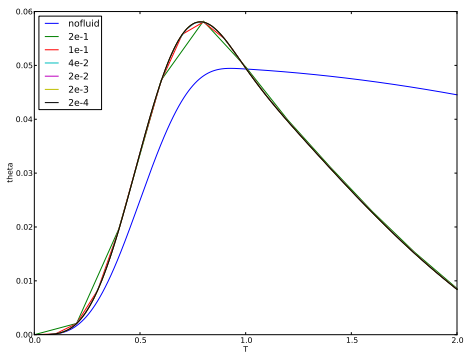
- Use Runge-Kutta based predictor [Van Zuijlen & Bijl 2005]

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

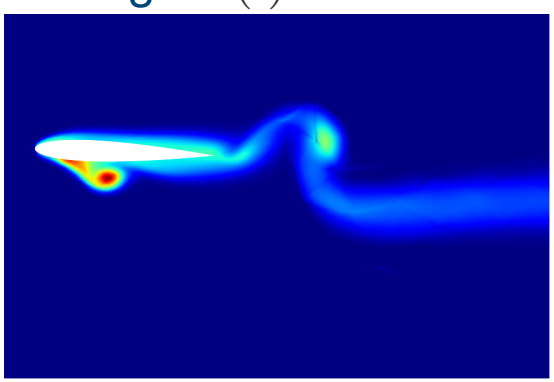
- The remaining structure and fluid components obtained by back-solving of the block upper-triangular system
- Consistent forces, no subiterations required

Verification: Benchmark Pitching Airfoil System

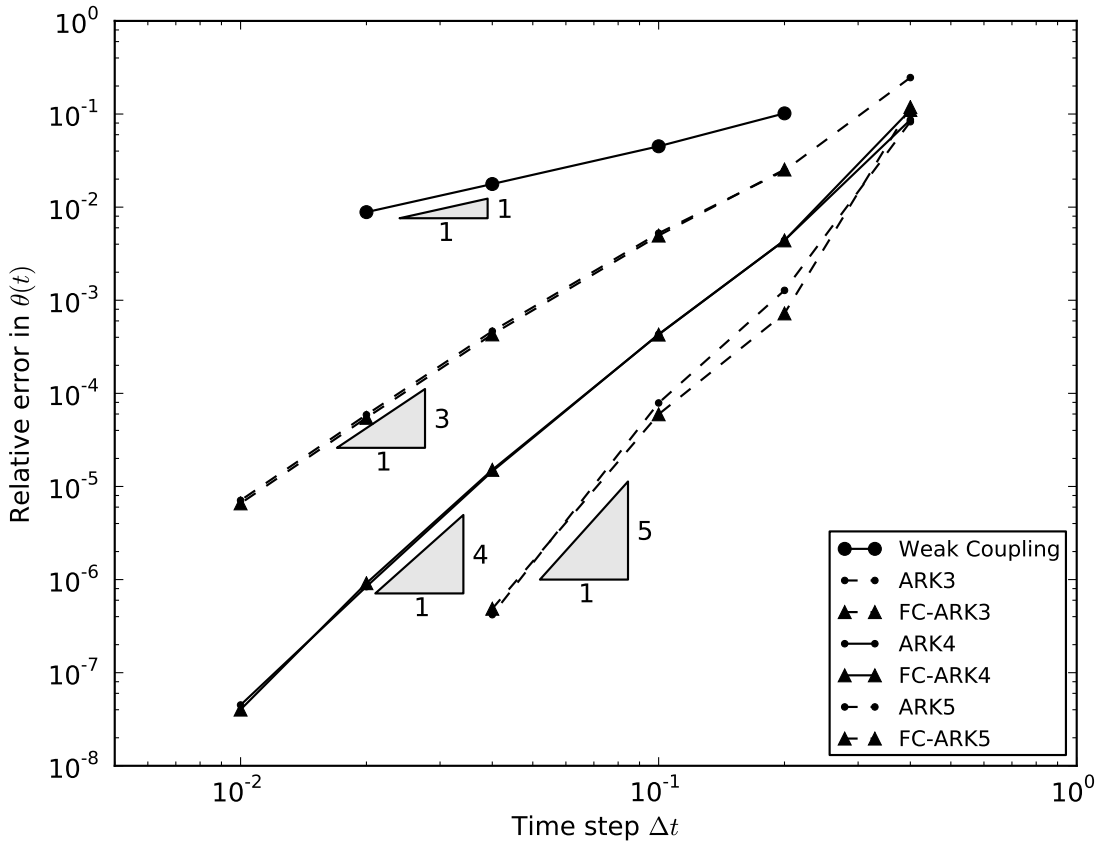
- Simple FSI benchmark problem for studying the high-order accuracy of the IMEX scheme
- Rigid pitching/heaving NACA 0012 airfoil, torsional spring
- Up to 5th order of accuracy, similar to solving fully coupled system



Angle $\theta(t)$ vs time t

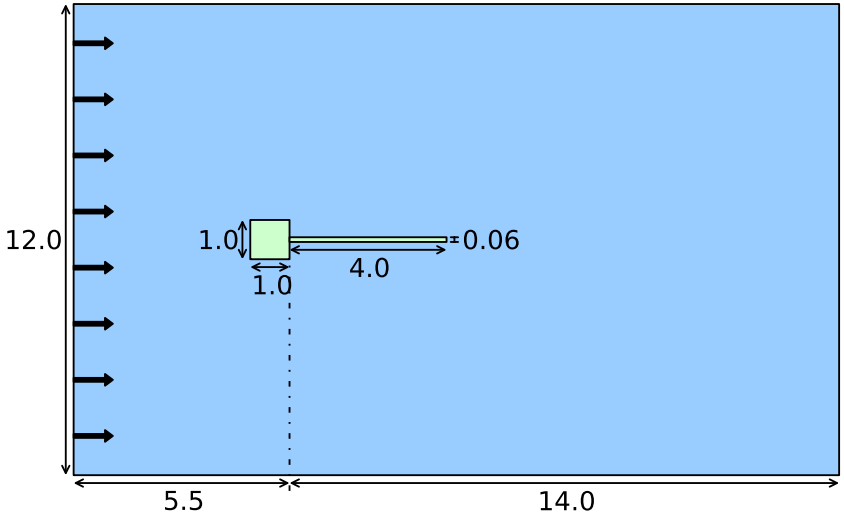


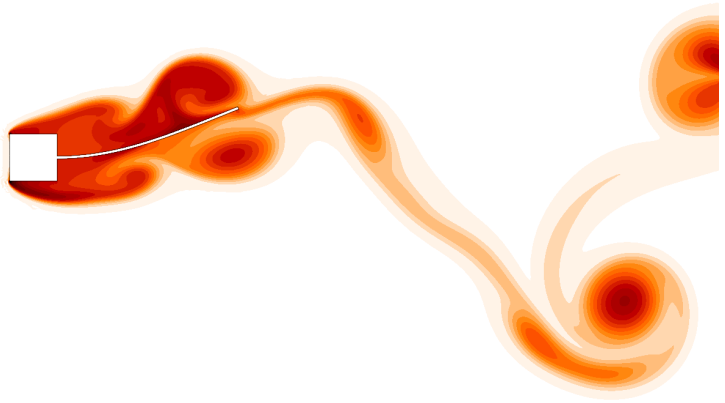
Entropy



Verification: Cantilever System

- Standard FSI benchmark problem. [Wall & Ramm 1998]
- Flow around elastic cantilever behind a square bluff body

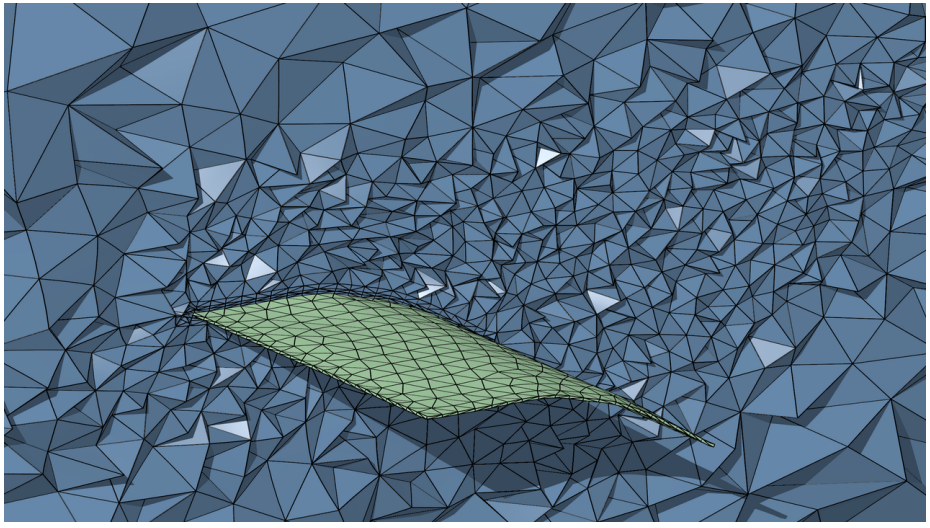




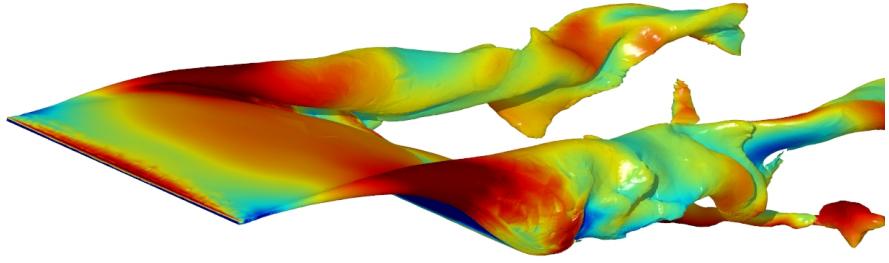
- Tip frequency: $f = 3.14\,\text{Hz}$ (Literature: $2.98 - 3.25\,\text{Hz}$)
- Tip displacement: $d_{max} = 1.09\,\text{cm}$ (Literature: $0.95 - 1.25\,\text{cm}$)

Flow around Membrane, 3-D

- Angle of attack $22.6°$, Reynolds number 2000.
- Flexible structure reduces leading edge separation.
- Fluid: 108k degree 3 tetrahedra (11M DOF)
- Solid: 1k degree 3 tetrahedra



Mesh

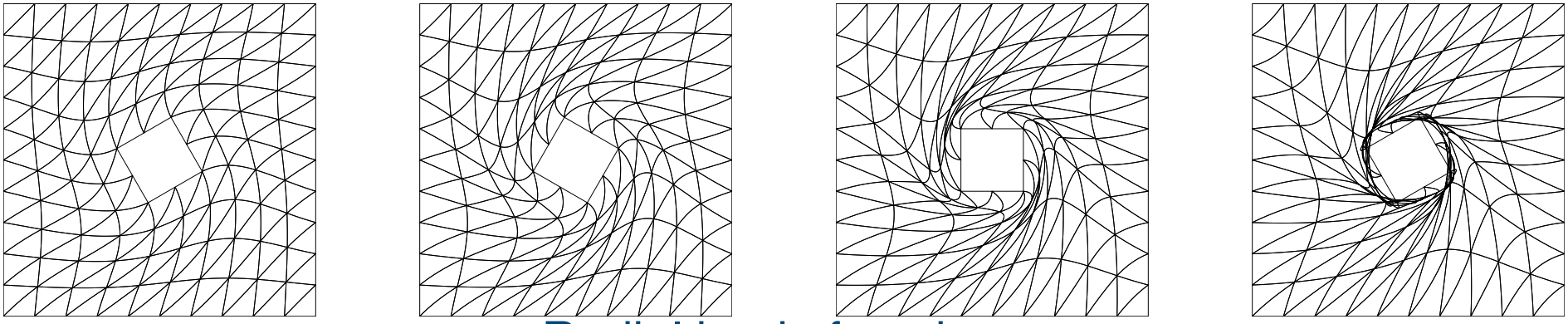


Flow field

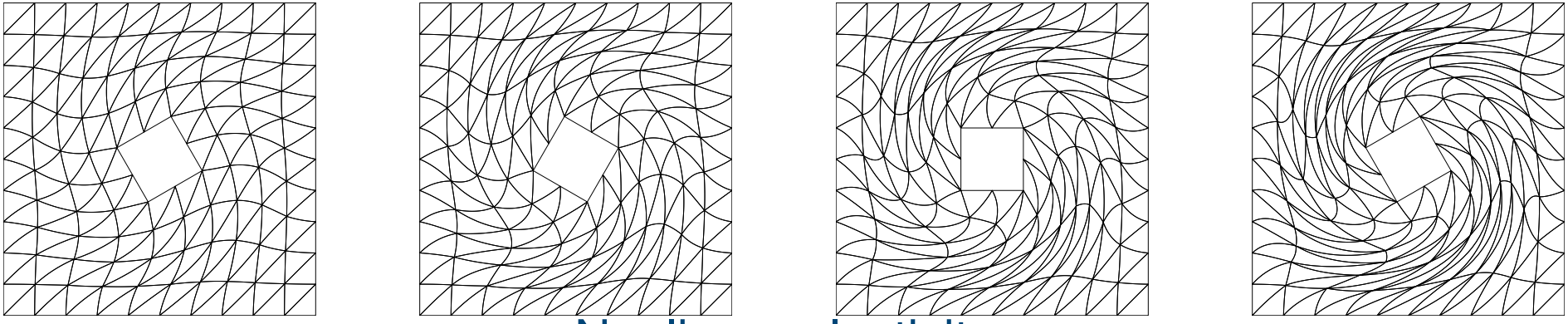
Outline

Domains with Large Deformations

- For large deformations, it is in general not possible to deform the meshes smoothly – *remeshing required*
- For efficient numerical schemes, use *local* mesh operations



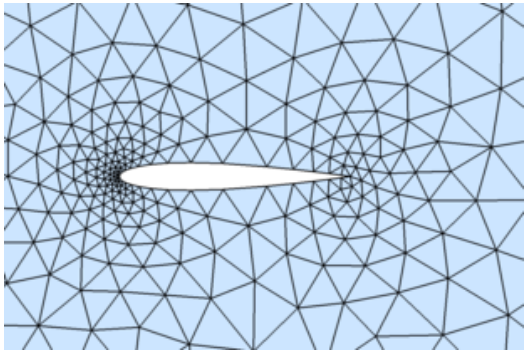
Radial basis functions

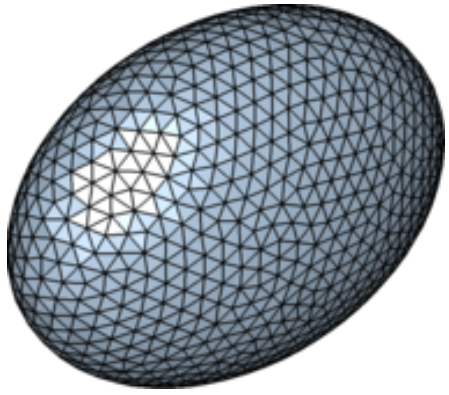


Nonlinear elasticity

The DistMesh Mesh Generator

- High quality meshes obtained using the *DistMesh* algorithm
  [Persson, Ph.D. thesis, '05]
  1. Start with *any* topologically correct initial mesh
  2. Move nodes to find force equilibrium in edges
     - Project boundary nodes using *implicit geometry* $\phi(\boldsymbol{x})$
     - Update element connectivities with Delaunay
- Excellent properties:
  - Very simple (1 page of MATLAB)
  - Implicit geometries $\rightarrow$ No CAD required
  - Very high element qualities
  - Moving meshes/deforming domains
- Widely used:
  - Numerous books and courses
  - Rewritten in C, C++, C#, Fortran 77/90, Python, Mathematica, Octave



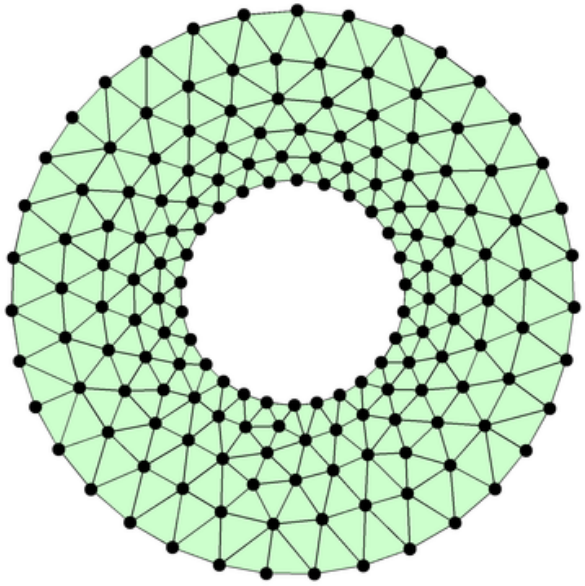


The DistMesh Mesh Generator

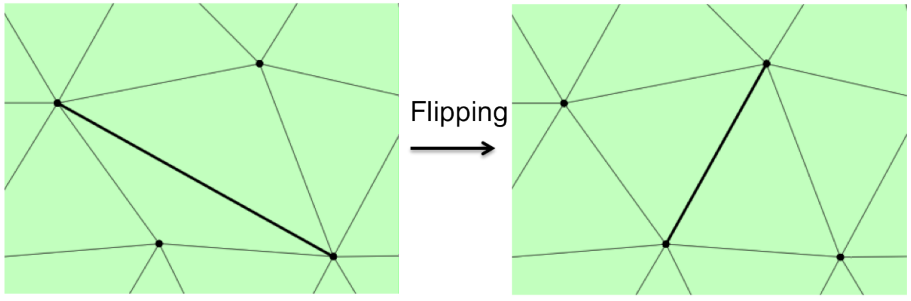
- Spring-based non-linear compressive force analogy for mesh motion

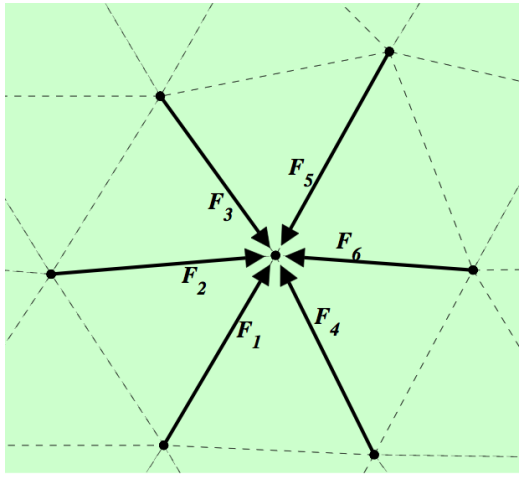
$$\boldsymbol{p}^{(n+1)} = \boldsymbol{p}^{(n)} + \delta \sum_i \boldsymbol{F_i}$$

$$|\boldsymbol{F}_i(l)| = \begin{cases} k(l - l_0) & \text{if } l \geq l_0, \\ 0 & \text{if } l < l_0, \end{cases}$$

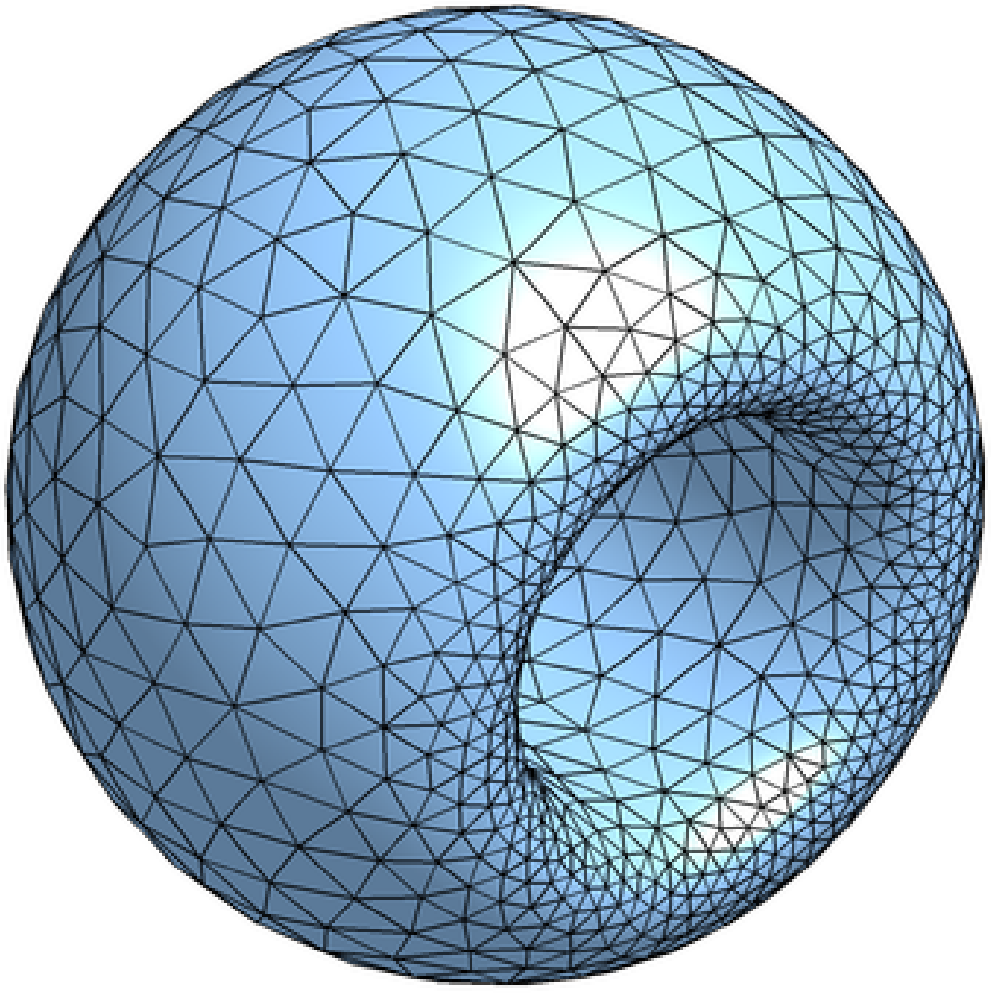


- Perform topological transformations ("edge flips") to improve element connectivities



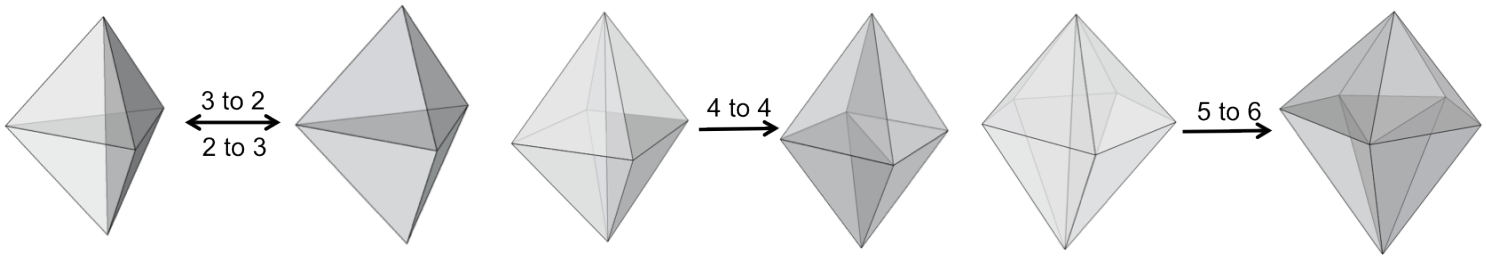


The DistMesh Mesh Generator on Surfaces

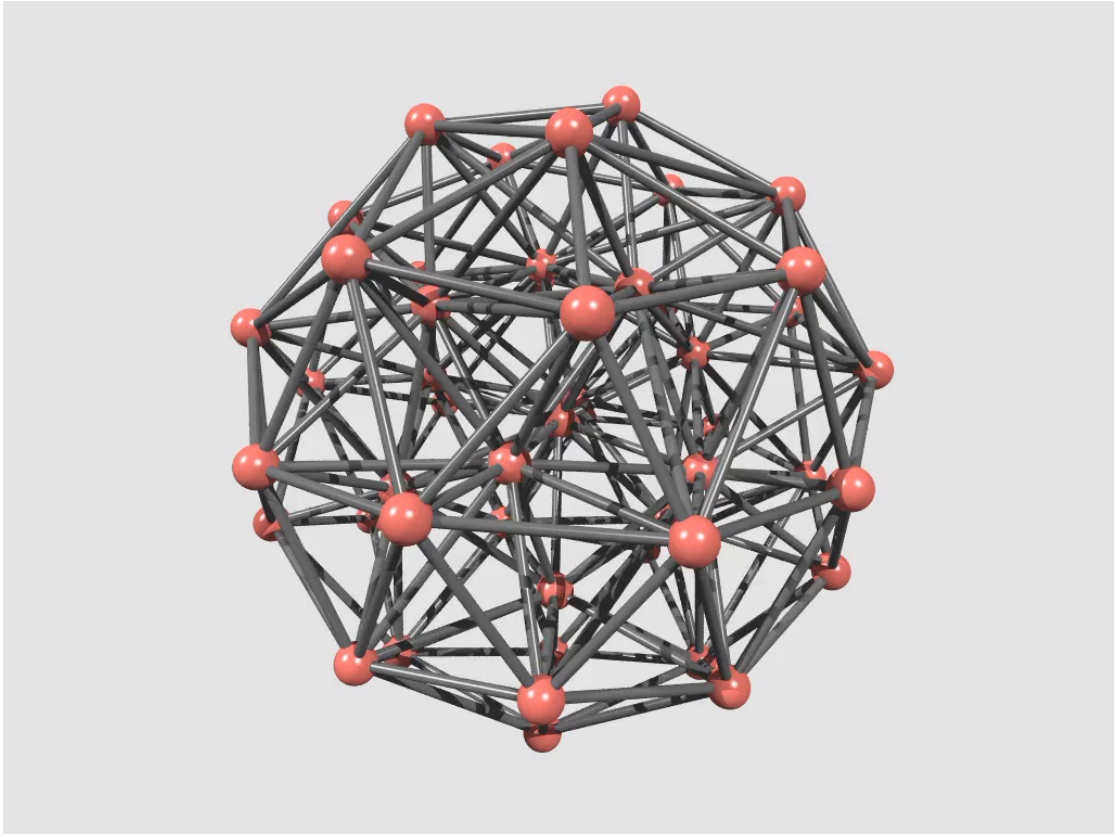


Element flips and DistMesh in 3D

- Local element flips for 3D tetrahedra:

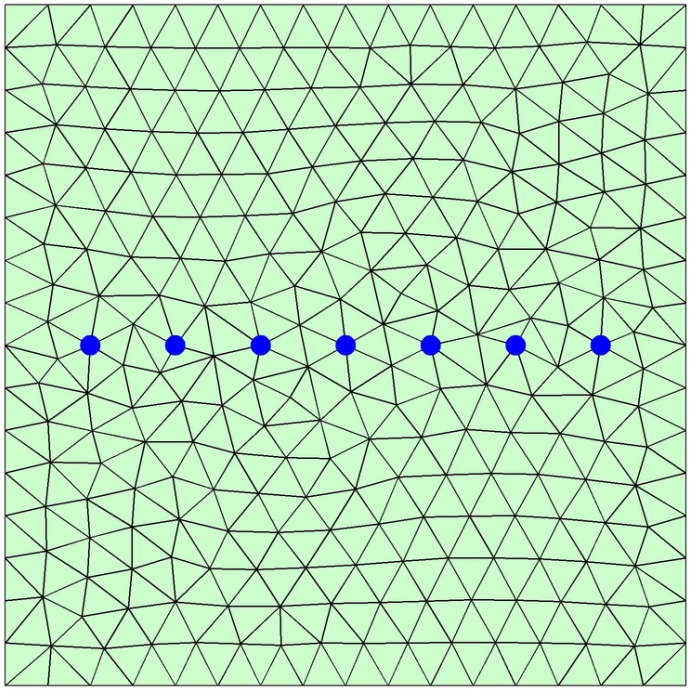


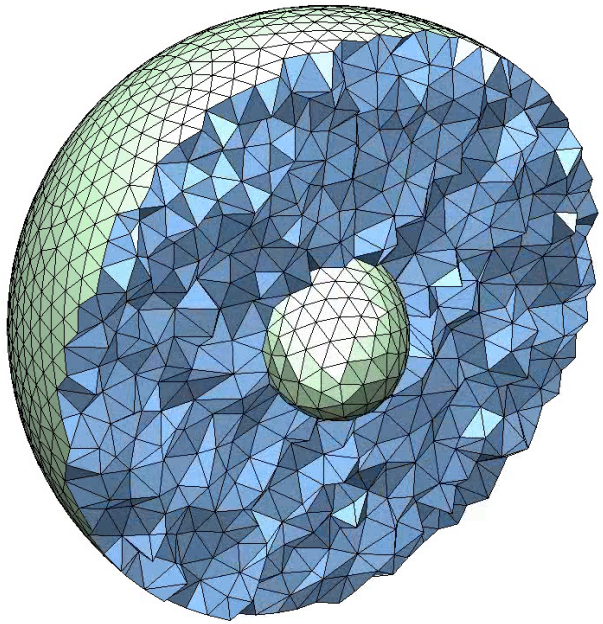
- Restricts the topology changes to a small number of elements



Moving Meshes

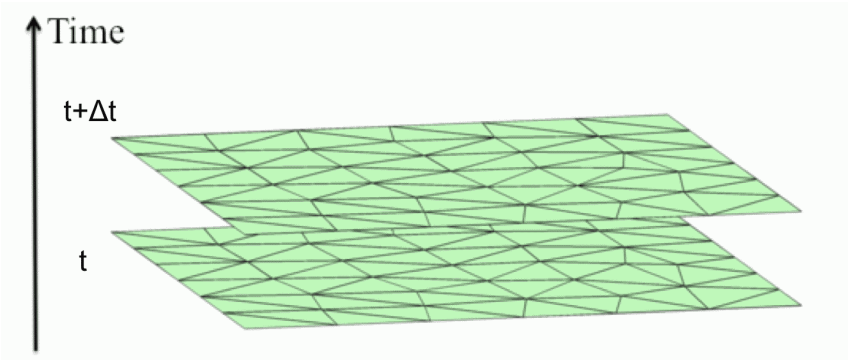
- In addition to generating high-quality initial meshes, the DistMesh algorithm is excellent for iterative generation of moving meshes
- The resulting mesh sequence involves two types of operations:
  1. Smooth node movements
  2. *Localized* element topology updates
- This allows for integration with efficient numerical schemes

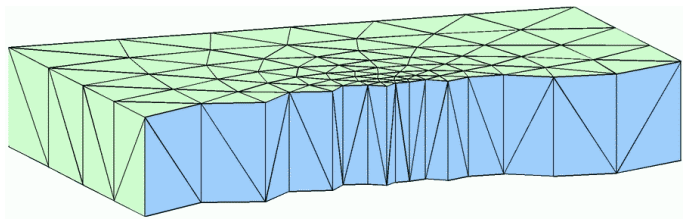




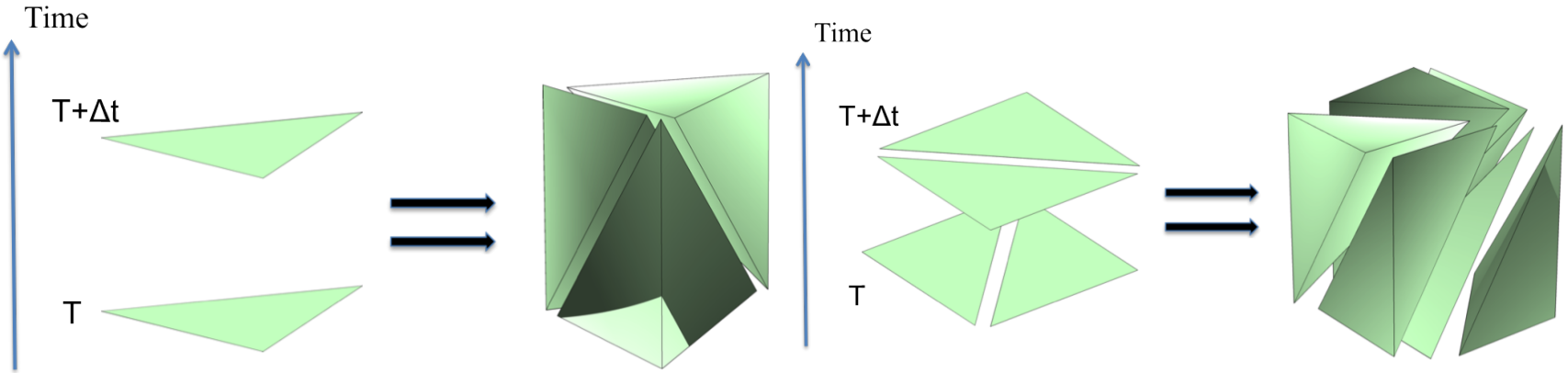
Space-time Mesh Generation

- Local mesh operations significantly simplify the process of space-time slab mesh generation



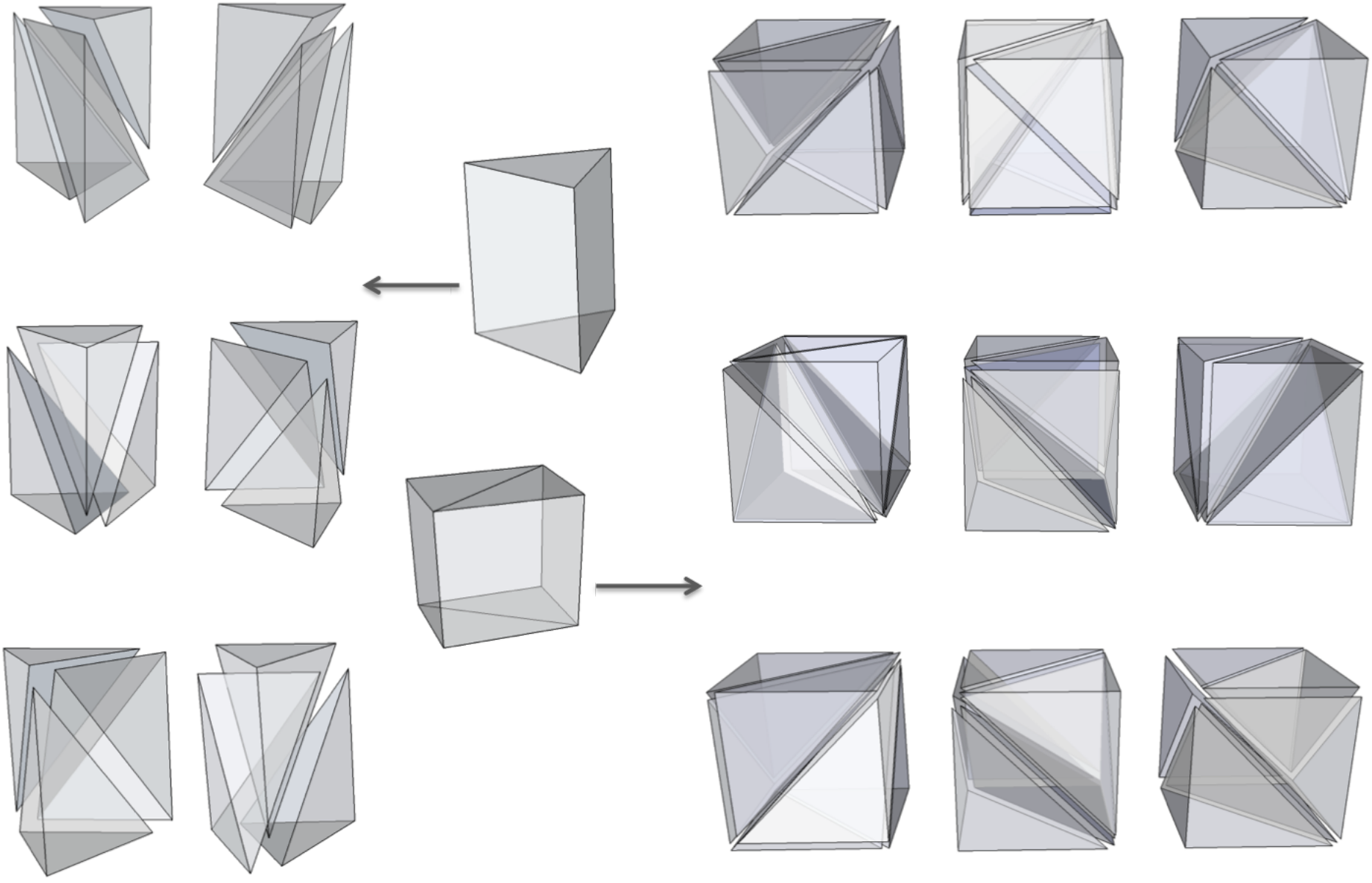


- Each local prism triangulation depends on the choice of the diagonals on the lateral faces



- A depth-first algorithm for the global assignment of diagonals

Local Triangulation of Prisms

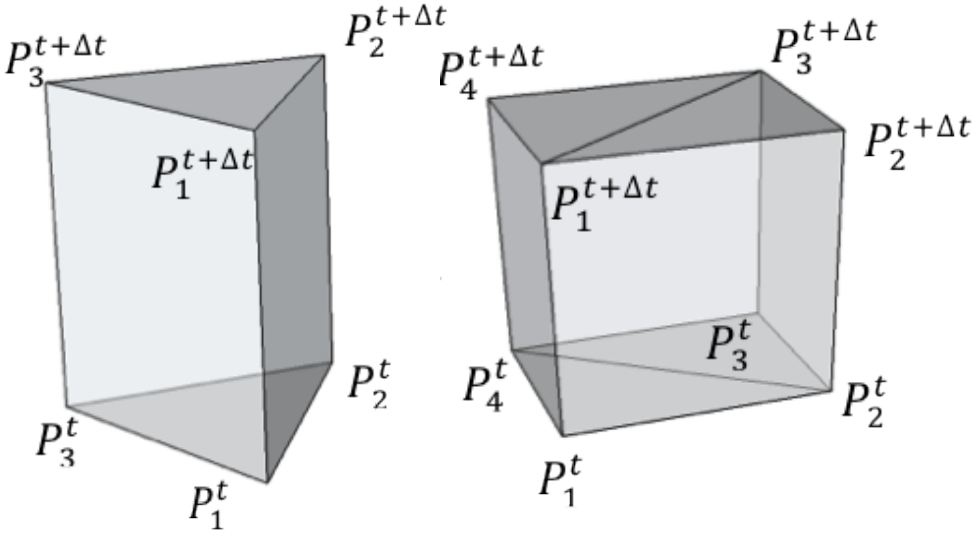


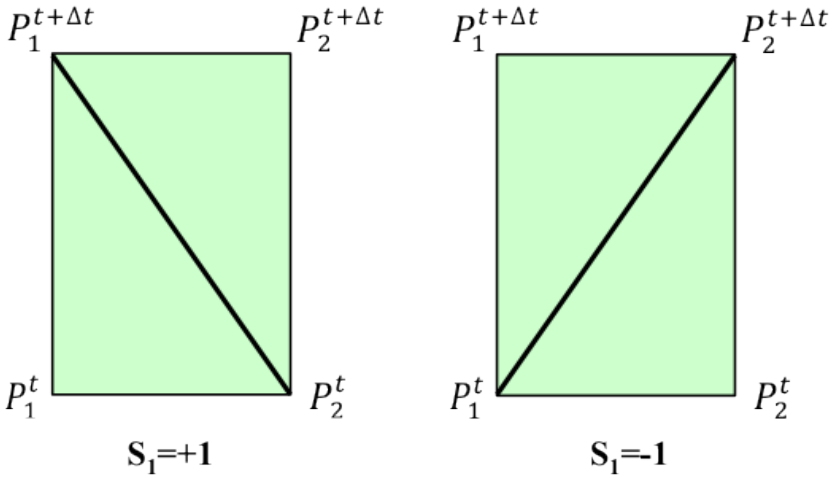
Local Triangulation of Prisms

- Index the vertices of each prism
- Define a sign function for each lateral face as

$$S_i(p_i^t, p_i^{t+\Delta t}, p_j^t, p_j^{t+\Delta t}) = \begin{cases} -1 & \text{if the diagonal edge is } \overline{p_i^t p_j^{t+\Delta t}} \\ +1 & \text{if the diagonal edge is } \overline{p_i^{t+\Delta t} p_j^t} \end{cases}, \quad 1 \le i \le n$$

where $j = (i \mod n) + 1$ and n is the total number of lateral faces

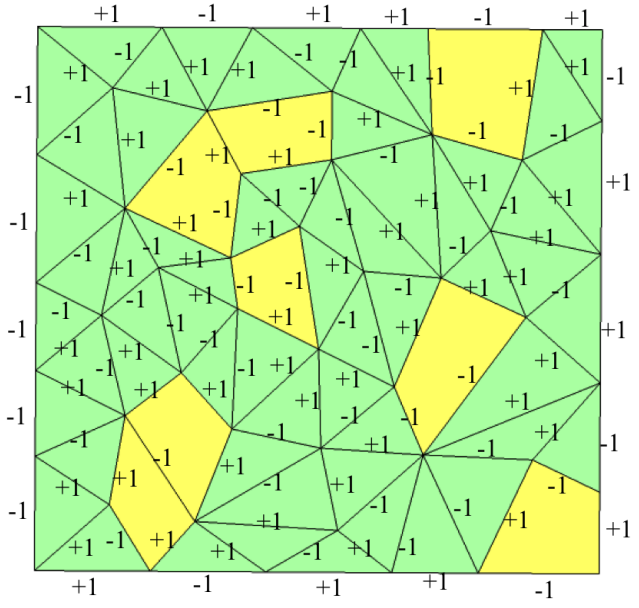




Global Space-Time Mesh Generation

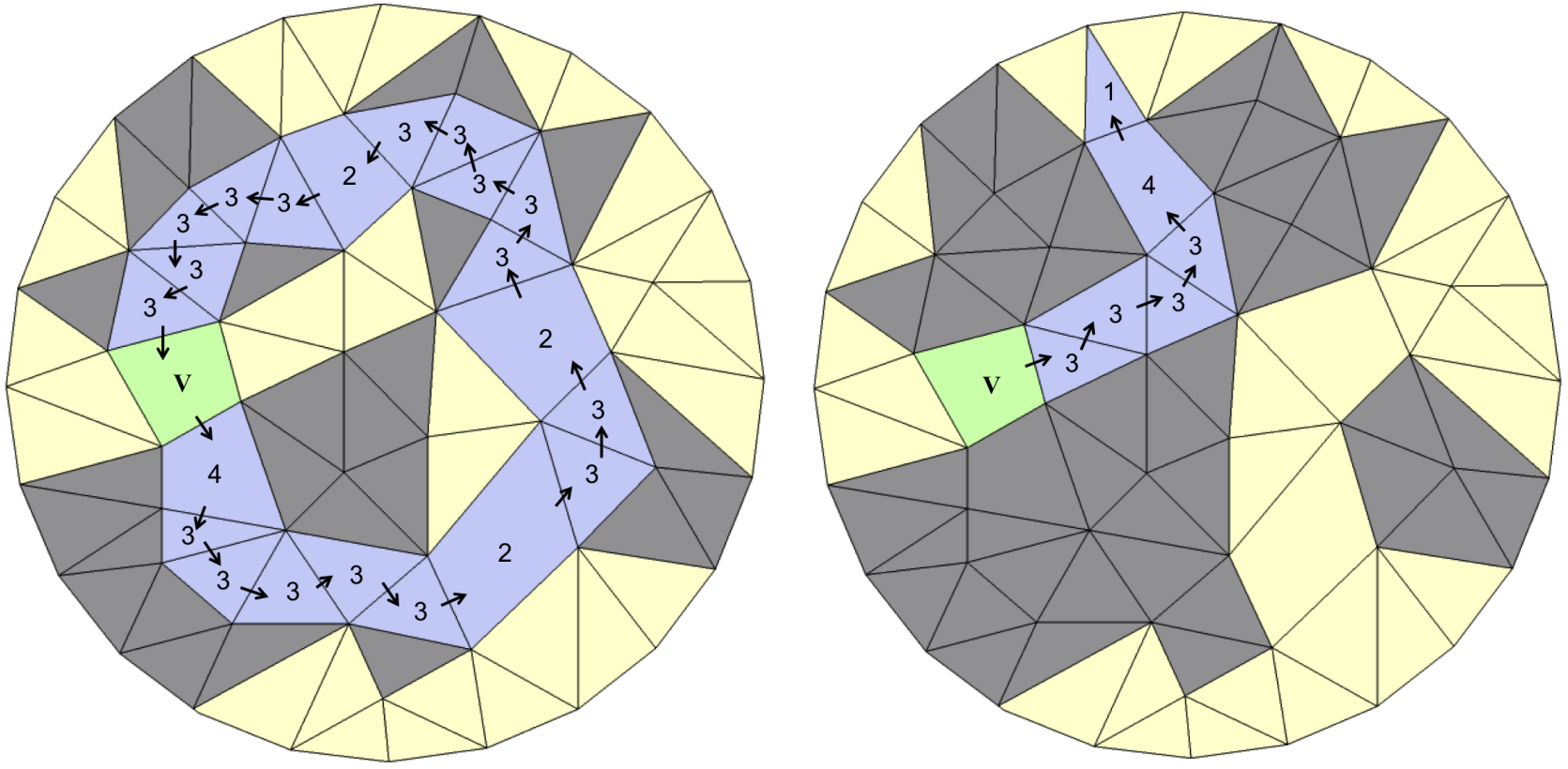
- Generate a global Space-Time unstructured mesh by assigning one diagonal for each lateral face of each prism:
  - For each prism, the diagonals of lateral faces can give a valid triangulation.
  - Each prism should match the diagonals of shared lateral faces with their neighbor prisms.

- Equivalently, consider a 2D mesh with triangles and quadrilaterals. Assign -1 or $+1$ to each edge:
- We use a depth-first search algorithm for sign assignment.



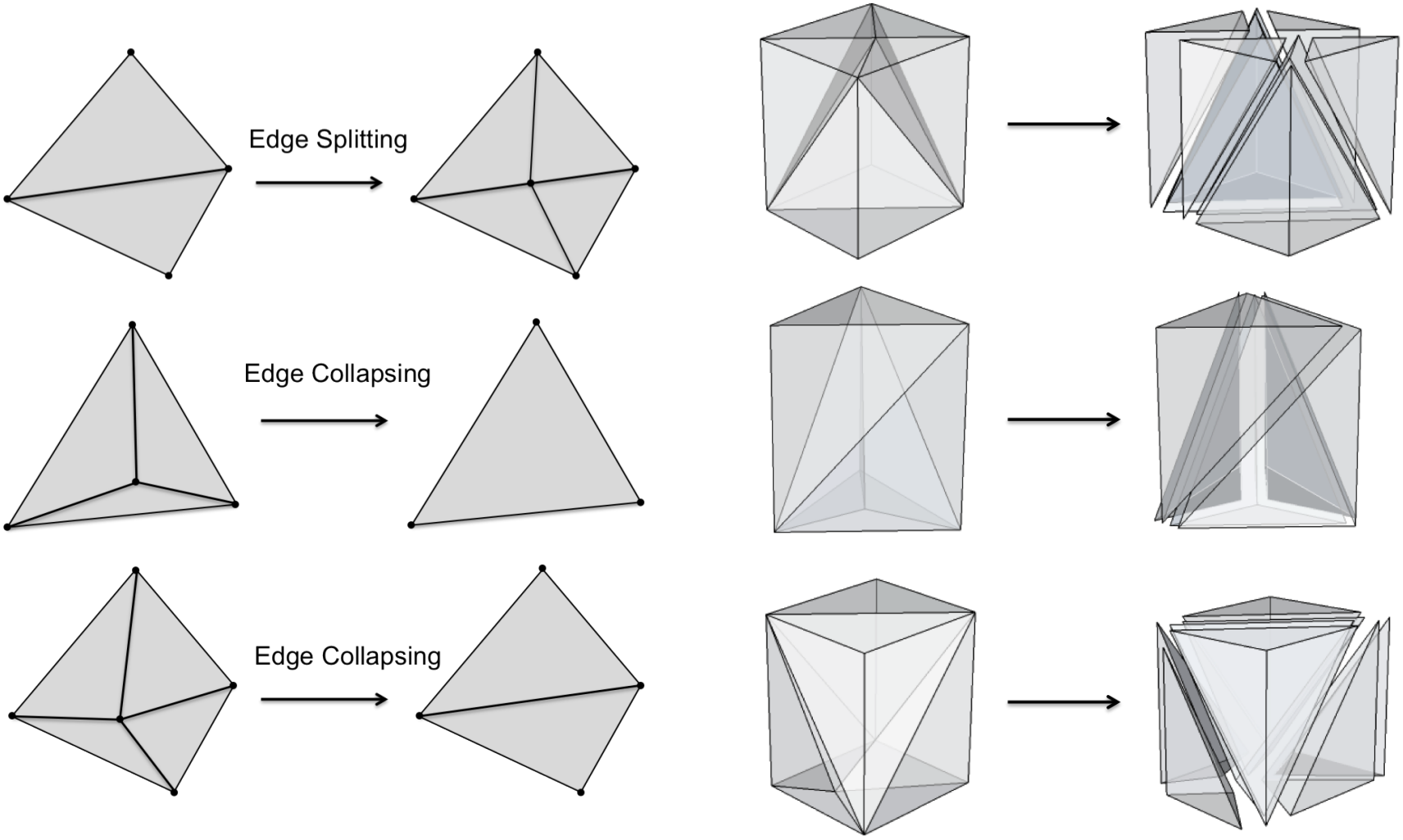
Examples of Diagonal Matching

- Efficient algorithm for generation of globally consistent space-time mesh (see paper for details)



Edge collapsing and Edge splitting

Two more local mesh operations for adding and removing nodes:



Space-Time Discontinuous Galerkin Formulation

- Fully unstructured space-time DG method:
  - Fully consistent discretization in both space and time
  - Allows for arbitrary mesh deformations and topology changes
- Define the broken DG spaces $\mathcal{V}_T^h$ and Σ_T^h associated with a triangulation $\mathcal{T}_{[0,T]}^h = \{K\}$ of the space-time domain $\Omega[0,T]$ as:

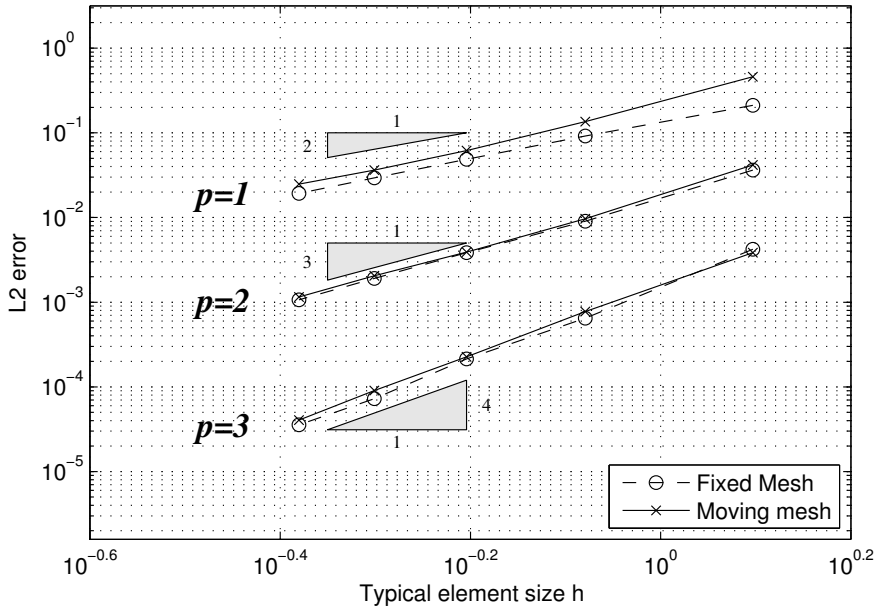
$$\mathcal{V}_T^h = \{\boldsymbol{v} \in [L^2(\Omega[0,T])]^5 \mid \boldsymbol{v}|_K \in [\mathcal{P}_p(K)]^5 \ \ \forall K \in \mathcal{T}_{[0,T]}^h\},$$
$$\Sigma_T^h = \{\boldsymbol{\sigma} \in [L^2(\Omega[0,T])]^{5\times 3} \mid \boldsymbol{\sigma}|_K \in [\mathcal{P}_p(K)]^{5\times 3} \ \ \forall K \in \mathcal{T}_{[0.T]}^h\},$$

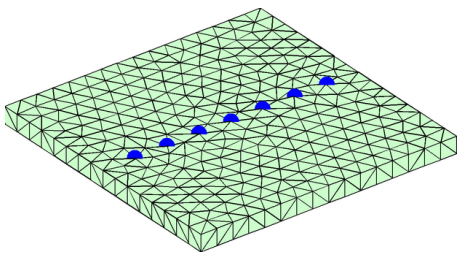
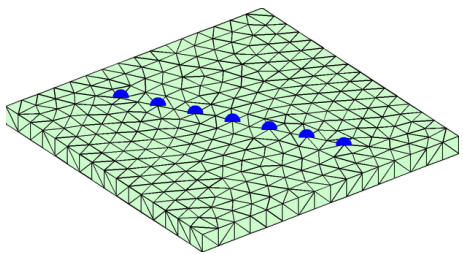
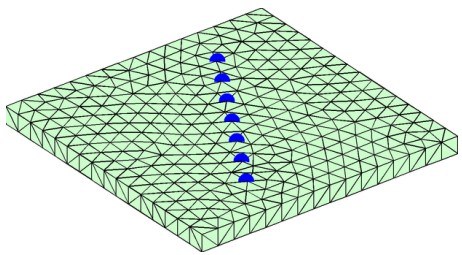
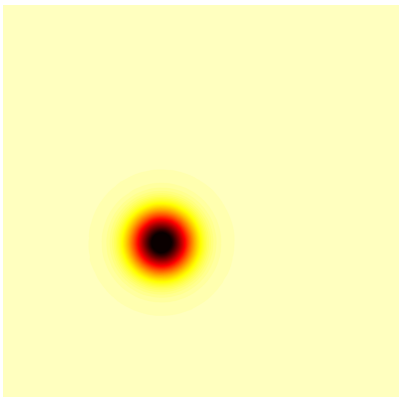
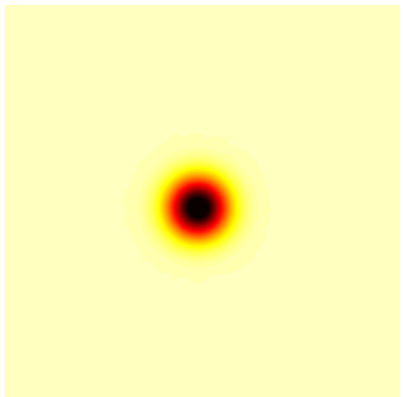
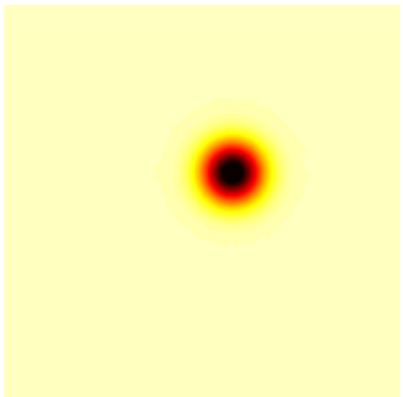
- Discretize the first-order system using a standard DG formulation on the space-time domain $\Omega[t_1, t_2]$.

$$-\int_K \tilde{\boldsymbol{F}}^{\text{inv}}(\boldsymbol{u}^h) : \nabla_{XT}\boldsymbol{v}^h \, dx + \oint_{\partial K} \widehat{(\tilde{\boldsymbol{F}}^{\text{inv}} \cdot \boldsymbol{n})} \cdot \boldsymbol{v}^h \, ds$$
$$= -\int_K \boldsymbol{F}^{\text{vis}}(\boldsymbol{u}^h, \boldsymbol{q}^h) : \nabla_X \boldsymbol{v}^h \, dx + \oint_{\partial K} \widehat{(\boldsymbol{F}^{\text{vis}} \cdot \boldsymbol{n}_s)} \cdot \boldsymbol{v}^h \, ds, \qquad \forall \boldsymbol{v}^h \in \mathcal{V}_T^h$$
$$\int_K \boldsymbol{q}^h : \boldsymbol{\sigma}^h \, dx = -\int_K \boldsymbol{u}^h \cdot (\nabla_X \cdot \boldsymbol{\sigma}^h) \, dx + \oint_{\partial K} (\widehat{\boldsymbol{u}^h} \otimes \boldsymbol{n}_s) : \boldsymbol{\sigma}^h \, ds, \qquad \forall \boldsymbol{\sigma}^h \in \Sigma_T^h.$$

Example: Euler Vortex, Convergence

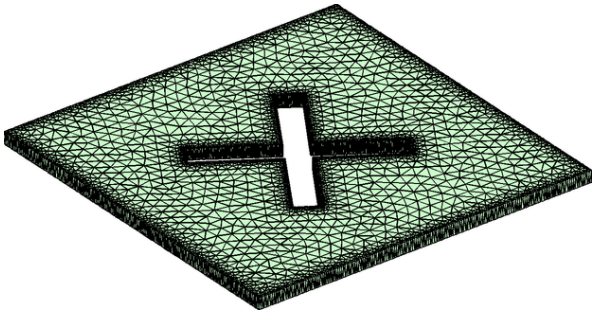
- Propagate an Euler Vortex on a fixed domain but moving mesh
- Optimal order of convergence $O(h^{p+1})$ for fixed and moving mesh.

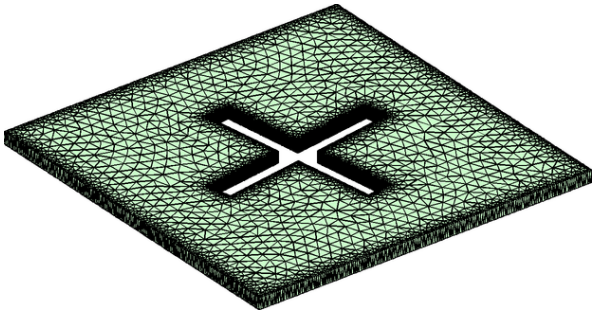


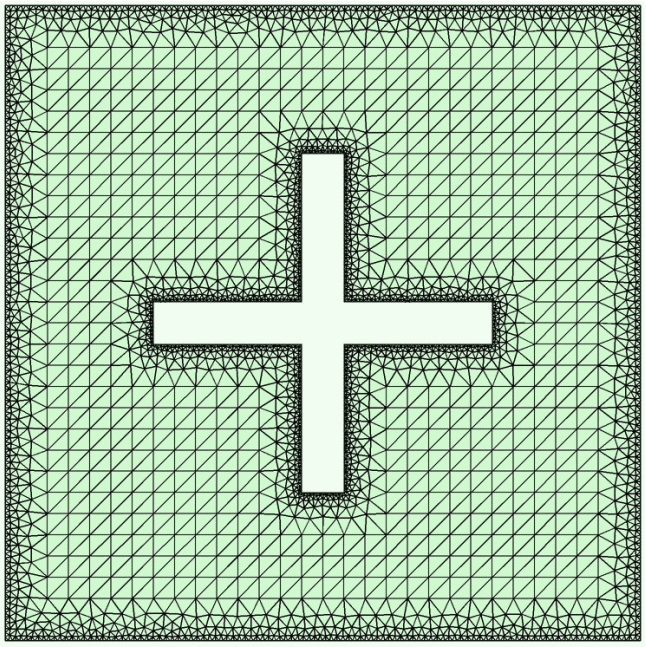







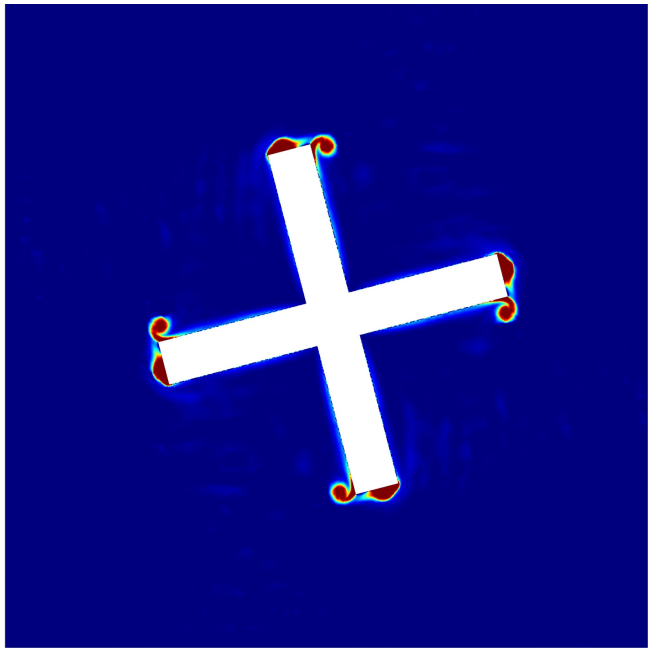
Example: Spinning Cross

- Flow around a spinning cross with $\omega = 1$, Reynolds number 3000, Mach 0.2, polynomial degree p=2.
- Graded mesh around cross moves rigidly with geometry movement
- Mesh improvement techniques applied to the remaining elements



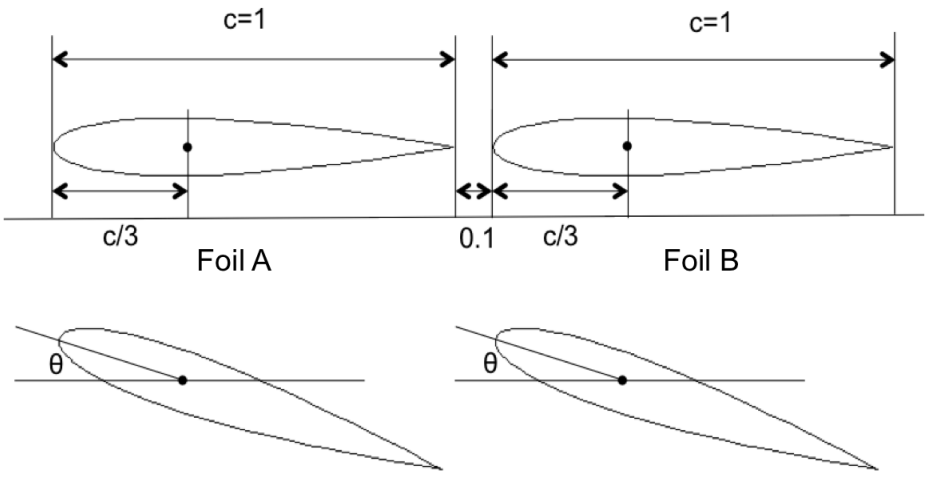


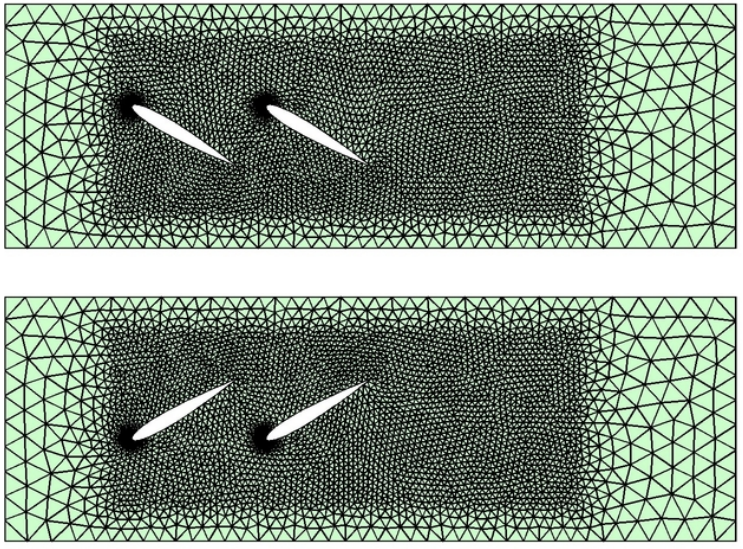




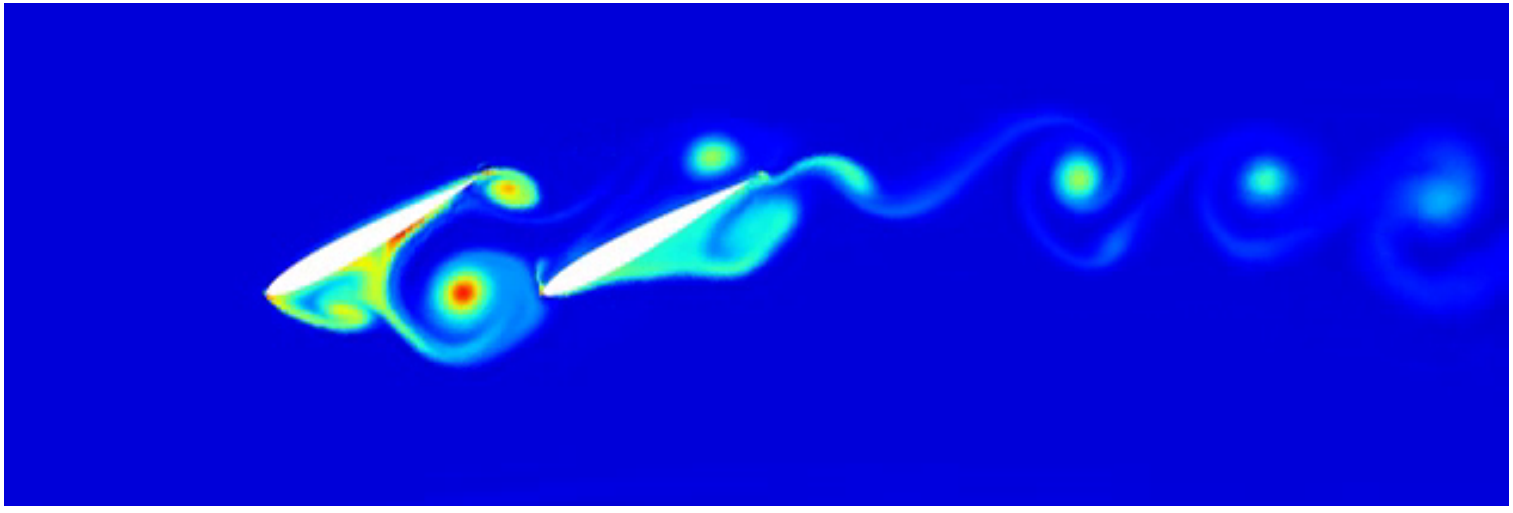
Example: Tandem Foils

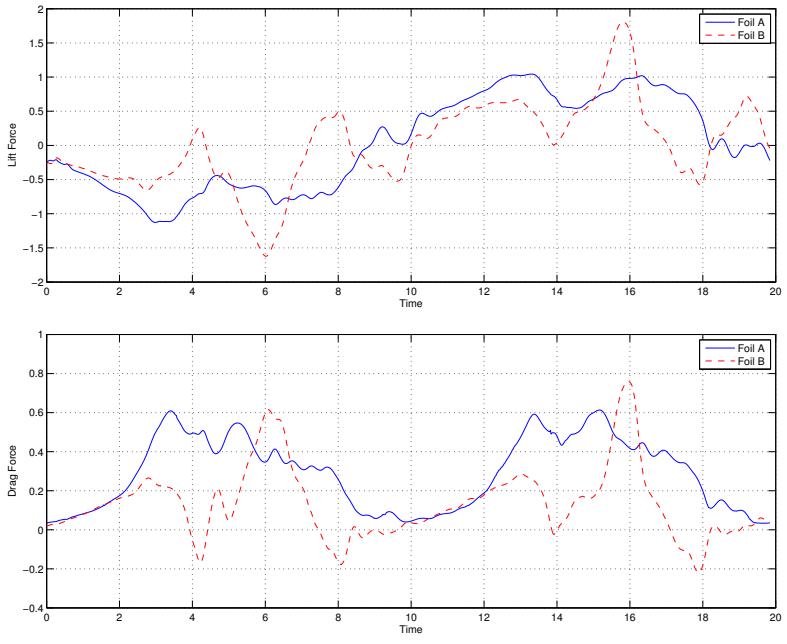
- Two foils are placed very close and rotated based on $\theta = A\sin(-2\pi ft)$ with $A = \pi/6$ and $f = 0.05$. Reynolds number 3000, Mach 0.2, polynomial degree p=2.





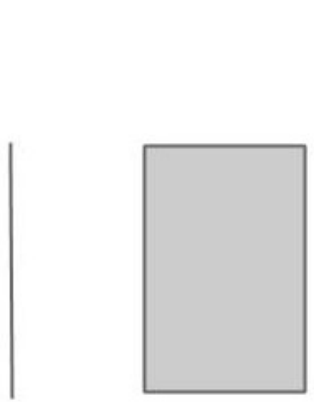
Example: Tandem Foils



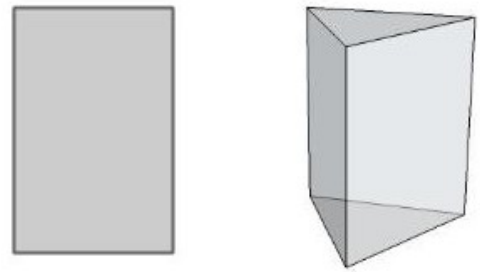


Fully unstructured 4D space-time mesh generation

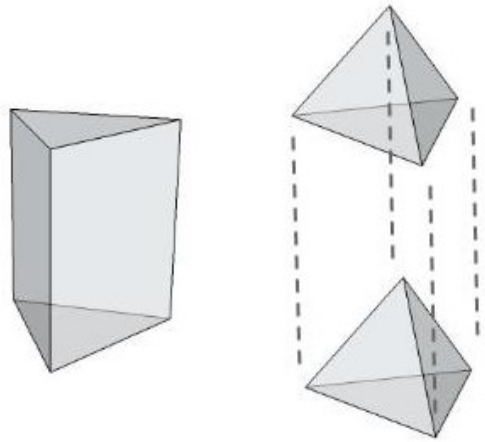
- 4D is difficult to visualize, and we extend concepts such as ‘Prisms’ and ‘Lateral Faces’ using combinatorial notions



2D ‘Prism’



3D Prism



4D ‘Prism’

	2D Prism	3D Prism	4D Prism
Geometry of Bottom/Top Face	Line Segment	Triangle	Tetrahedra
Geometry of Lateral Faces	Line Segment	Rectangle	Triangular prism
Number of Lateral Faces	2	3	4
Number of Diagonals	$\binom{2}{2}$	$\binom{3}{2}$	$\binom{4}{2}$

Fully unstructured 4D space-time mesh generation

- The following result can be shown for obtaining valid simplex triangulations of a simple 4D prism:

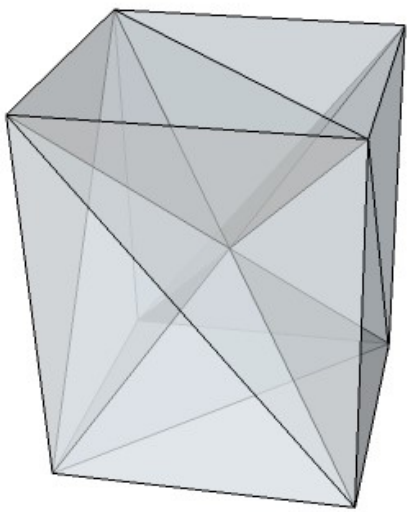
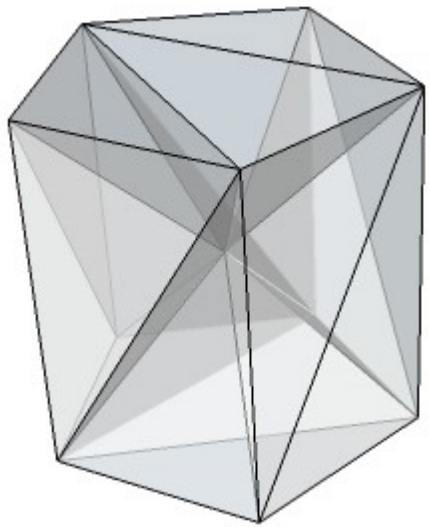
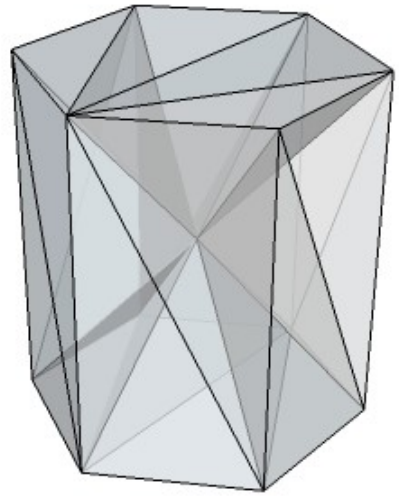
Theorem

If the 4D prism mesh is constructed purely by simple 4D-prisms, the indexing approach can always triangulate the prism mesh into a valid simplex mesh. More precisely, in each simple 4D-prism, if we have the ordered vertices $\{p_{(1)}^{V,t}, p_{(2)}^{V,t}, p_{(3)}^{V,t}, p_{(4)}^{V,t}\}$ *with* $I_{(1)} < I_{(2)} < I_{(3)} < I_{(4)}$*, the simple 4D-prism is triangulated by the following four 4D-simplices with vertex sets*

$$
\begin{aligned}
T_1 &= \{p_{(1)}^{V,t}, p_{(1)}^{V,t+\Delta t}, p_{(2)}^{V,t+\Delta t}, p_{(3)}^{V,t+\Delta t}, p_{(4)}^{V,t+\Delta t}\},\\
T_2 &= \{p_{(1)}^{V,t}, p_{(2)}^{V,t}, p_{(2)}^{V,t+\Delta t}, p_{(3)}^{V,t+\Delta t}, p_{(4)}^{V,t+\Delta t}\},\\
T_3 &= \{p_{(1)}^{V,t}, p_{(2)}^{V,t}, p_{(3)}^{V,t}, p_{(3)}^{V,t+\Delta t}, p_{(4)}^{V,t+\Delta t}\},\\
T_4 &= \{p_{(1)}^{V,t}, p_{(2)}^{V,t}, p_{(3)}^{V,t}, p_{(4)}^{V,t}, p_{(4)}^{V,t+\Delta t}\}
\end{aligned}
$$

Fully unstructured 4D space-time mesh generation

- Obtain a *global* 4D simplex mesh by appropriate choices of lateral face orientations
- This allows for an optimal space-time mesh (no additional nodes) for all elements not involved in topology changes
- For tetrahedra involved in flips, resort to a “point insertion technique” which is compatible with any face configuration
- Small additional number of nodes, since flips are unusual

Simplex point insertion

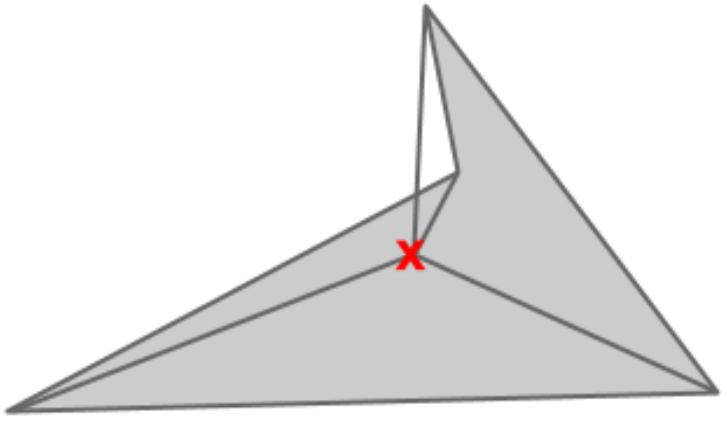
- Need to determine position of the additional node
- For simplicity, consider the cross-section polygon at half-height
- Find the node location that produces non-inverted elements and maximizes the element qualities

$$\min_{\boldsymbol{x}_e} \sum_{K \in \mathcal{T}^h} \frac{1}{Q(K)^4}$$

$$s.t.\ \boldsymbol{A}\boldsymbol{x}_e \leq \boldsymbol{b}$$

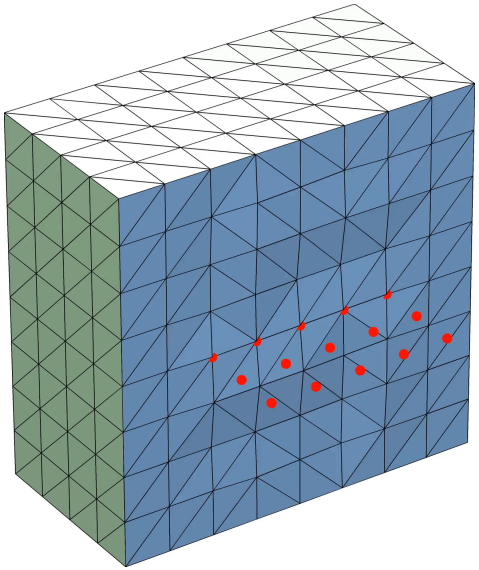
where the inequality constraints express the visibility conditions

- Only apply to elements involved in flips – few additional nodes/elements

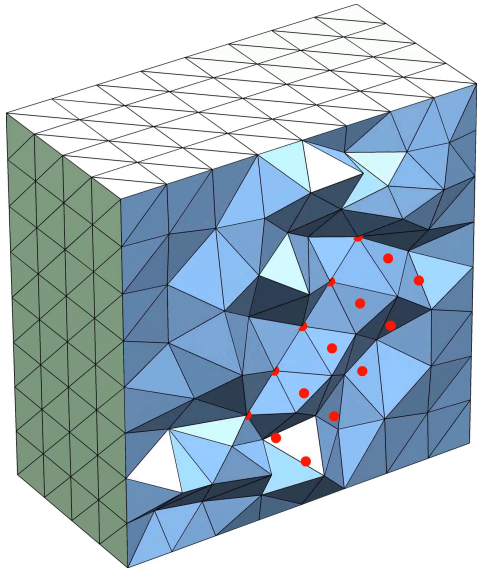


Inverted elements

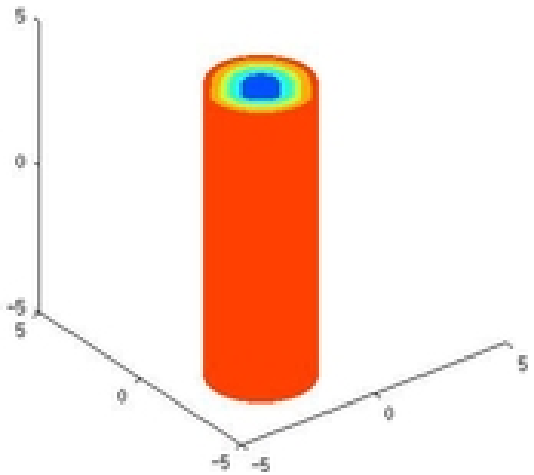
Extruded 3D Euler vortex convergence test



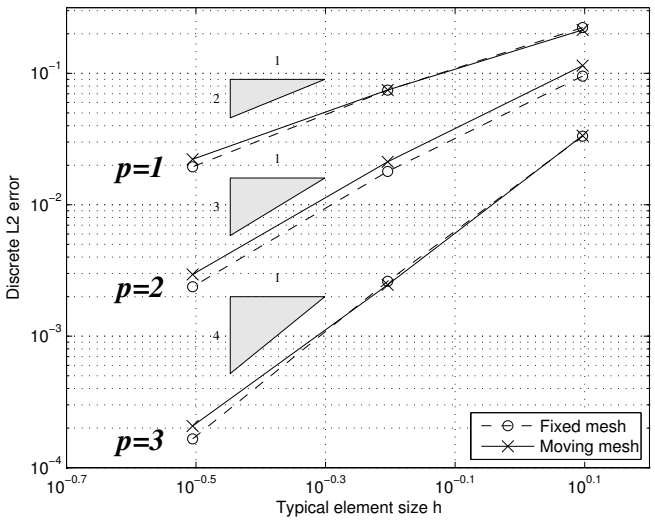
Initial mesh at $t = 0$



Moving mesh at $t = T$



Sample solution



Convergence Plot

Summary

- DG and related high-order methods are getting sufficiently mature to handle realistic problems
- For moving domains with large deformations, novel mesh generation techniques and numerical schemes are required
- Constructive methods for generation of unstructured space-time 3D/4D simplex meshes
- Applications in DNS/LES/DDES flow problems, flapping flight, wind turbine simulations, etc