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, M. Fortunato, J. Wilkening, J. Peraire

MathCCES seminar
Center for Computational Engineering Science, RWTH Aachen

University of California Berkeley

December 8, 2016

Berkeley Lab

Outline

1. Introduction and Motivation

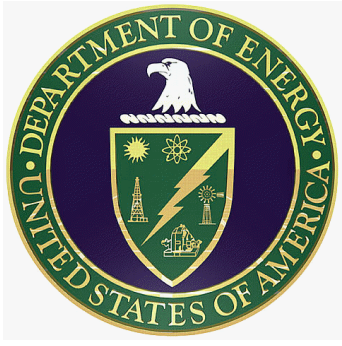
2. Numerical Schemes – Discretization and Solvers
   - The Discontinuous Galerkin Method
   - Time-Stepping and Parallel Implicit Solvers
   - Shock Capturing using Artificial Viscosity

3. Methods for Deforming Domains
   - High-Order ALE Formulation
   - Time-Dependent PDE-Constrained Optimization
   - Unstructured Mesh Space-Time Methods

Outline

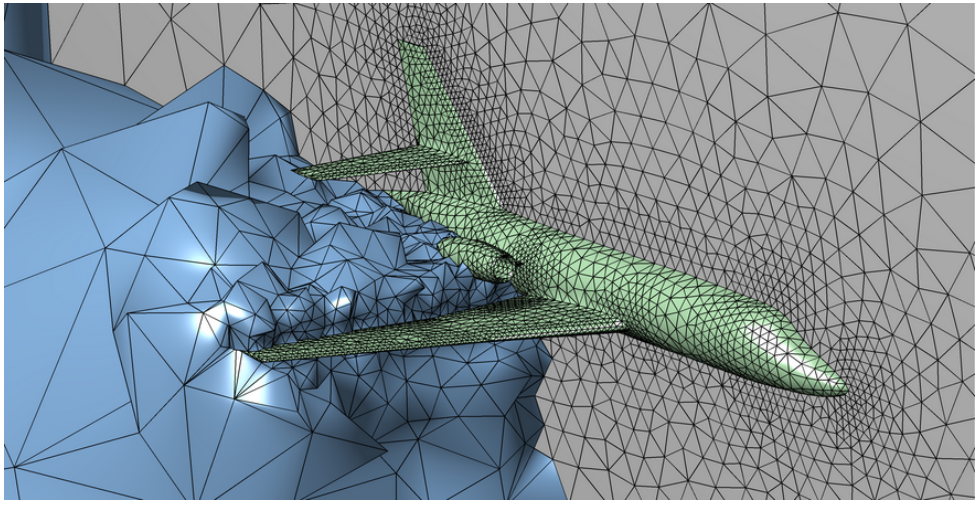
Motivation

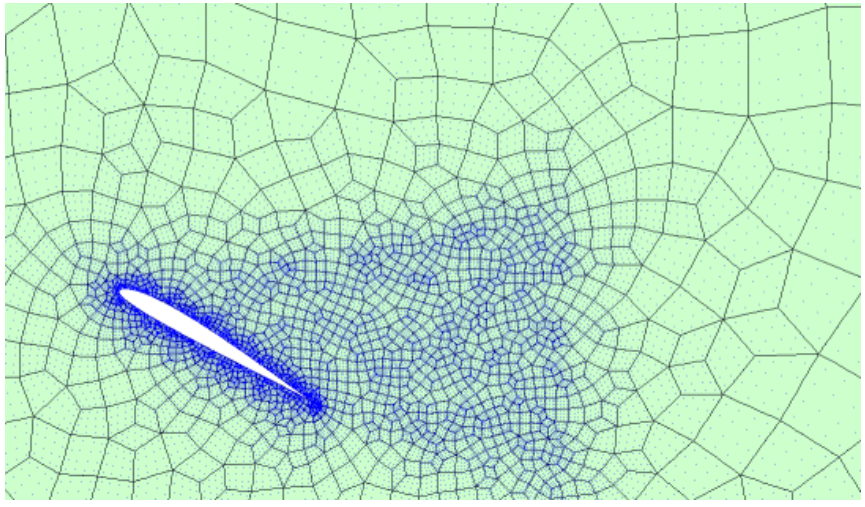
- Need for higher fidelity predictions in computational mechanics
  - Turbulent flows, wave propagation, multiscale phenomena, non-linear interactions
- Many practical applications involve time-varying geometries
  - Fluid/structure interaction, flapping flight, wind turbines, rotor-stator flows
- Goal: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



Why Unstructured Meshes?

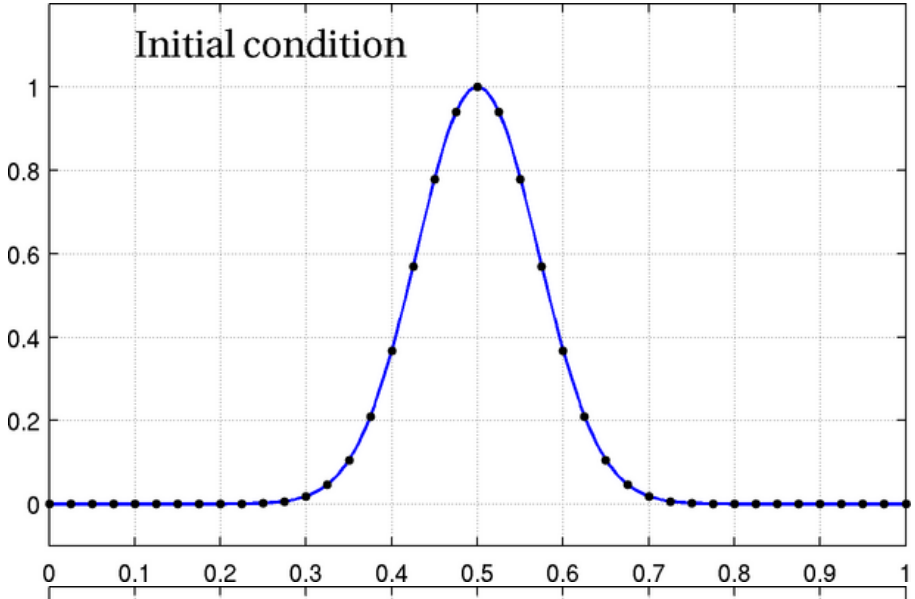
- Complex *geometries* need flexible element topologies
- Complex *solution fields* need spatially variable resolution
- Fully automated mesh generators for CAD geometries are based on unstructured simplex elements
- Real-world simulation software dominated by unstructured mesh discretization schemes

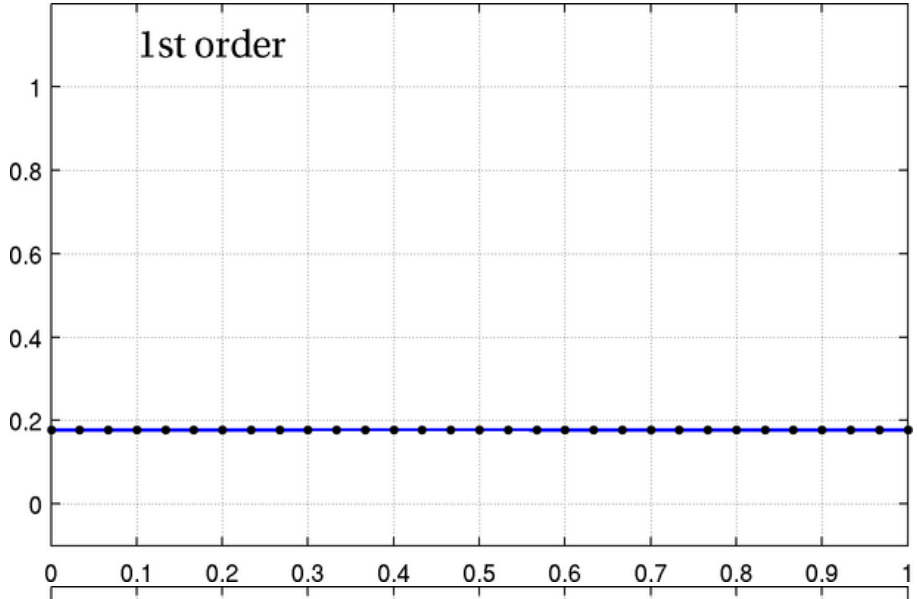


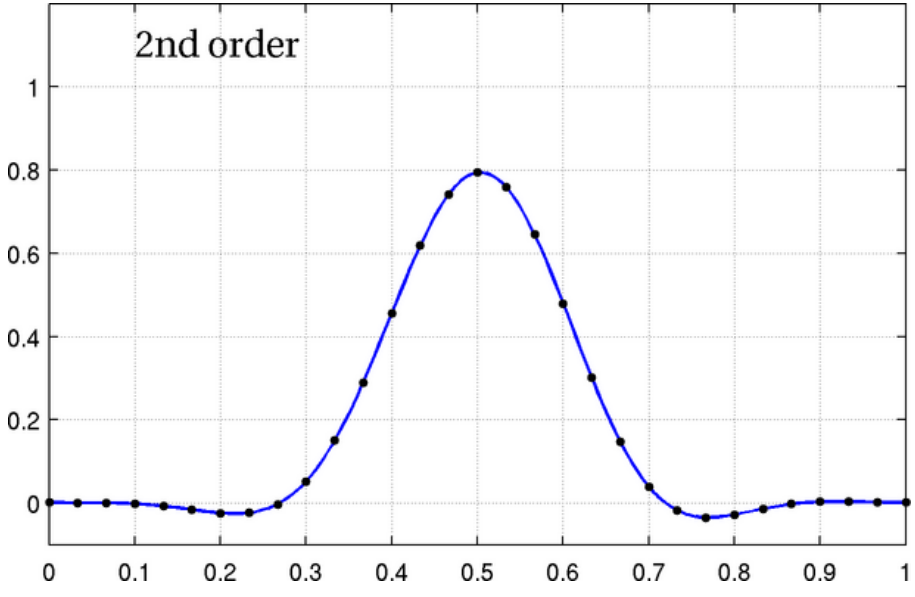


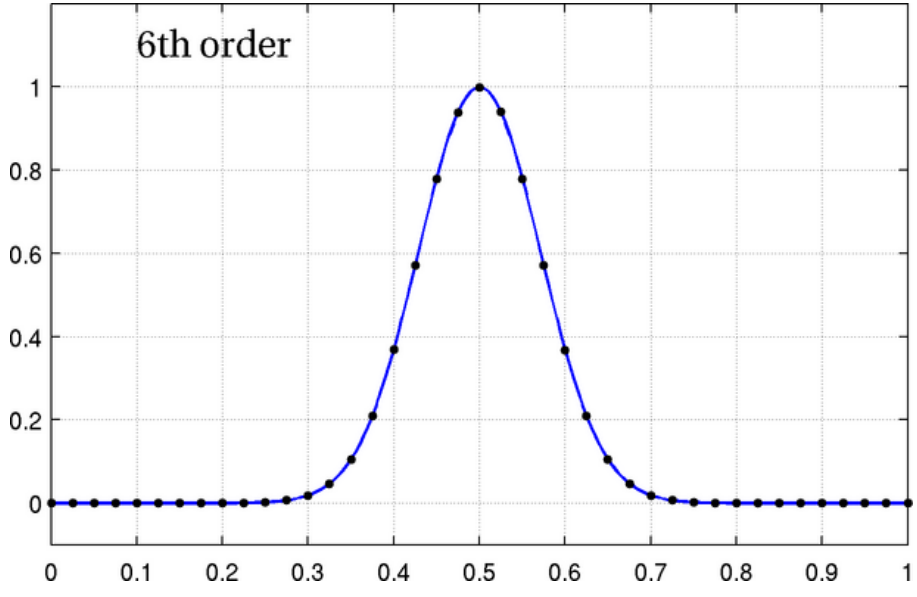
Why high-order accurate methods?

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



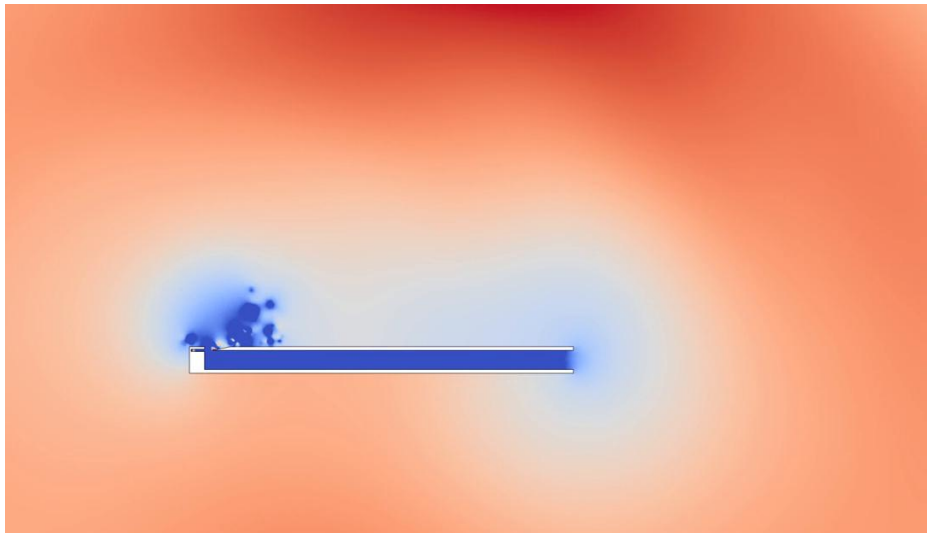


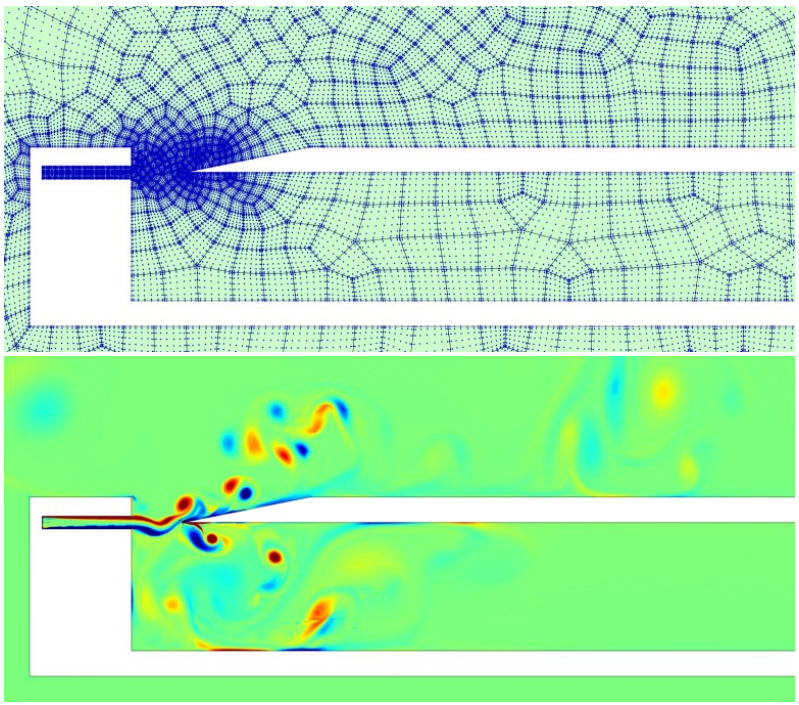


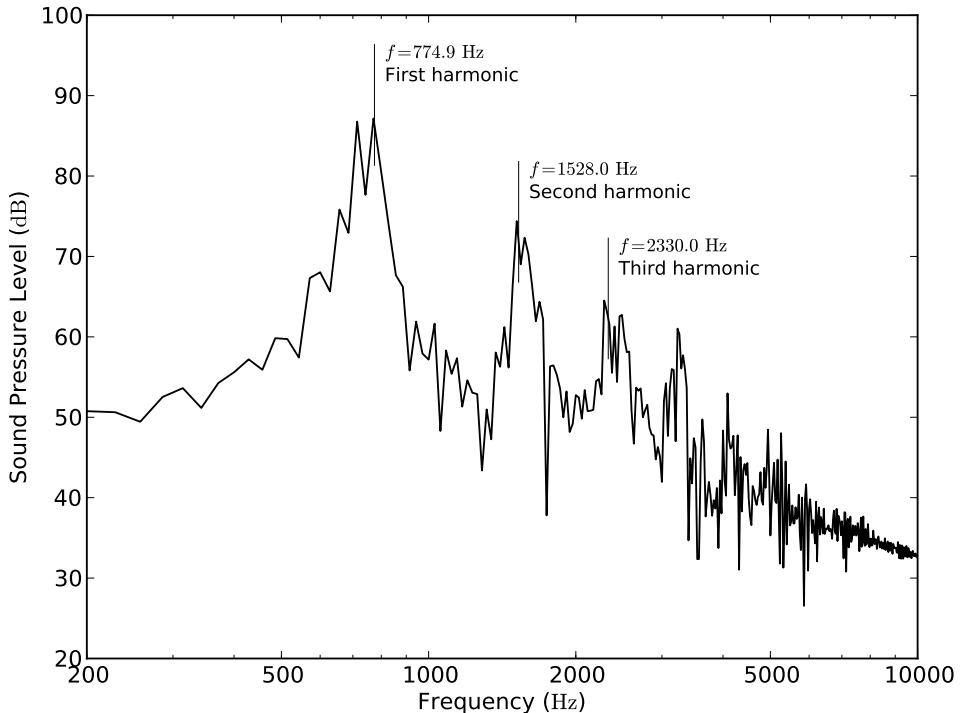


Example: High-order aero-acoustics

- Aero-acoustics simulation of a recorder model
- Line-DG scheme, $p = 7$
- High-order essential to capture turbulent flow sources and propagate waves







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:

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

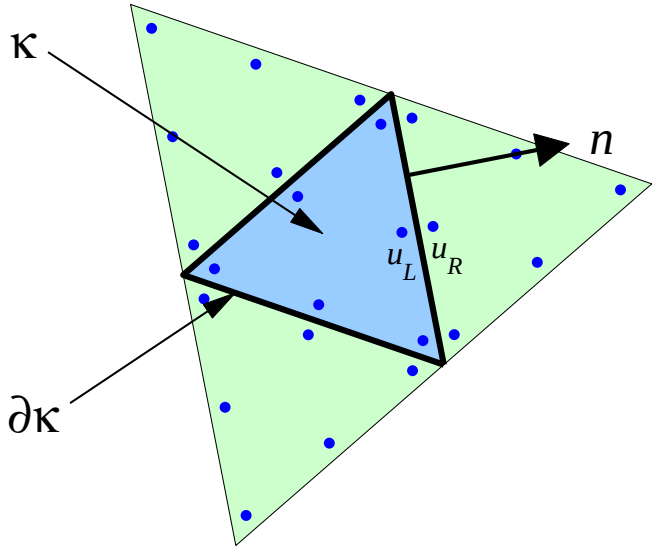
The Discontinuous Galerkin Method

- 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



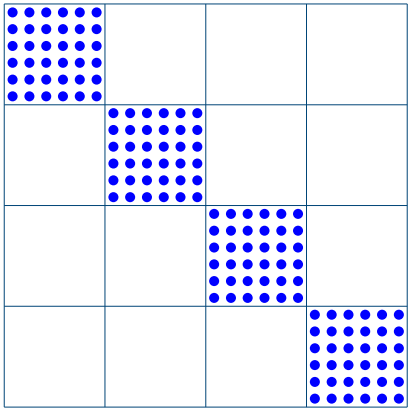
The DG Method – Observations

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

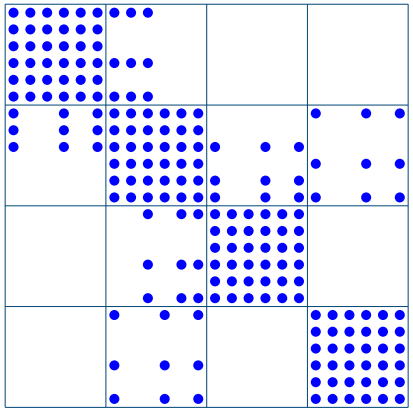
$$(\boldsymbol{u}_h)_t A_\kappa + \int_{\partial\kappa} \hat{\mathcal{F}}_i(\boldsymbol{u}_h^+, \boldsymbol{u}_h^-, \hat{\boldsymbol{n}})\, ds = 0$$

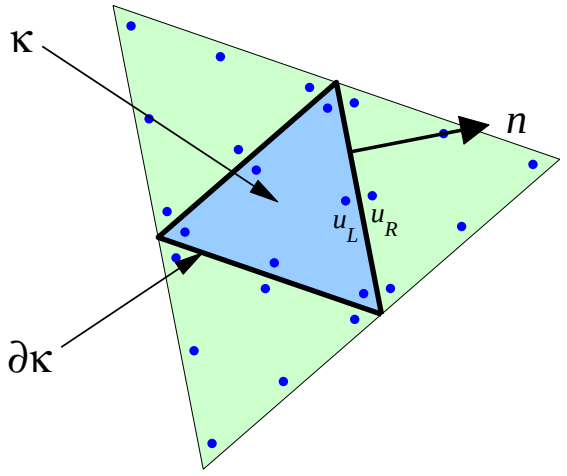
- Boundary conditions enforced naturally for any degree p
- Block-diagonal mass matrix (no overlap between basis functions)
- Block-wise compact stencil – neighboring elements connected

Mass Matrix



Jacobian





Viscous Discretization

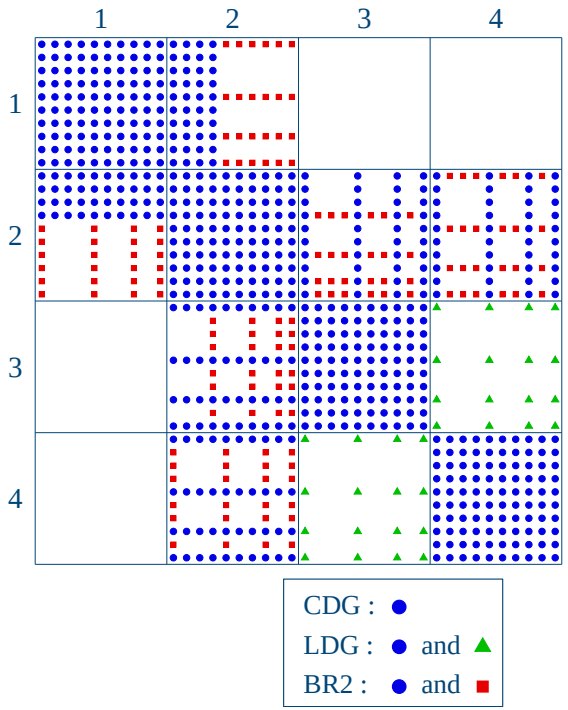
- General approach for second derivatives:
  - Write as system of first order equations [Arnold et al 02]:

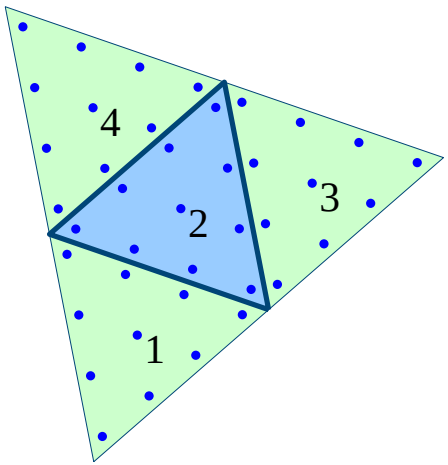
$$\boldsymbol{u}_t + \nabla \cdot \mathcal{F}_i(\boldsymbol{u}) - \nabla \cdot \mathcal{F}_v(\boldsymbol{u}, \boldsymbol{\sigma}) = 0$$

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

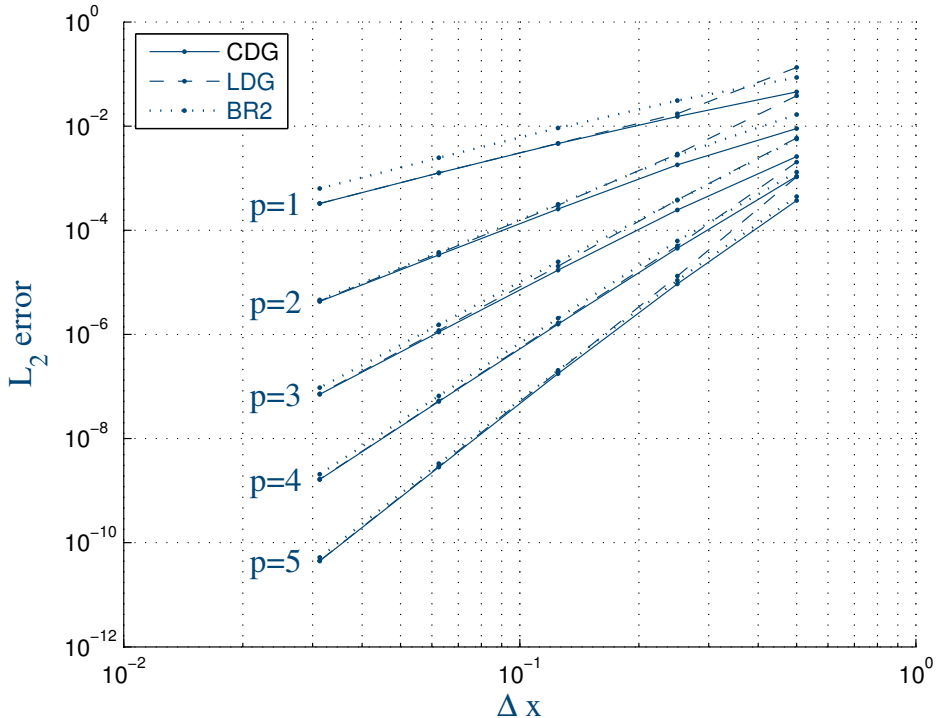
  - Discretize using DG, choose appropriate numerical fluxes $\hat{\boldsymbol{\sigma}}$, $\hat{u}$
- Various schemes have been proposed:
  - *BR2* [Bassi/Rebay 1998]: Different lifting operator for each edge, compact connectivities, similar to Interior Penalty (IP)
  - *LDG* [Cockburn/Shu 1998]: Upwind/Downwind,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

The 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

Temporal Discretization: DIRK

- Diagonally Implicit RK (DIRK) are implicit Runge-Kutta schemes defined by lower triangular Butcher tableau $\rightarrow$ **decoupled implicit stages**
- Overcomes issues with high-order BDF and IRK
  - Limited accuracy of A-stable BDF schemes (2nd order)
  - High cost of general implicit RK schemes (coupled stages)

$$\boldsymbol{u}^{(0)} = \boldsymbol{u}_0(\boldsymbol{\mu})$$

$$\boldsymbol{u}^{(n)} = \boldsymbol{u}^{(n-1)} + \sum_{i=1}^{s} b_i \boldsymbol{k}_i^{(n)}$$

$$\boldsymbol{u}_i^{(n)} = \boldsymbol{u}^{(n-1)} + \sum_{j=1}^{i} a_{ij} \boldsymbol{k}_j^{(n)}$$

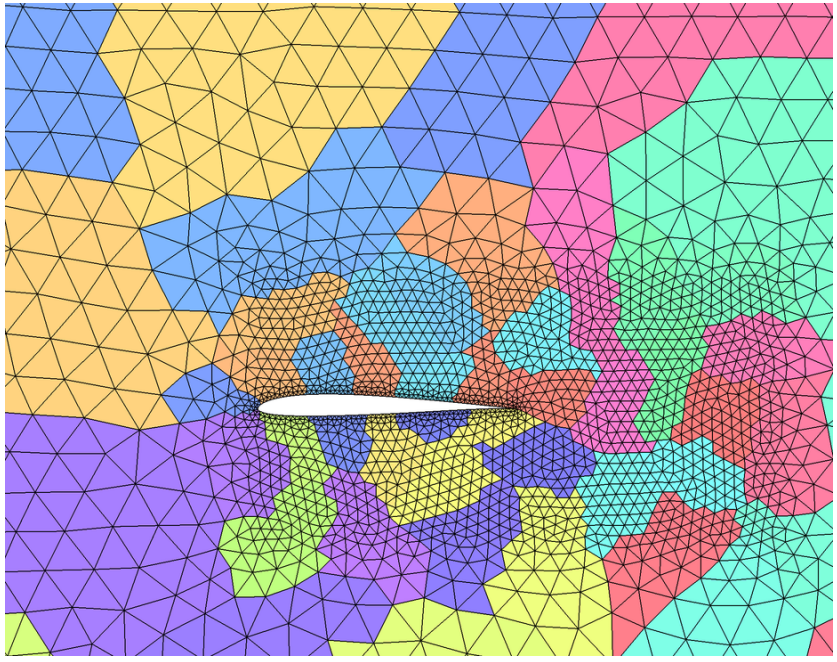
$$\mathbb{M}\boldsymbol{k}_i^{(n)} = \Delta t_n \boldsymbol{r}\left(\boldsymbol{u}_i^{(n)},\ \boldsymbol{\mu},\ t_{n-1} + c_i \Delta t_n\right)$$

$$\begin{array}{c|cccc} c_1 & a_{11} & & & \\ c_2 & a_{21} & a_{22} & & \\ \vdots & \vdots & \vdots & \ddots & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

Butcher Tableau for DIRK scheme

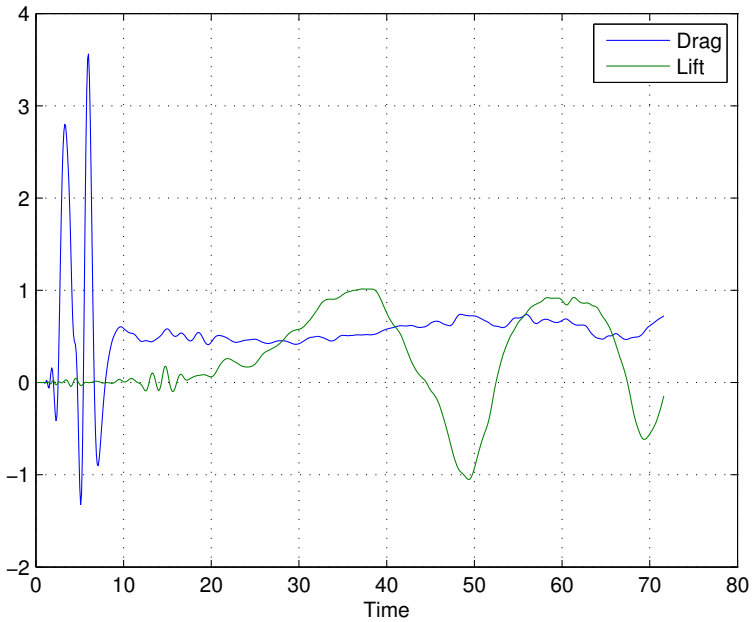
Preconditioning for Newton-Krylov Solvers

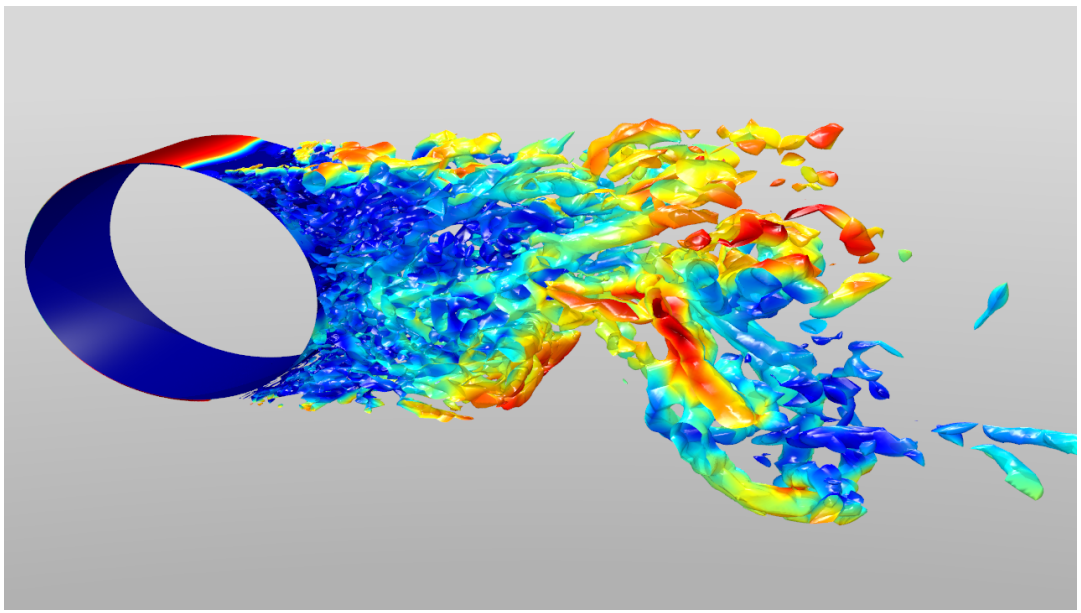
- Implicit solvers typically required because of CFL restrictions from viscous effects, low Mach numbers, and adaptive/anisotropic grids
- Jacobian matrices are large even at $p = 2$ or $p = 3$, however:
  - They are required for non-trivial preconditioners
  - They are very expensive to recompute
- Block-ILU(0) preconditioners and Minimum Discarded Fill (MDF) element ordering [Persson/Peraire 2008]
- Distributed parallel solvers developed in [Persson '09]
- IMEX schemes for geometrically induced stiffness (e.g. boundary layers) [Persson 2011]



Cylinder – Delayed Detached Eddy Simulation

- DES/DDES [Spalart 1997,2006]: Hybrid RANS/LES model for problems with significant flow separation
- Edge of boundary layer moving; stabilize by continuous AV
- Model problem: Flow over cylinder, $\mathrm{Re} = 1.5 \cdot 10^6$
- Left: Force coefficients C_D, C_L, Right: Q-criterion isosurfaces

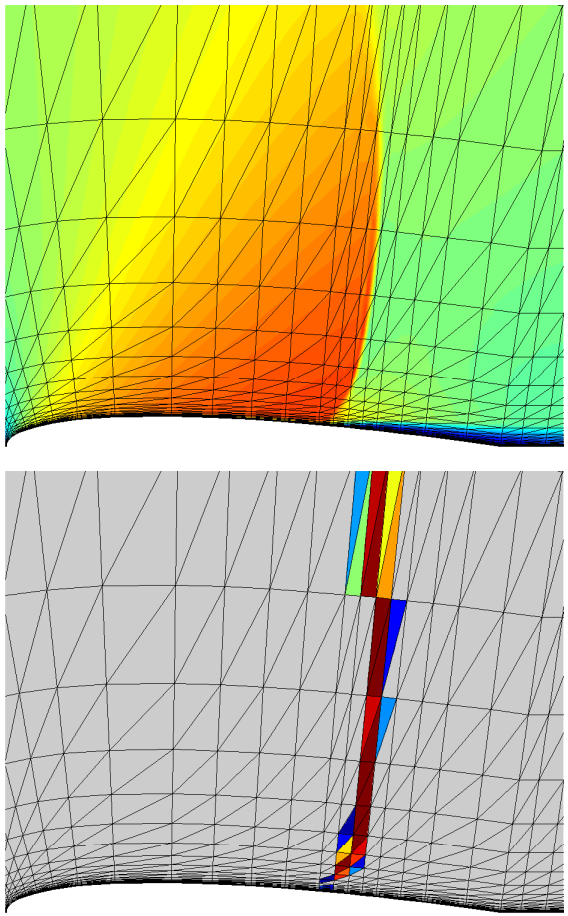




Outline

Artificial Viscosity for Underresolved Features

- Cannot resolve all solution features (shocks, RANS, singularities), and low dissipation makes DG sensitive to under-resolution
- Stabilize by sensors and artificial viscosity [Persson & Peraire 2006]
- Regularity of solution determined from the decay rate of expansion coefficients in orthogonal basis
- Periodic Fourier case: $f(x) = \sum_{k=-\infty}^{\infty} g_k e^{ikx}$



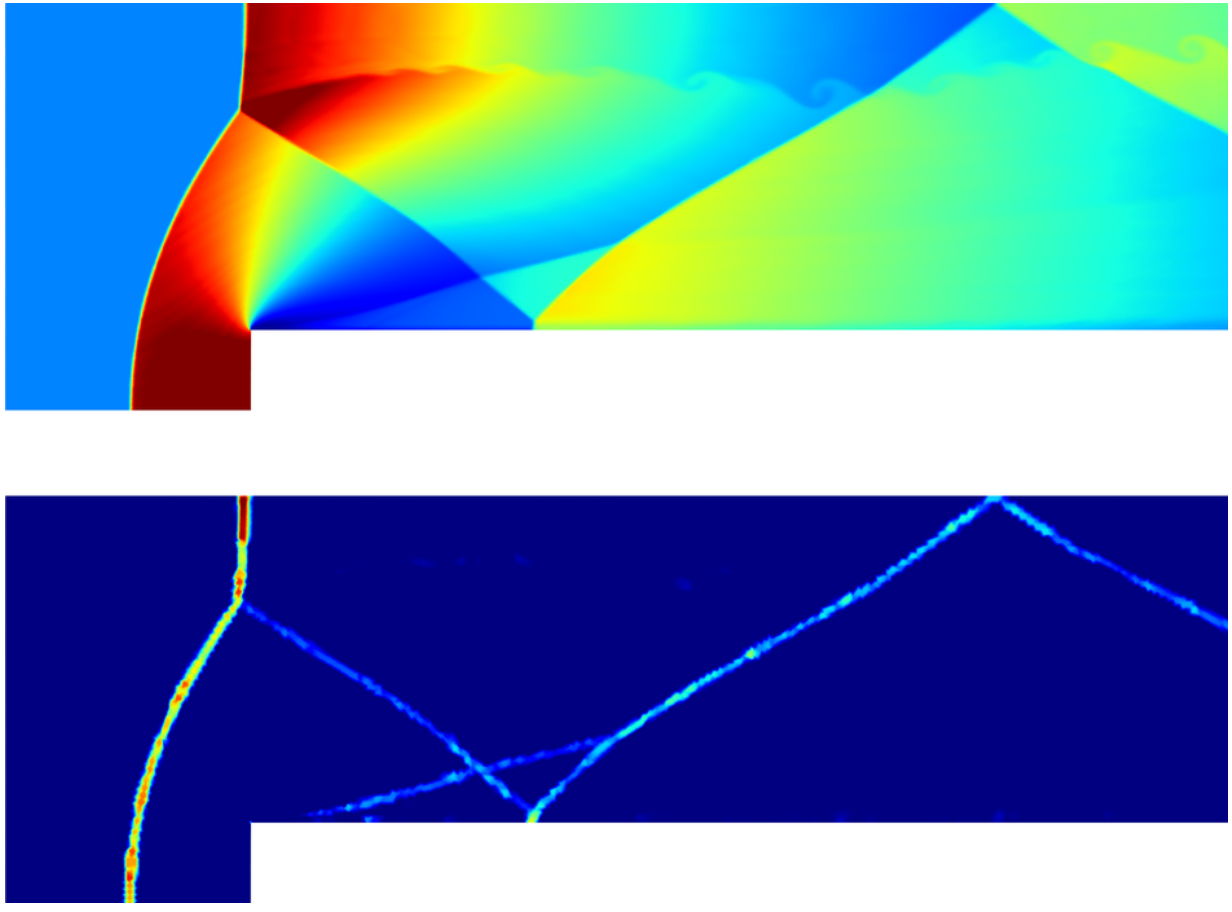
If $f(x)$ has m continuous derivatives $\rightarrow \ |g_k| \sim k^{-(m+1)}$

- For simplices: Expand solution in orthonormal Koornwinder basis:

$$u = \sum_{i=1}^{N(p)} u_i \psi_i, \quad \hat{u} = \sum_{i=1}^{N(p-1)} u_i \psi_i, \quad s_e = \log_{10}\left(\frac{(u-\hat{u}, u-\hat{u})_e}{(u,u)_e}\right)$$

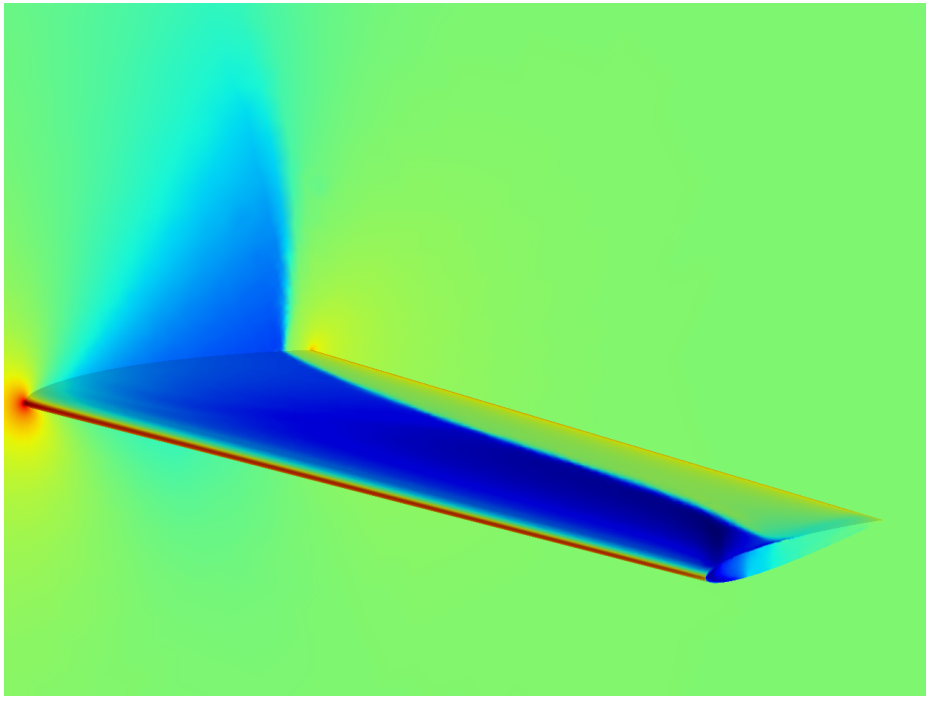
Nonlinear stabilization using artificial viscosity

- s_e is a highly sensitive yet selective sensor
- General $\rightarrow$ applicable to any type of under-resolved features
- Allows for full Newton convergence of sensor
- Subgrid resolution $\rightarrow$ smooth propagation of moving shocks

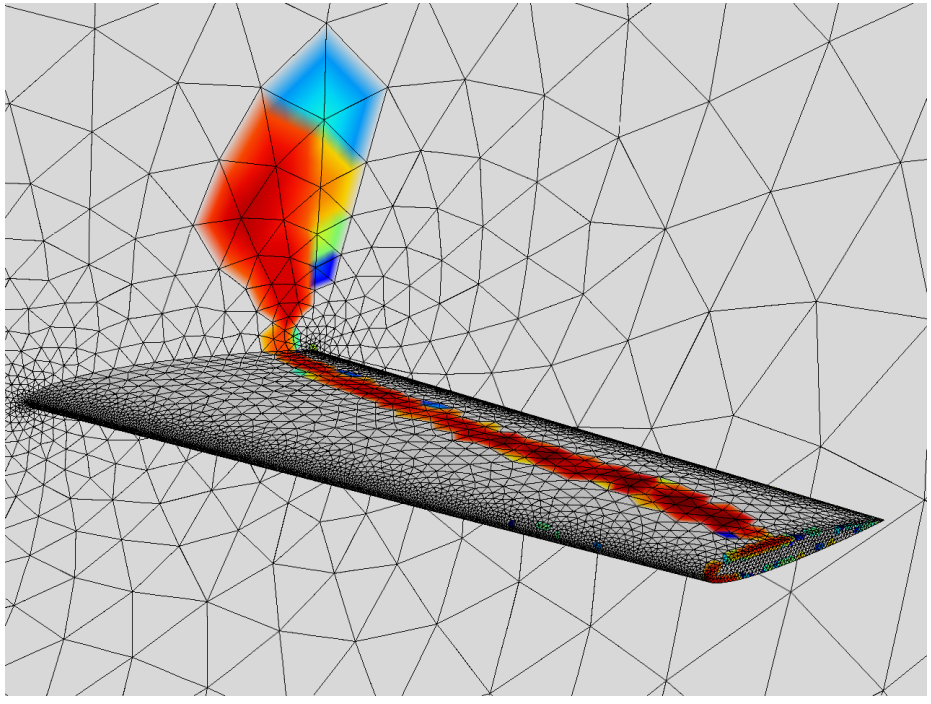


3-D Transonic Flow over Tapered Wing

- 3-D wing, fully unstructured tetrahedral mesh
- Freestream $\mathrm{Ma} = 0.8$, AoA = 3°
- Polynomial degrees $p = 3$



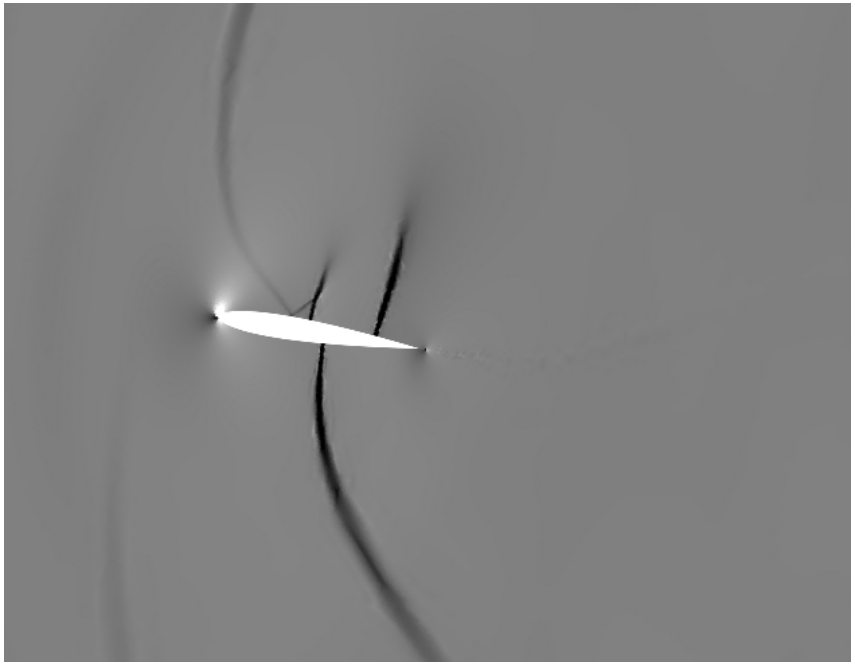
Pressure



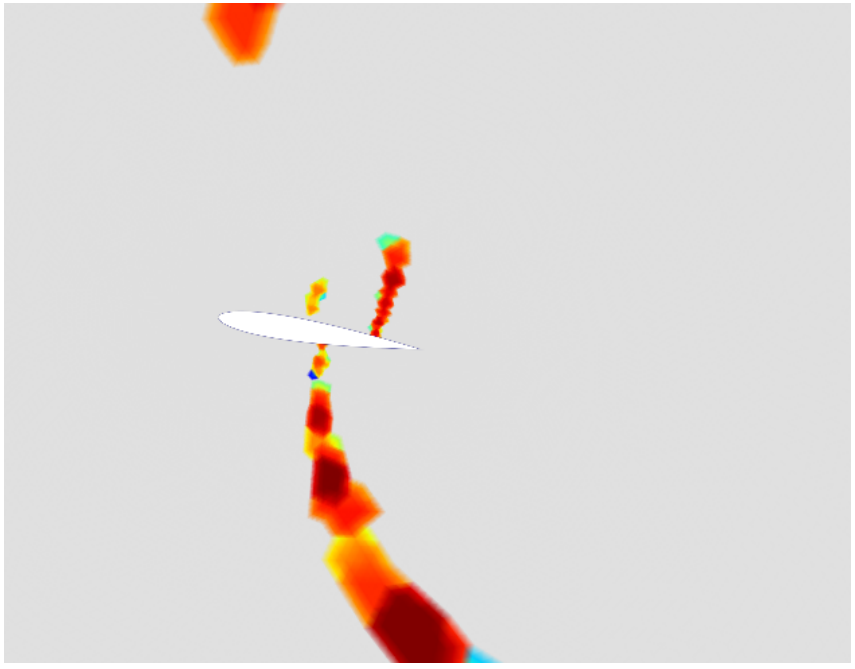
Artificial viscosity

Pitching wing in transonic flow

- ALE formulation for the Euler equations combined with sensor-based artificial diffusion
- Freestream $\mathrm{Ma} = 0.6$, AoA harmonic with amplitude= 30°
- Implicit solver, re-used Jacobians



x-gradient of density



Arfificial viscosity

Outline

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 $g = \det(\boldsymbol{G})$
- 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 then be written in the reference configuration V as

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

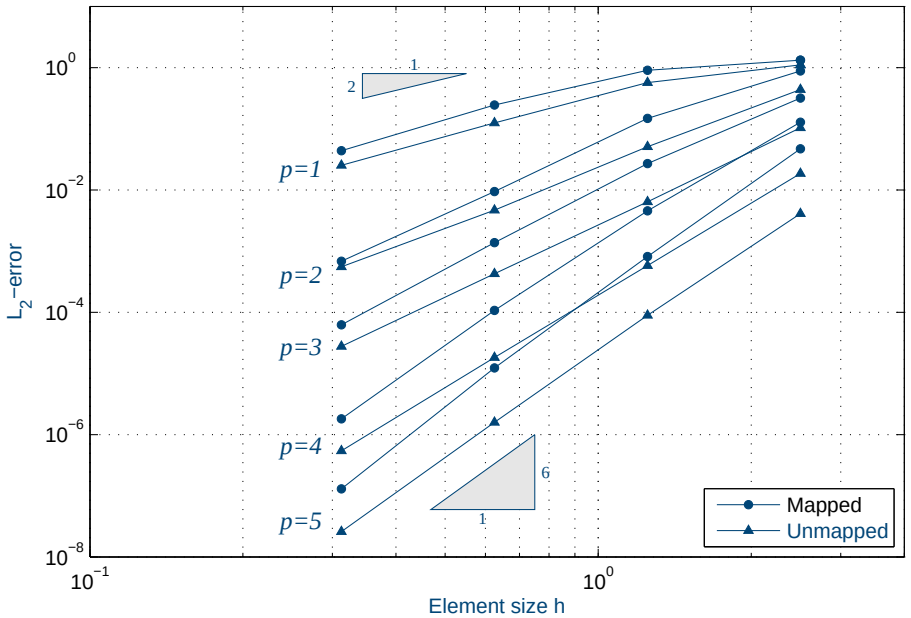
  where

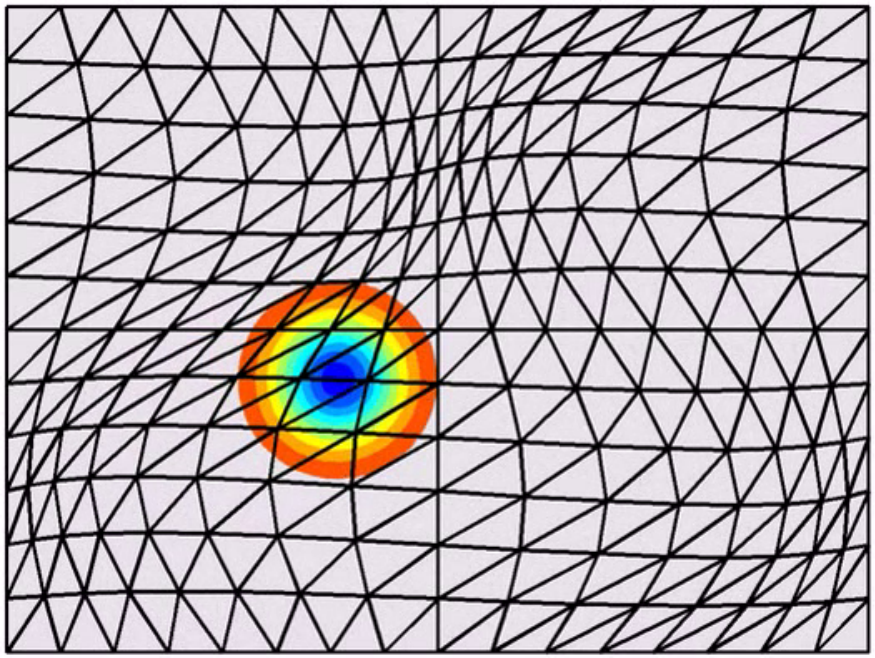
$$\boldsymbol{U}_X = g\boldsymbol{U}_x\ , \qquad \boldsymbol{F}_X = g\boldsymbol{G}^{-1}\boldsymbol{F}_x - \boldsymbol{U}_X\boldsymbol{G}^{-1}\boldsymbol{v}_X$$

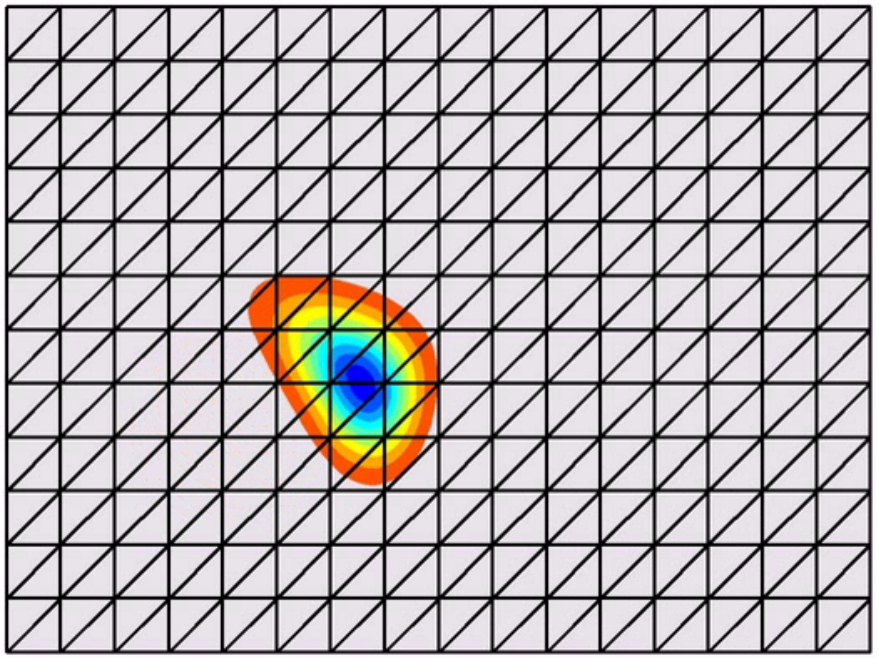
$$\boldsymbol{\nabla}_x \boldsymbol{U}_x = \boldsymbol{\nabla}_X(g^{-1}\boldsymbol{U}_X)\boldsymbol{G}^{-T} = (g^{-1}\boldsymbol{\nabla}_X \boldsymbol{U}_X - \boldsymbol{U}_X \boldsymbol{\nabla}_X(g^{-1}))\boldsymbol{G}^{-T}$$

ALE Formulation for Deforming Domains

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

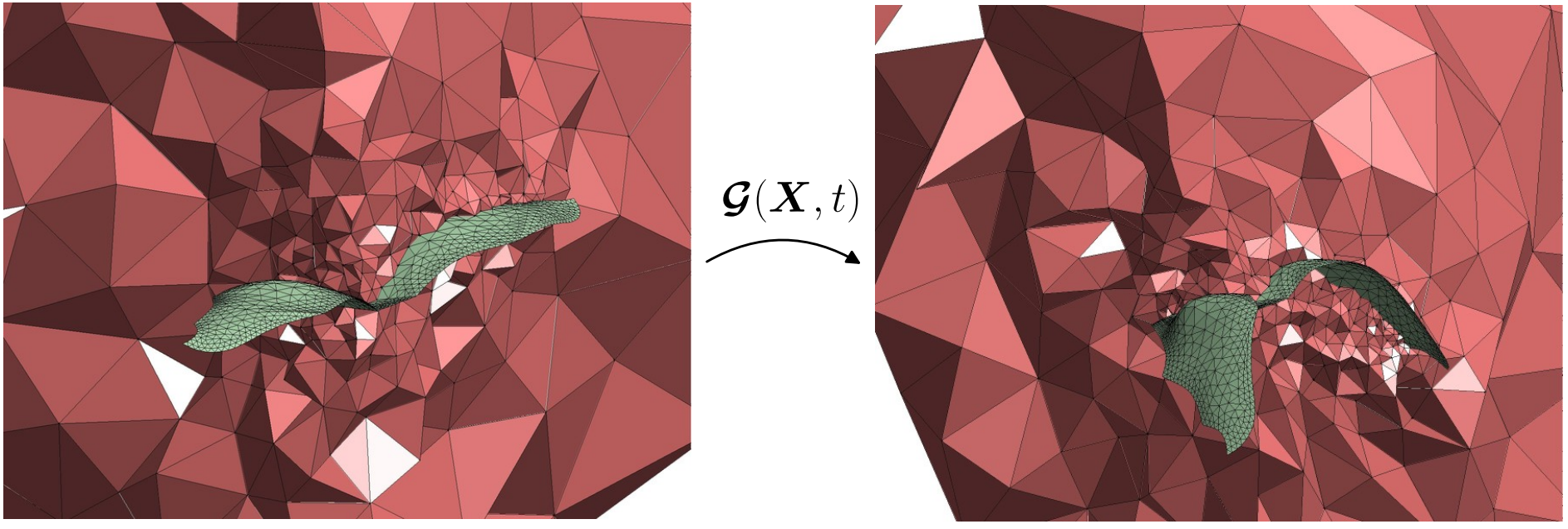




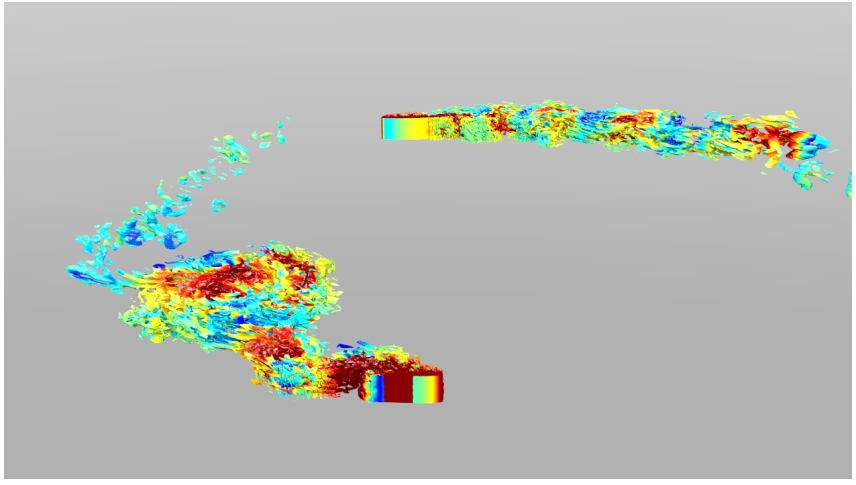


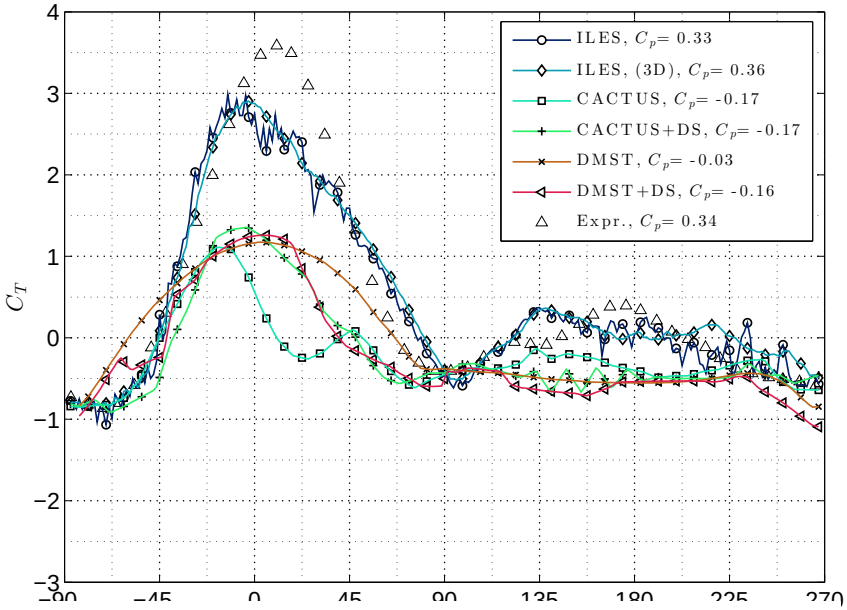
Nonlinear Elasticity for Deforming Domains

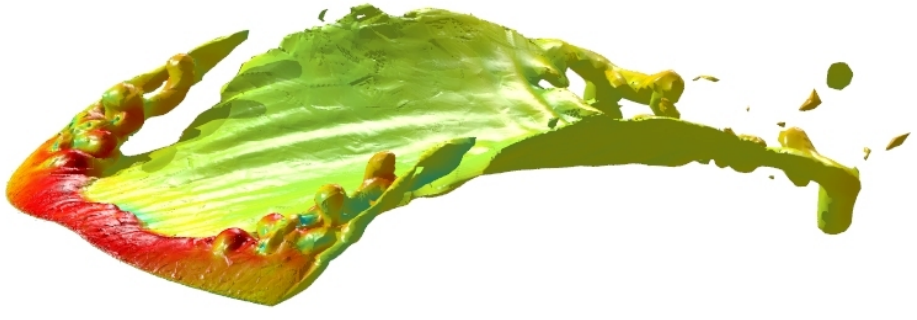
- *Non-linear solid mechanics approach*: [Persson & Peraire 2009]
  - An initial reference mesh corresponds to an undeformed solid
  - External forces come from the true moving boundary constraints
  - Solving for a force equilibrium gives the deformed (curved) boundary conforming mesh
- High-order ALE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

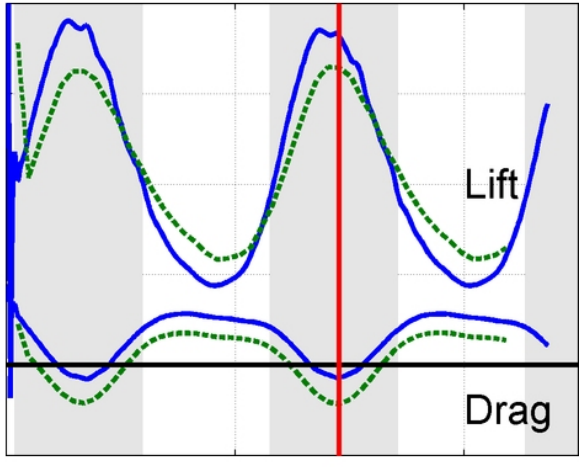


Moving Domain Applications





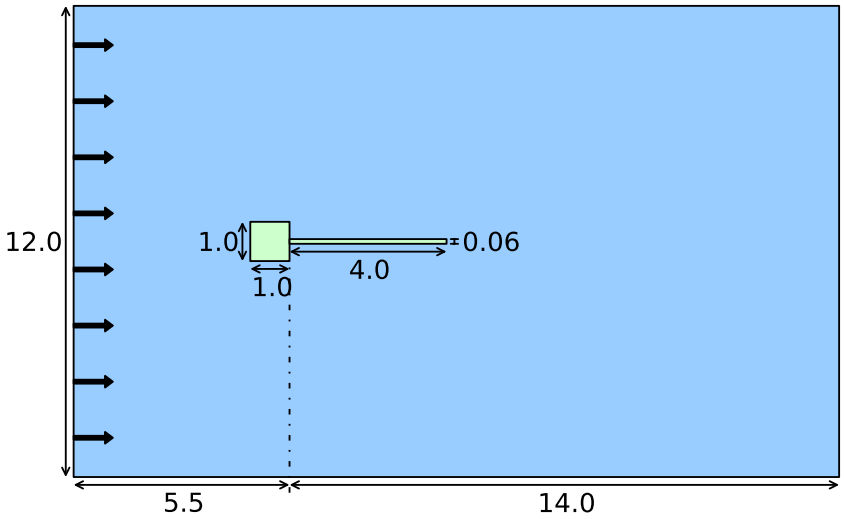


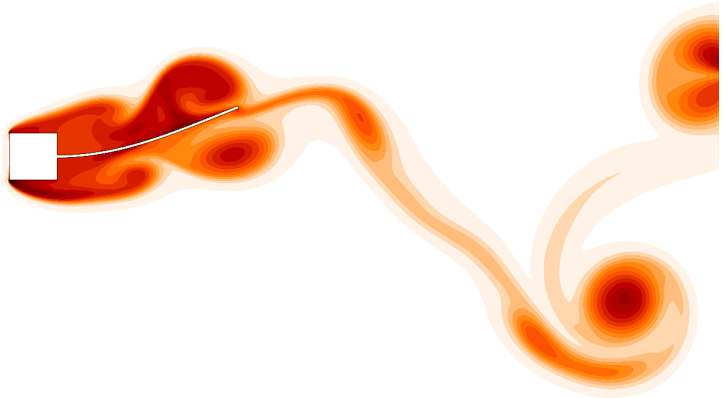


Partitioned FSI using IMEX schemes

- [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:

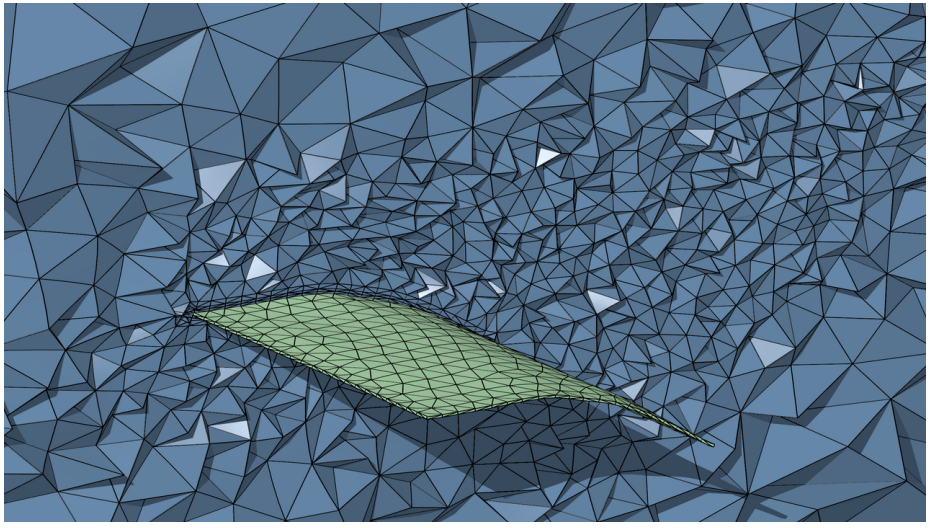
$$r = \begin{bmatrix} r^f(u^f; x(u^s)) \\ r^s(u^s; t(u^f)) \end{bmatrix} = \underbrace{\begin{bmatrix} r^f(u^f; x(u^s)) \\ r^s(u^s; \tilde{t}) \end{bmatrix}}_{\text{implicit}} + \underbrace{\begin{bmatrix} \\ r^{fs}(t(u^f) - \tilde{t}) \end{bmatrix}}_{\text{explicit}}$$



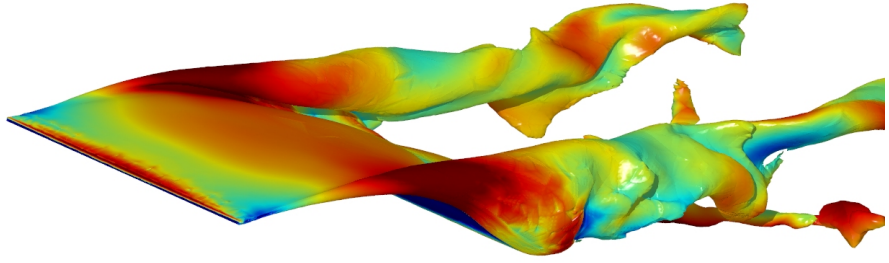


Flow around Membrane, 3-D

- 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

Discretization of PDE-Constrained Optimization

- *Continuous* PDE-constrained optimization problem

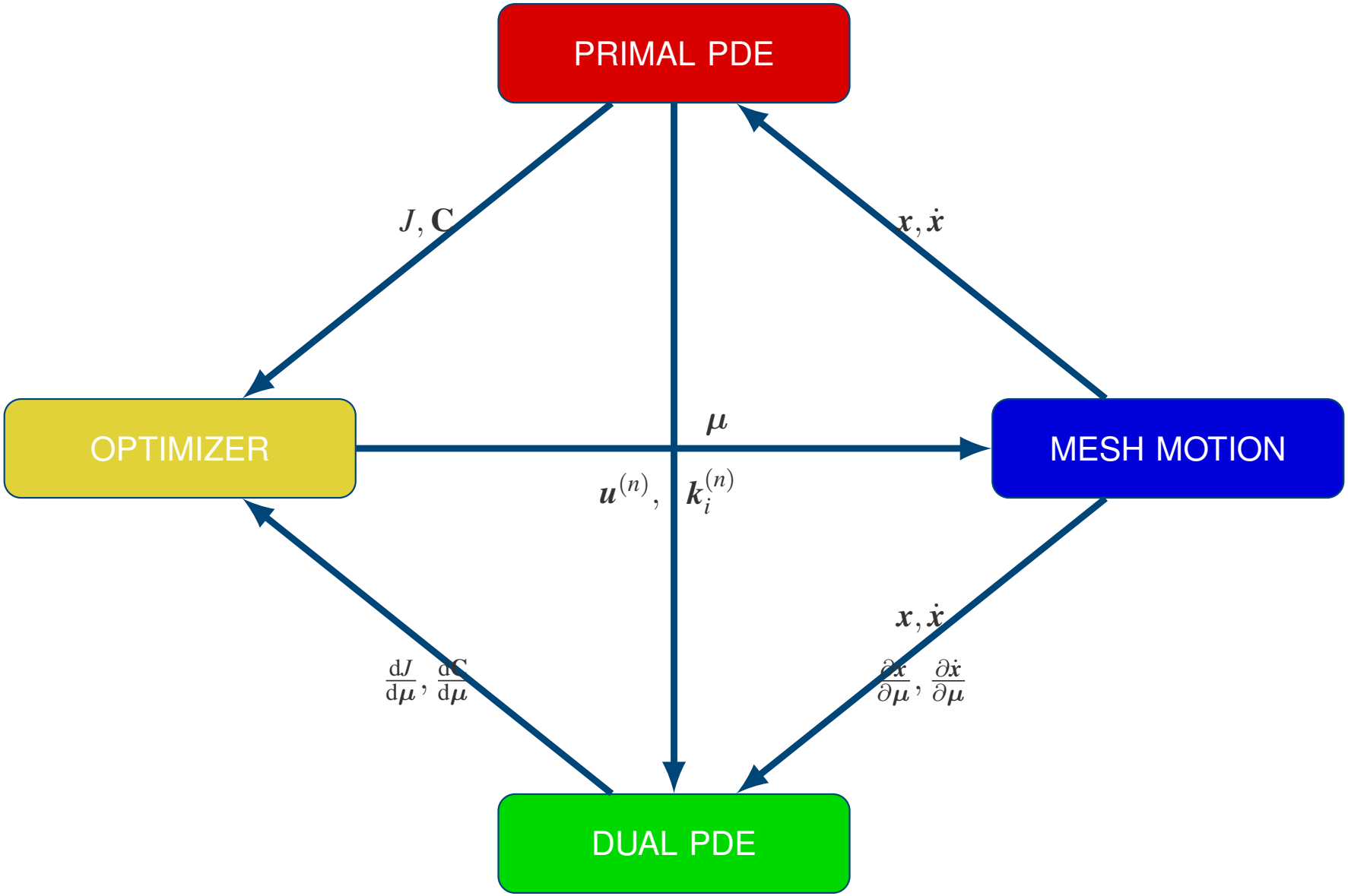
$$\begin{aligned} \underset{\boldsymbol{U},\ \boldsymbol{\mu}}{\text{minimize}} \quad & \mathcal{J}(\boldsymbol{U}, \boldsymbol{\mu}) \\ \text{subject to} \quad & \boldsymbol{C}(\boldsymbol{U}, \boldsymbol{\mu}) \leq 0 \\ & \frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{F}(\boldsymbol{U}, \nabla \boldsymbol{U}) = 0 \ \text{ in } \ v(\boldsymbol{\mu}, t) \end{aligned}$$

- *Fully discrete* PDE-constrained optimization problem

$$\begin{aligned} \underset{\substack{\boldsymbol{u}^{(0)},\ \ldots,\ \boldsymbol{u}^{(N_t)} \in \mathbb{R}^{N_{\boldsymbol{u}}}, \\ \boldsymbol{k}_1^{(1)},\ \ldots,\ \boldsymbol{k}_s^{(N_t)} \in \mathbb{R}^{N_{\boldsymbol{u}}}, \\ \boldsymbol{\mu} \in \mathbb{R}^{n_{\boldsymbol{\mu}}}}}{\text{minimize}} \quad & J(\boldsymbol{u}^{(0)},\ \ldots,\ \boldsymbol{u}^{(N_t)},\ \boldsymbol{k}_1^{(1)},\ \ldots,\ \boldsymbol{k}_s^{(N_t)},\ \boldsymbol{\mu}) \\ \text{subject to} \quad & \mathbf{C}(\boldsymbol{u}^{(0)},\ \ldots,\ \boldsymbol{u}^{(N_t)},\ \boldsymbol{k}_1^{(1)},\ \ldots,\ \boldsymbol{k}_s^{(N_t)},\ \boldsymbol{\mu}) \leq 0 \\ & \boldsymbol{u}^{(0)} - \boldsymbol{u}_0(\boldsymbol{\mu}) = 0 \\ & \boldsymbol{u}^{(n)} - \boldsymbol{u}^{(n-1)} + \sum_{i=1}^{s} b_i \boldsymbol{k}_i^{(n)} = 0 \\ & \mathbb{M}\boldsymbol{k}_i^{(n)} - \Delta t_n \boldsymbol{r}\left(\boldsymbol{u}_i^{(n)},\ \boldsymbol{\mu},\ t_i^{(n-1)}\right) = 0 \end{aligned}$$

Generalized Reduced-Gradient Approach

Optimizer drives, Primal returns QoI values, Dual returns QoI gradients



Adjoint Method to Compute QoI Gradients

- Consider the *fully discrete* output functional $F(\boldsymbol{u}^{(n)}, \boldsymbol{k}_i^{(n)}, \boldsymbol{\mu})$
  - Represents either the **objective** function or a **constraint**
- The *total derivative* with respect to the parameters $\boldsymbol{\mu}$, required in the context of gradient-based optimization, takes the form

$$\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{\mu}} = \frac{\partial F}{\partial \boldsymbol{\mu}} + \sum_{n=0}^{N_t} \frac{\partial F}{\partial \boldsymbol{u}^{(n)}} \frac{\partial \boldsymbol{u}^{(n)}}{\partial \boldsymbol{\mu}} + \sum_{n=1}^{N_t} \sum_{i=1}^{s} \frac{\partial F}{\partial \boldsymbol{k}_i^{(n)}} \frac{\partial \boldsymbol{k}_i^{(n)}}{\partial \boldsymbol{\mu}}$$

- The sensitivities, $\frac{\partial \boldsymbol{u}^{(n)}}{\partial \boldsymbol{\mu}}$ and $\frac{\partial \boldsymbol{k}_i^{(n)}}{\partial \boldsymbol{\mu}}$, are expensive to compute, requiring the solution of $n_{\boldsymbol{\mu}}$ linear evolution equations
- **Adjoint method**: alternative method for computing $\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{\mu}}$ that require one linear evolution evoluation equation for each quantity of interest, F

Fully Discrete Adjoint Equations: Dissection

- **Linear** evolution equations solved **backward** in time
- **Primal** state $\boldsymbol{u}_i^{(n)}$ required at each stage of dual problem
- Heavily dependent on **chosen output**

$$\boldsymbol{\lambda}^{(N_t)} = \frac{\partial F}{\partial \boldsymbol{u}^{(N_t)}}^T$$

$$\boldsymbol{\lambda}^{(n-1)} = \boldsymbol{\lambda}^{(n)} + \frac{\partial F}{\partial \boldsymbol{u}^{(n-1)}}^T + \sum_{i=1}^{s} \Delta t_n \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} \left(\boldsymbol{u}_i^{(n)},\ \boldsymbol{\mu},\ t_{n-1} + c_i \Delta t_n \right)^T \boldsymbol{\kappa}_i^{(n)}$$

$$\mathbb{M}^T \boldsymbol{\kappa}_i^{(n)} = \frac{\partial F}{\partial \boldsymbol{u}^{(N_t)}}^T + b_i \boldsymbol{\lambda}^{(n)} + \sum_{j=i}^{s} a_{ji} \Delta t_n \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} \left(\boldsymbol{u}_j^{(n)},\ \boldsymbol{\mu},\ t_{n-1} + c_j \Delta t_n \right)^T \boldsymbol{\kappa}_j^{(n)}$$

- Gradient reconstruction via dual variables

$$\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{\mu}} = \frac{\partial F}{\partial \boldsymbol{\mu}} + \boldsymbol{\lambda}^{(0)^T} \frac{\partial \boldsymbol{u}_0}{\partial \boldsymbol{\mu}} + \sum_{n=1}^{N_t} \Delta t_n \sum_{i=1}^{s} \boldsymbol{\kappa}_i^{(n)^T} \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\mu}} (\boldsymbol{u}_i^{(n)},\ \boldsymbol{\mu},\ t_i^{(n)})$$

Energetically Optimal Flapping, Thrust Constraint

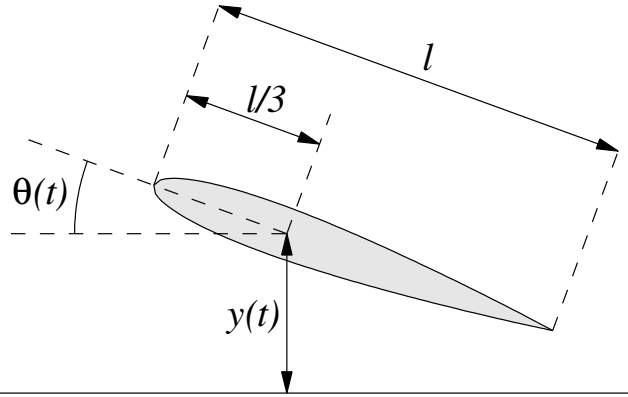
$$\underset{\boldsymbol{\mu}}{\text{minimize}} \quad -\int_{2T}^{3T}\int_{\Gamma} \boldsymbol{f}\cdot\dot{\boldsymbol{x}}\, dS\, dt$$

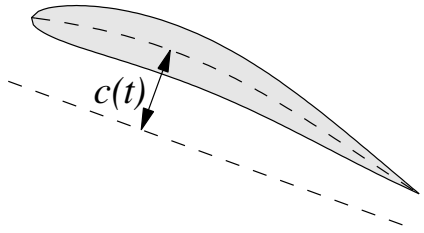
$$\text{subject to} \quad \int_{2T}^{3T}\int_{\Gamma} \boldsymbol{f}\cdot\boldsymbol{e}_1\, dS\, dt = q$$

$$\boldsymbol{U}(\boldsymbol{x},0) = \bar{\boldsymbol{U}}(\boldsymbol{x})$$

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

- Isentropic, compressible, Navier-Stokes
- Re = 1000, M = 0.2
- $y(t)$, $\theta(t)$, $c(t)$ parametrized via periodic cubic splines
- Black-box optimizer: SNOPT

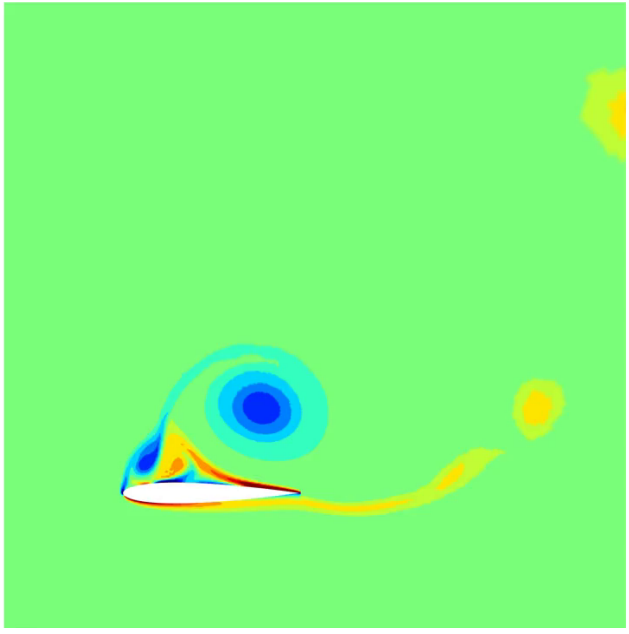
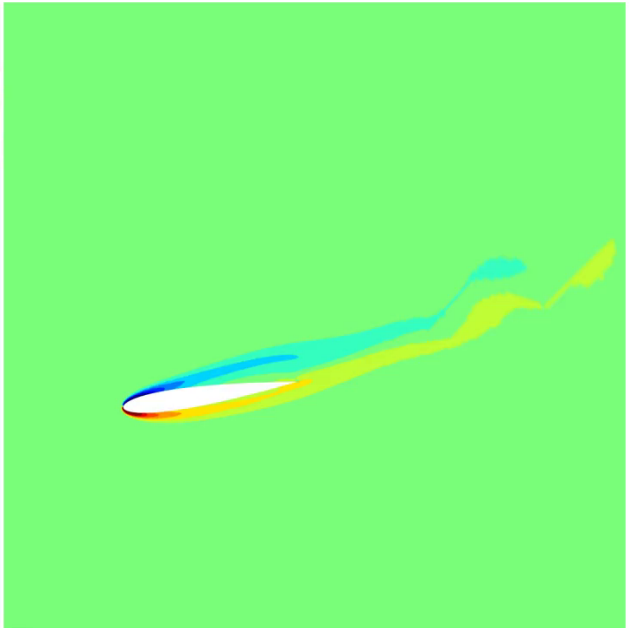
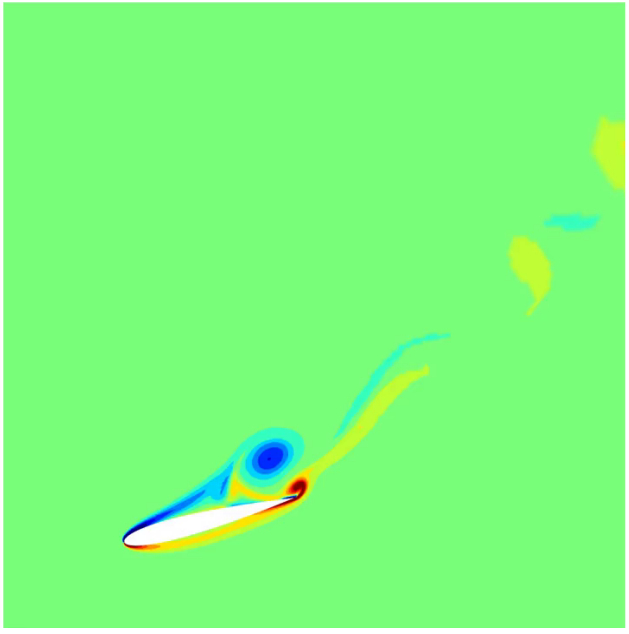




Airfoil schematic, kinematic description

Optimal Control - Fixed Shape

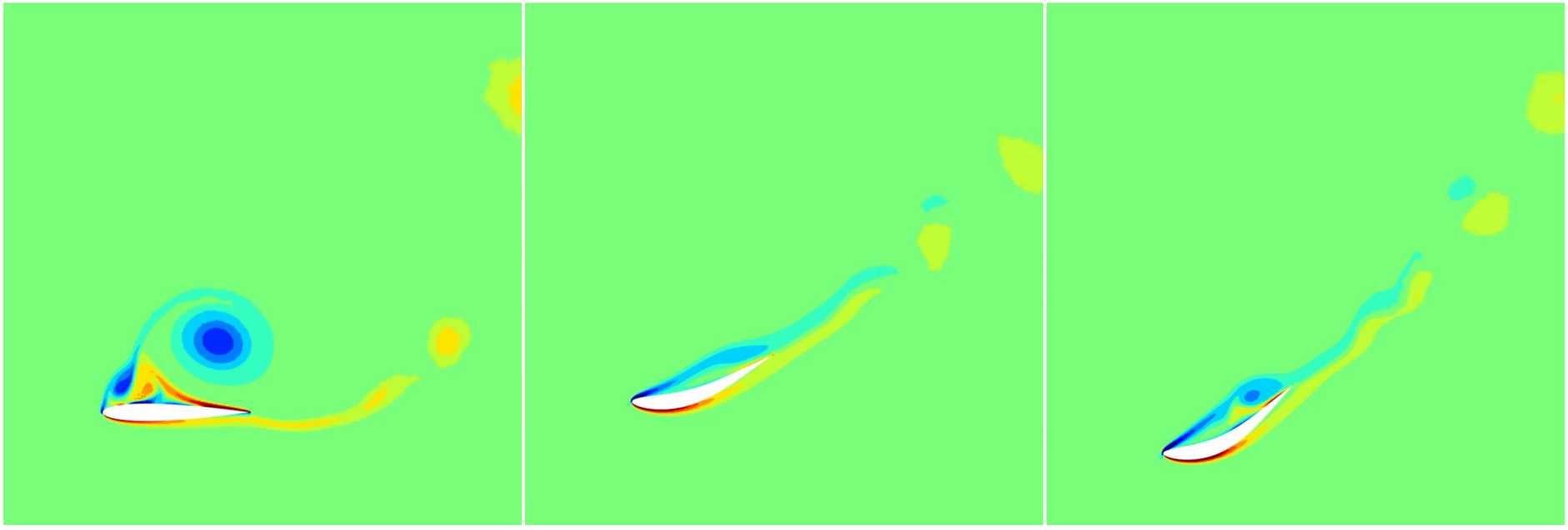
Fixed Shape, Optimal Rigid Body Motion (RBM), Varied x-Impulse

Energy = 9.4096 x-impulse = -0.1766	Energy = 0.45695 x-impulse = 0.000	Energy = 4.9475 x-impulse = -2.500
		
Initial Guess	Optimal RBM $J_x = 0.0$	Optimal RBM $J_x = -2.5$

Optimal Control, Time-Morphed Geometry

Optimal Rigid Body Motion (RBM) and Time-Morphed Geometry (TMG), Varied x-Impulse

Energy = 9.4096	Energy = 0.45027	Energy = 4.6182
x-impulse = -0.1766	x-impulse = 0.000	x-impulse = -2.500



Initial Guess	Optimal RBM/TMG $J_x = 0.0$	Optimal RBM/TMG $J_x = -2.5$

Adjoint Method for Periodic PDE-Constraints

- Following identical procedure as for non-periodic case, the adjoint equations corresponding to the periodic conservation law are

$$\boldsymbol{\lambda}^{(N_t)} = \boldsymbol{\lambda}^{(0)} + \frac{\partial F}{\partial \boldsymbol{u}^{(N_t)}}^T$$

$$\boldsymbol{\lambda}^{(n-1)} = \boldsymbol{\lambda}^{(n)} + \frac{\partial F}{\partial \boldsymbol{u}^{(n-1)}}^T + \sum_{i=1}^{s} \Delta t_n \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} \left(\boldsymbol{u}_i^{(n)},\ \boldsymbol{\mu},\ t_{n-1} + c_i \Delta t_n \right)^T \boldsymbol{\kappa}_i^{(n)}$$

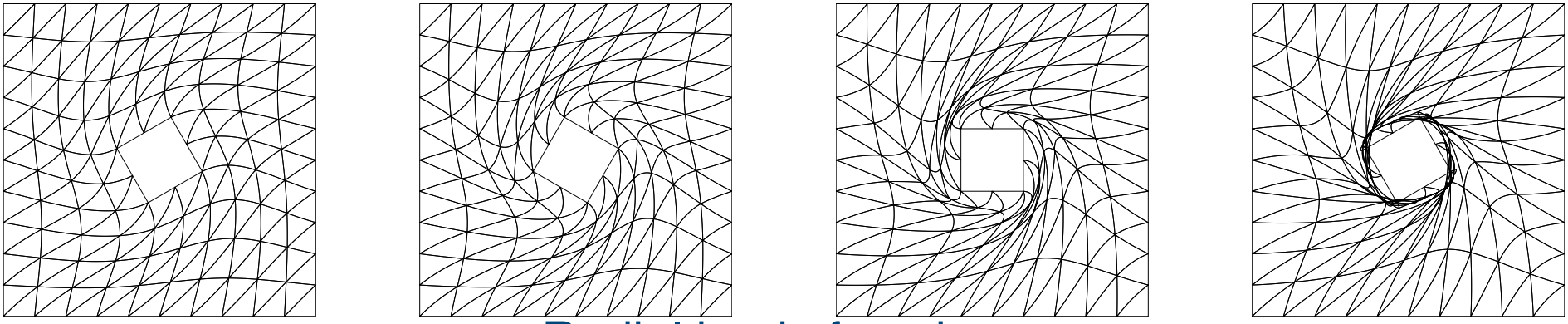
$$\mathbb{M}^T \boldsymbol{\kappa}_i^{(n)} = \frac{\partial F}{\partial \boldsymbol{u}^{(N_t)}}^T + b_i \boldsymbol{\lambda}^{(n)} + \sum_{j=i}^{s} a_{ji} \Delta t_n \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} \left(\boldsymbol{u}_j^{(n)},\ \boldsymbol{\mu},\ t_{n-1} + c_j \Delta t_n \right)^T \boldsymbol{\kappa}_j^{(n)}$$

- Dual problem is also periodic
- Solve *linear, periodic* problem using Krylov shooting method

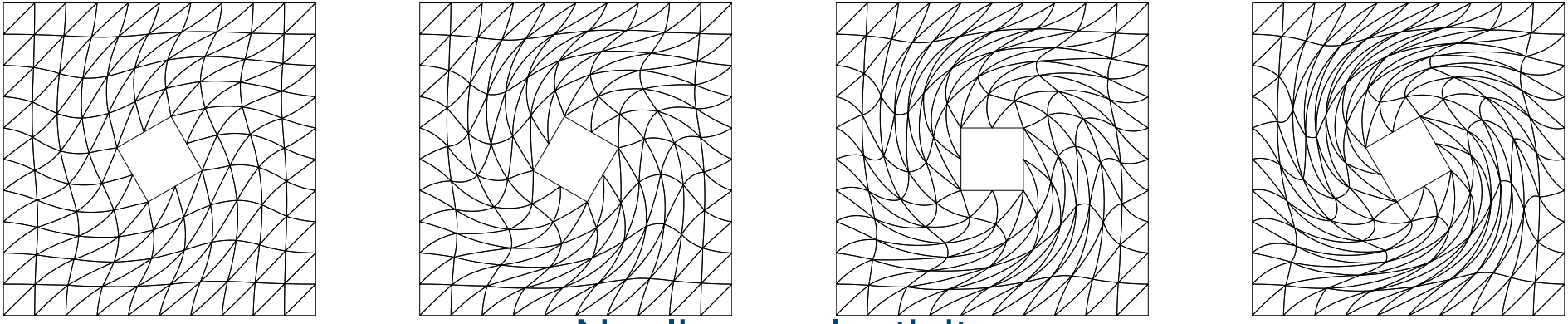
Outline

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



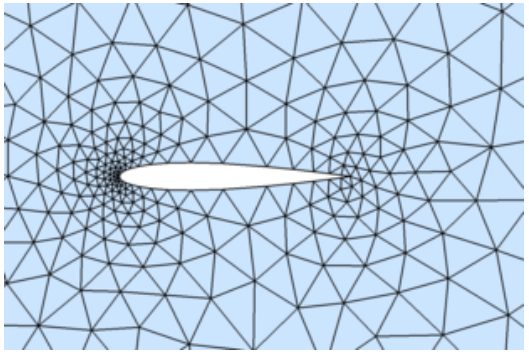
Radial basis functions

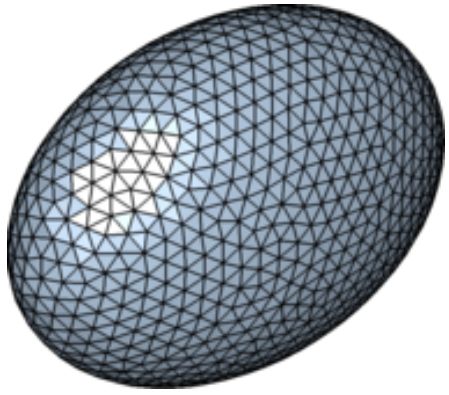


Nonlinear elasticity

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



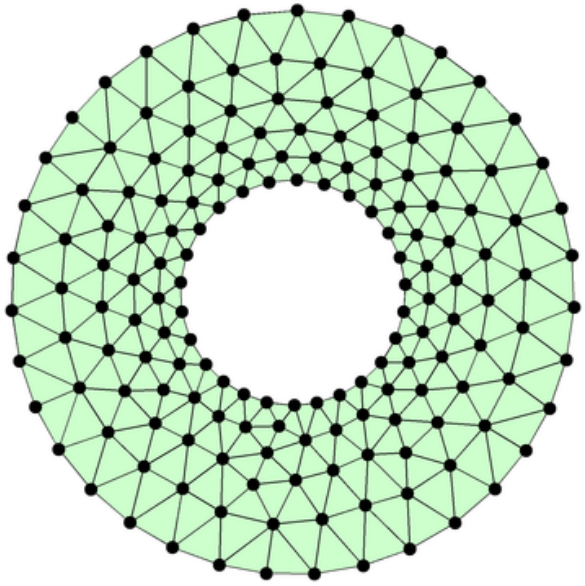


The DistMesh Mesh Generator

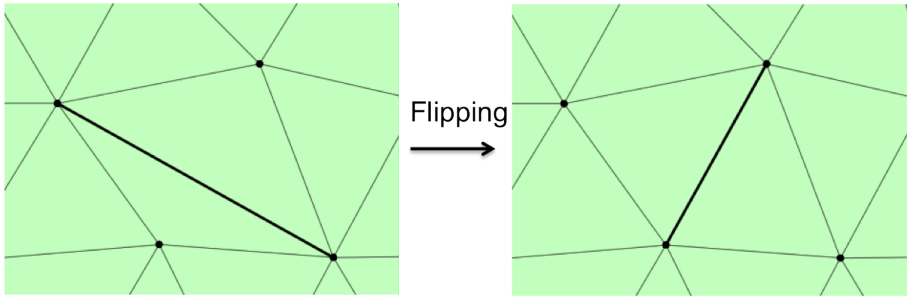
- Spring-based non-linear compressive force analogy for mesh motion

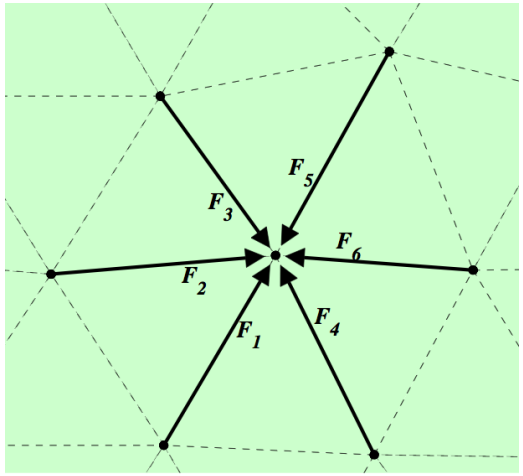
$$\boldsymbol{p}^{(n+1)} = \boldsymbol{p}^{(n)} + \delta \sum_i \boldsymbol{F}_i$$

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

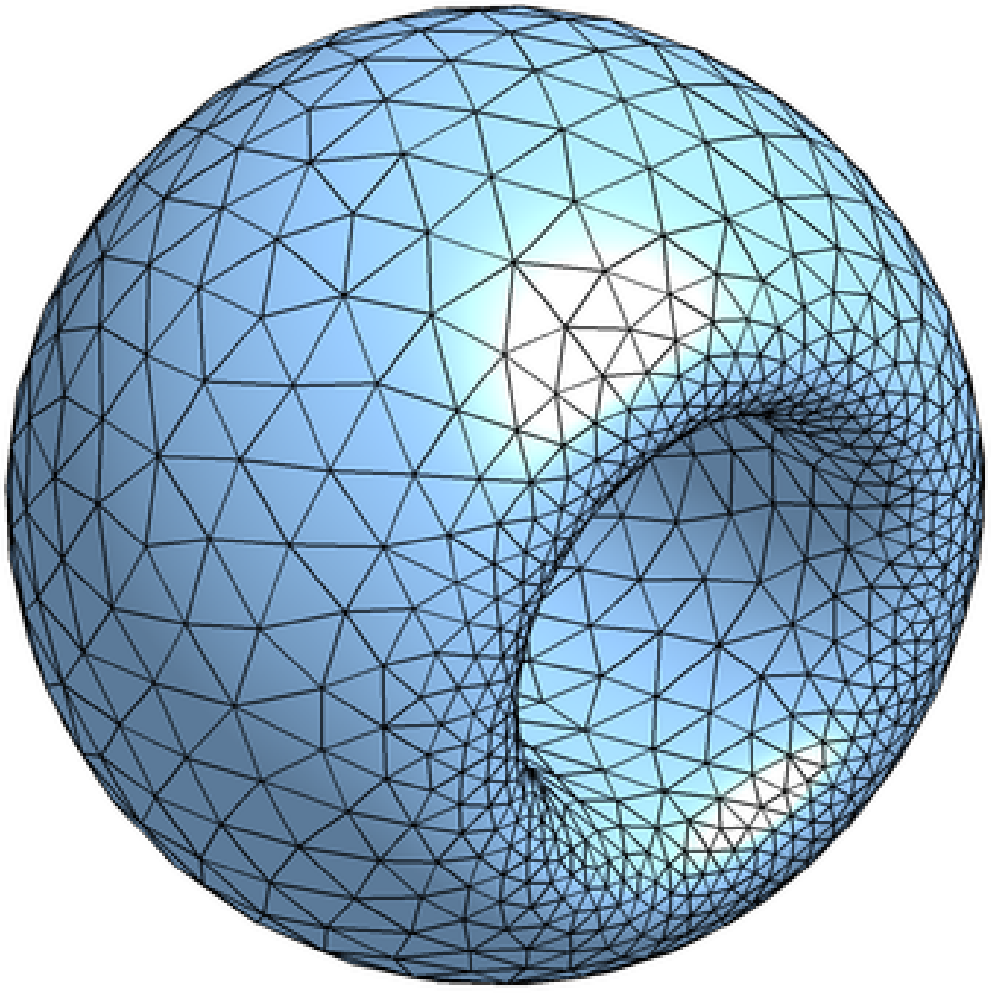


- Perform topological transformations ("edge flips") to improve element connectivities



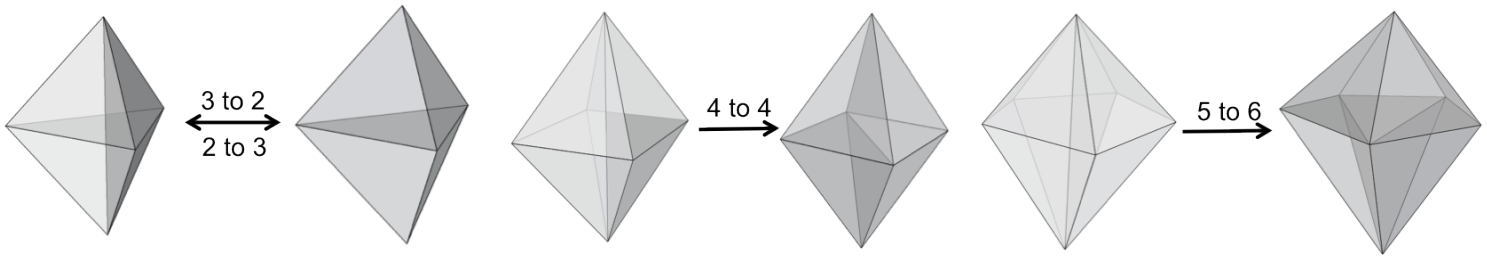


The DistMesh Mesh Generator on Surfaces

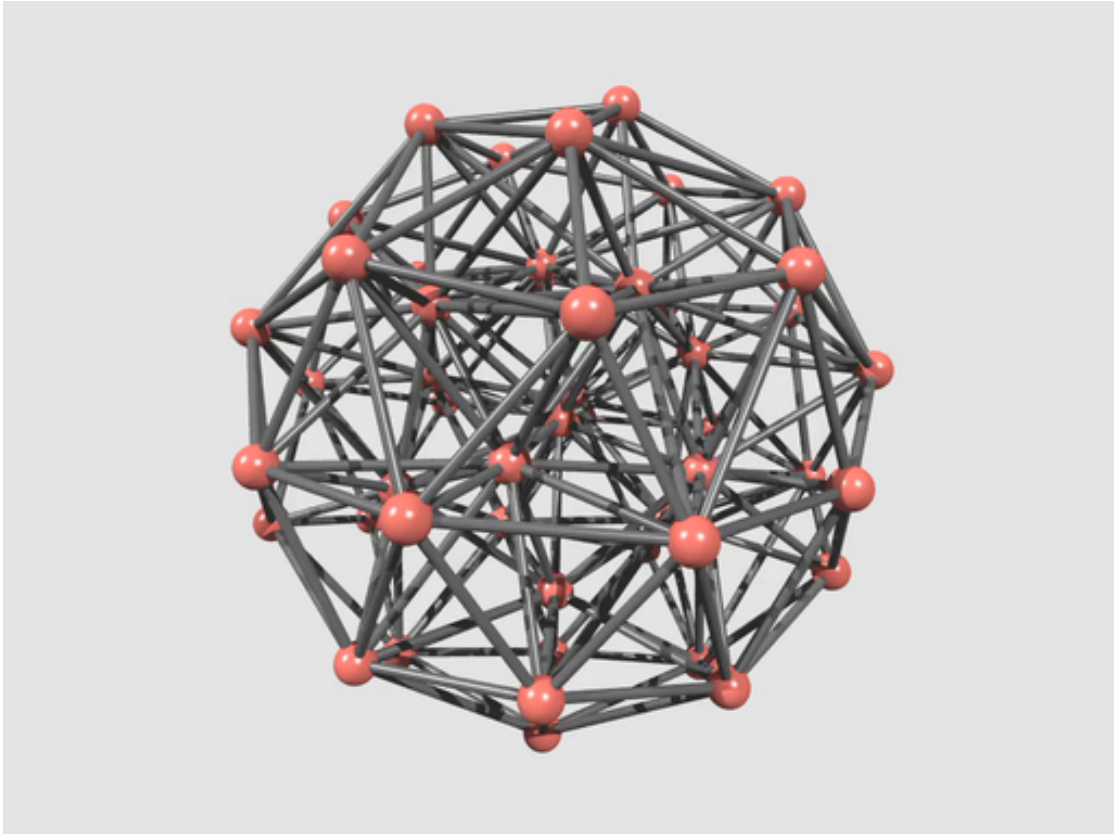


Element flips and DistMesh in 3D

- Local element flips for 3D tetrahedra:

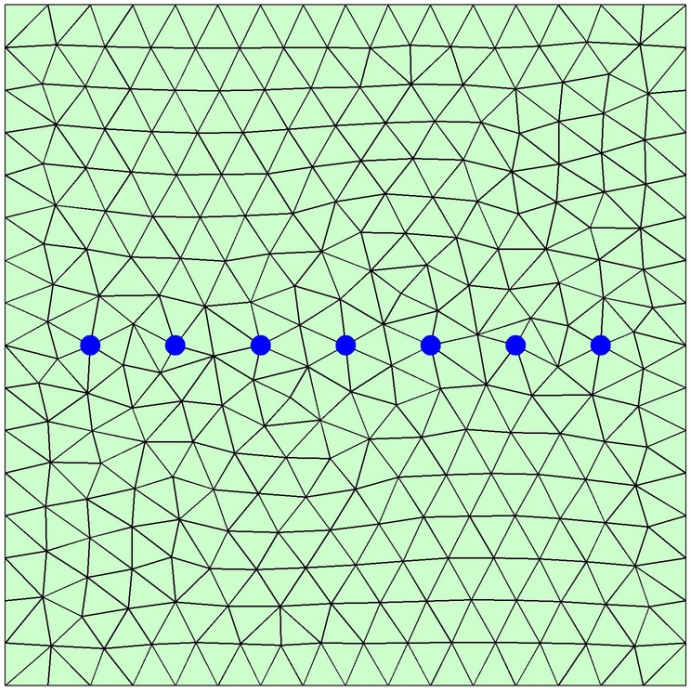


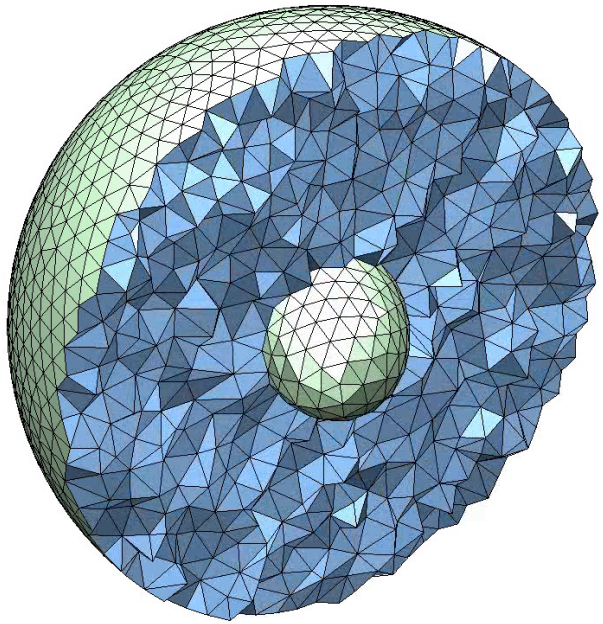
- Restricts the topology changes to a small number of elements



Moving Meshes

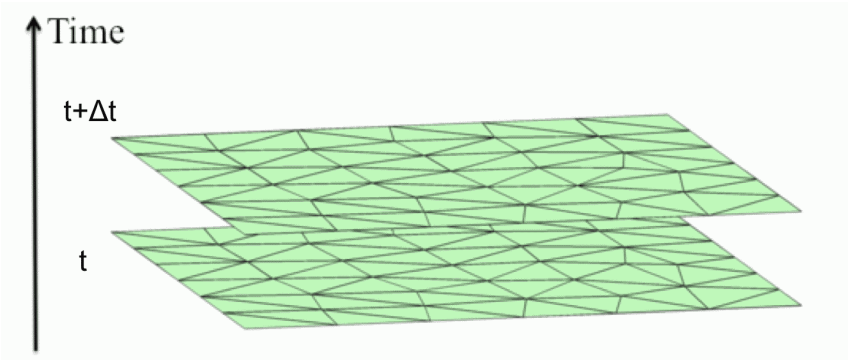
- In addition to generating high-quality initial meshes, the DistMesh algorithm is excellent for iterative generation of moving meshes
- The resulting mesh sequence involves two types of operations:
  1. Smooth node movements
  2. *Localized* element topology updates
- This allows for integration with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

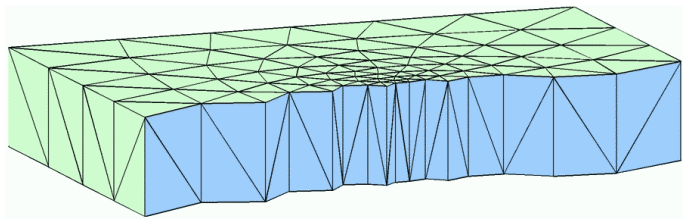




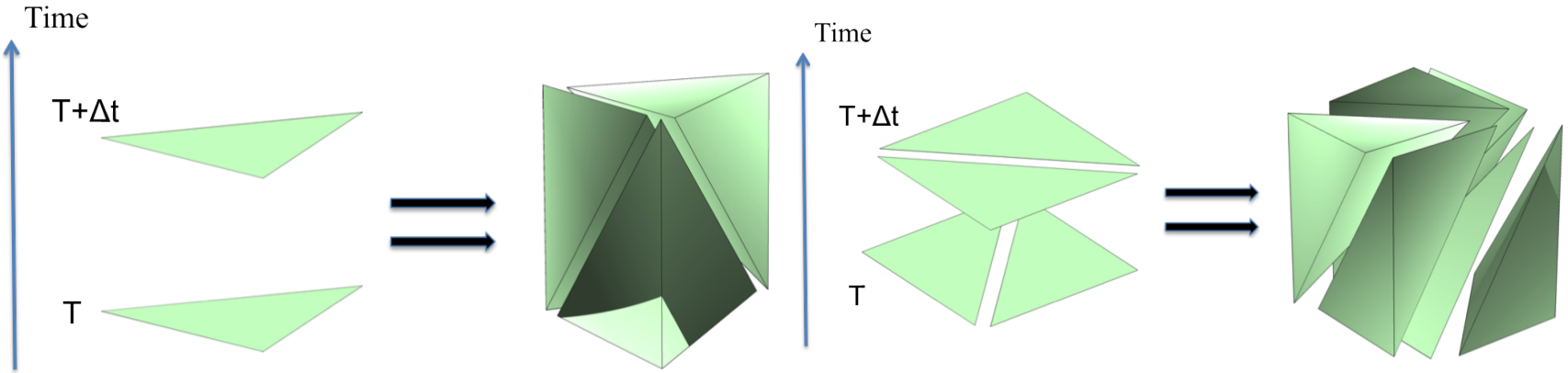
Space-time Mesh Generation

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





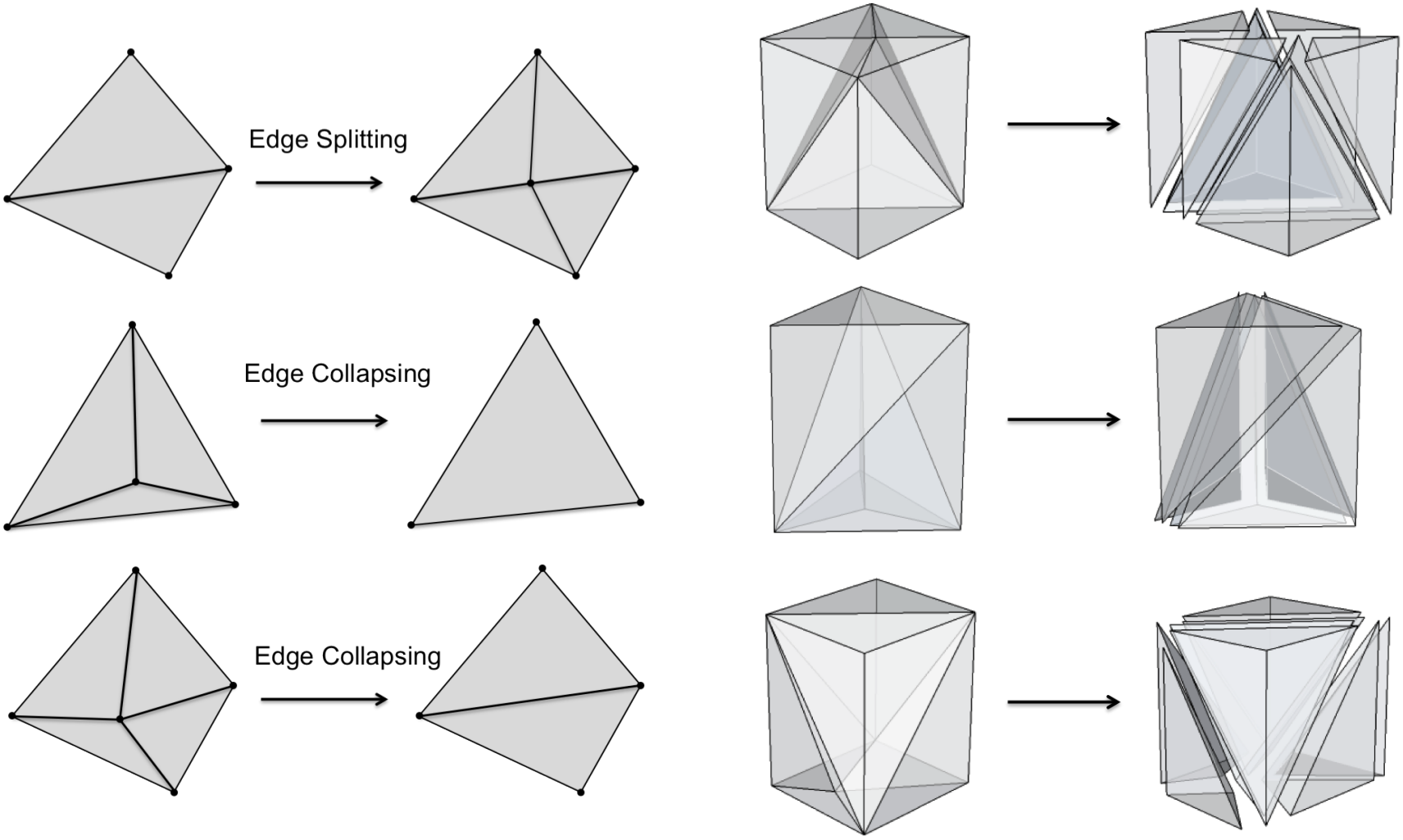
- Each local prism triangulation depends on the choice of the diagonals on the lateral faces



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

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:

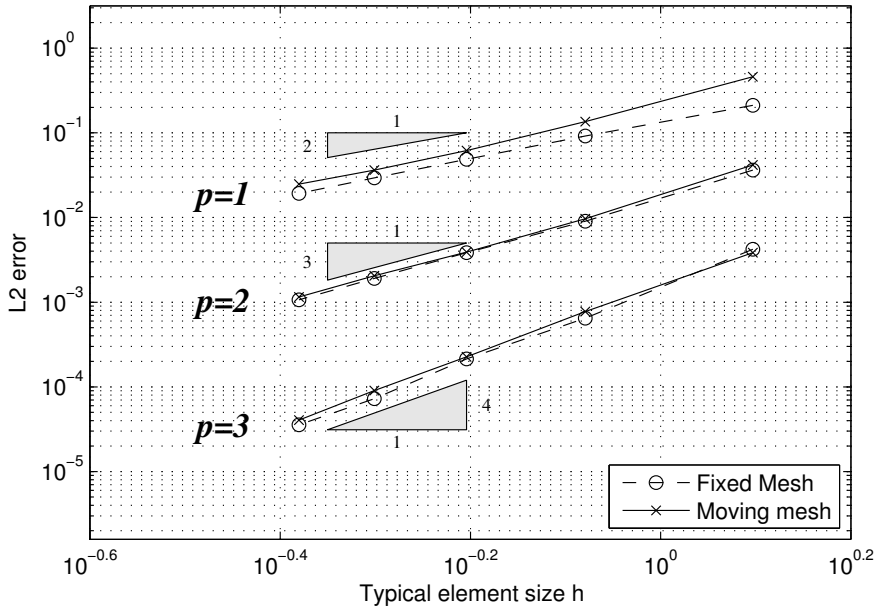
$$\mathcal{V}_T^h = \{\boldsymbol{v} \in [L^2(\Omega[0,T])]^5 \mid \boldsymbol{v}|_K \in [\mathcal{P}_p(K)]^5 \ \ \forall K \in \mathcal{T}_{[0,T]}^h\},$$
$$\Sigma_T^h = \{\boldsymbol{\sigma} \in [L^2(\Omega[0,T])]^{5\times 3} \mid \boldsymbol{\sigma}|_K \in [\mathcal{P}_p(K)]^{5\times 3} \ \ \forall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

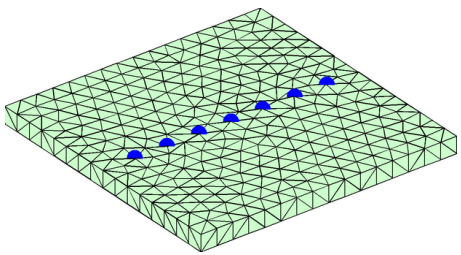
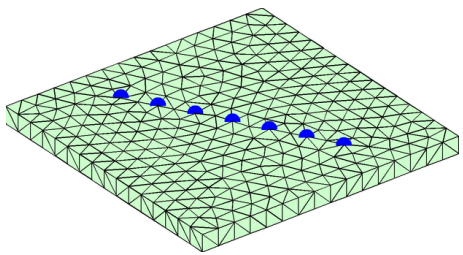
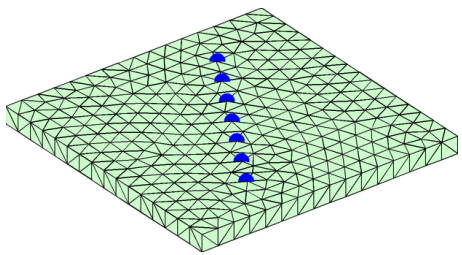
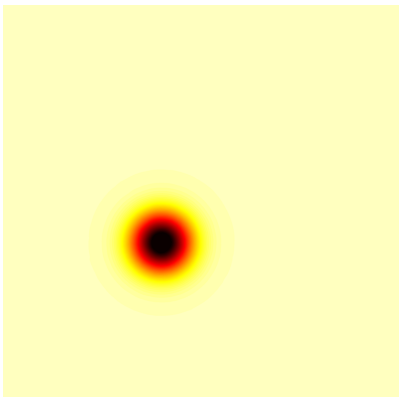
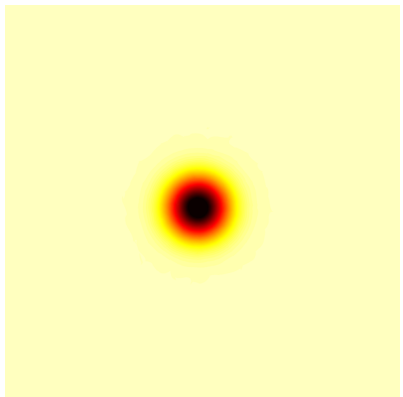
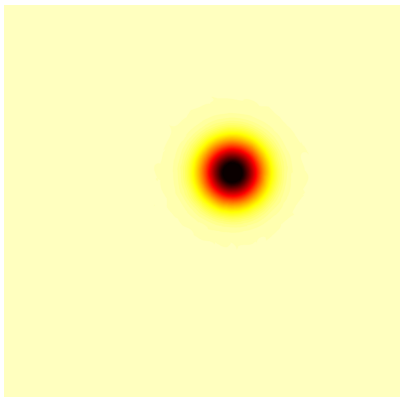
- Discretize the first-order system using a standard DG formulation on the space-time domain $\Omega[t_1, t_2]$.

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

Example: Euler Vortex, Convergence

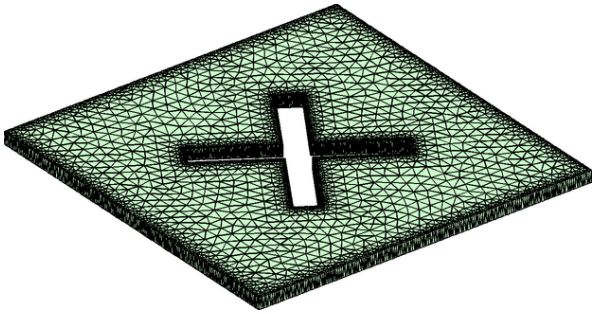
- Propagate an Euler Vortex on a fixed domain but moving mesh
- Optimal order of convergence $O(h^{p+1})$ for fixed and moving mesh.

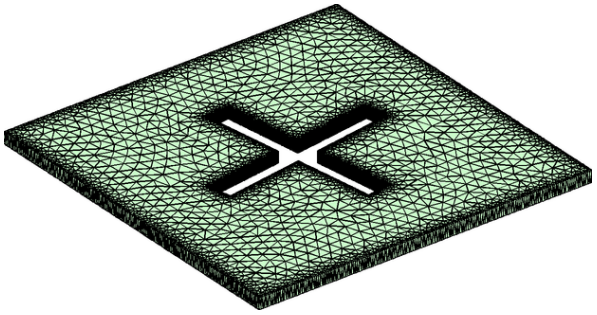


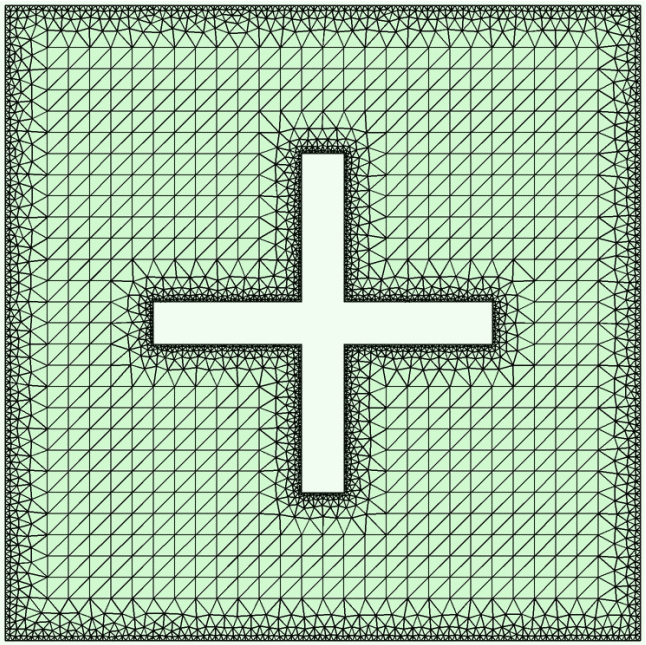







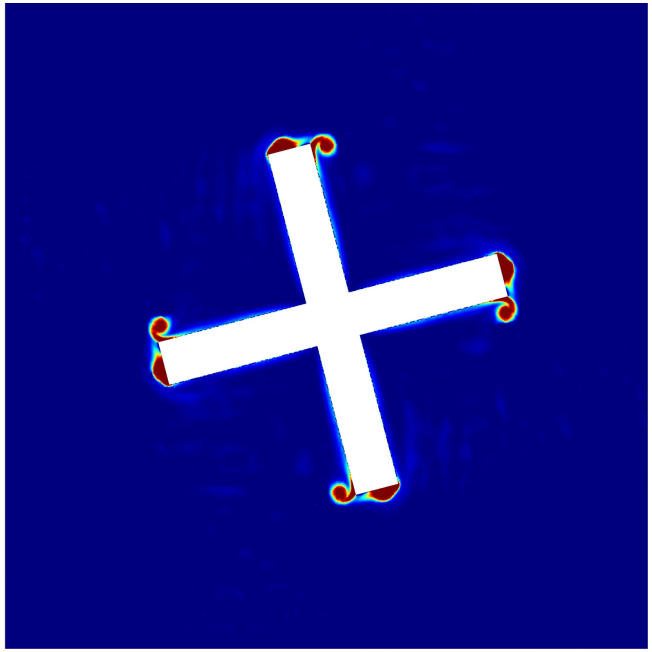
Example: Spinning Cross

- 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



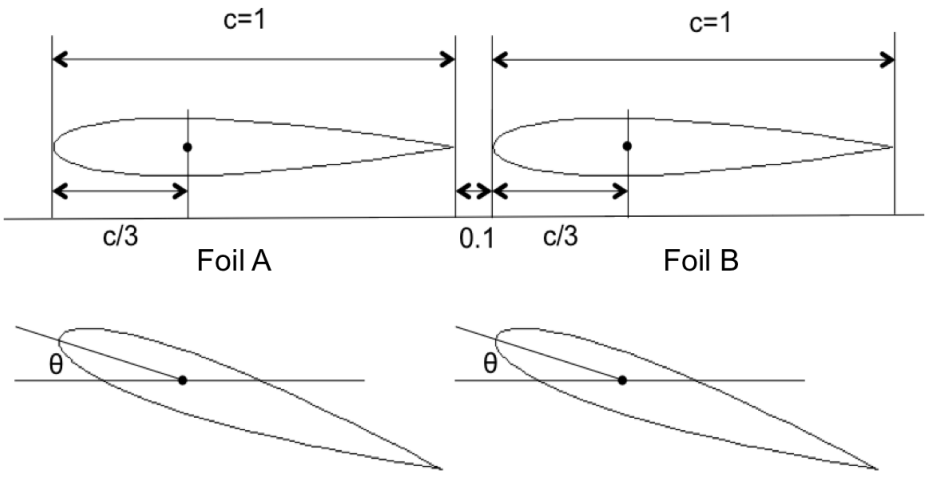


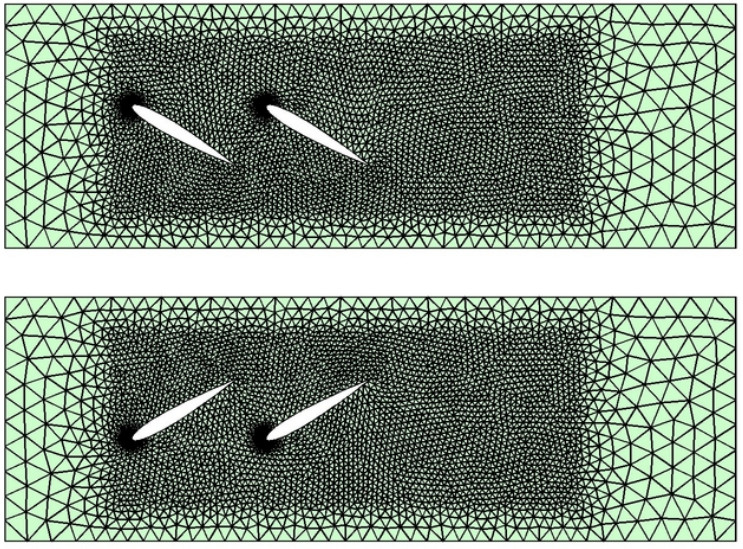




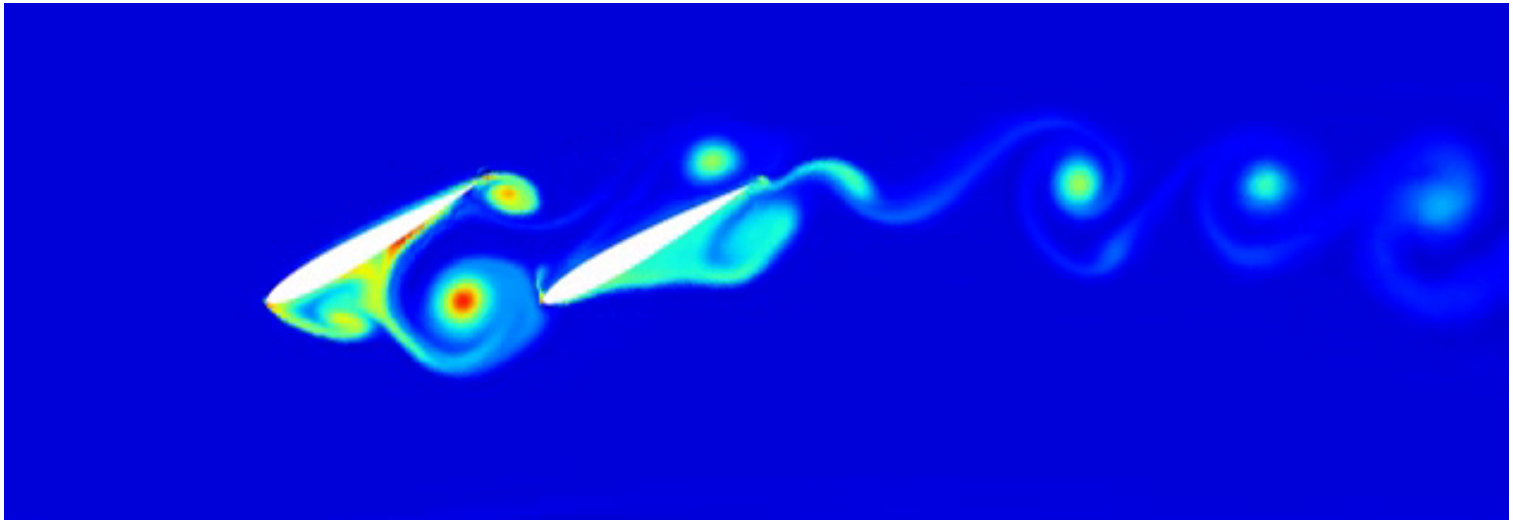
Example: Tandem Foils

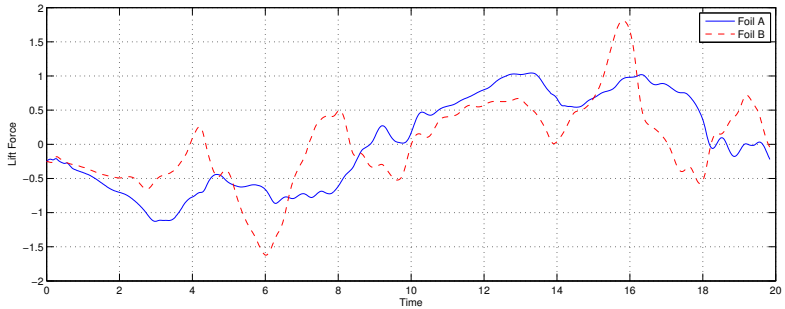
- Two foils are placed very close and rotated based on $\theta = A\sin(-2\pi ft)$ with $A = \pi/6$ and $f = 0.05$. Reynolds number 3000, Mach 0.2, polynomial degree p=2.

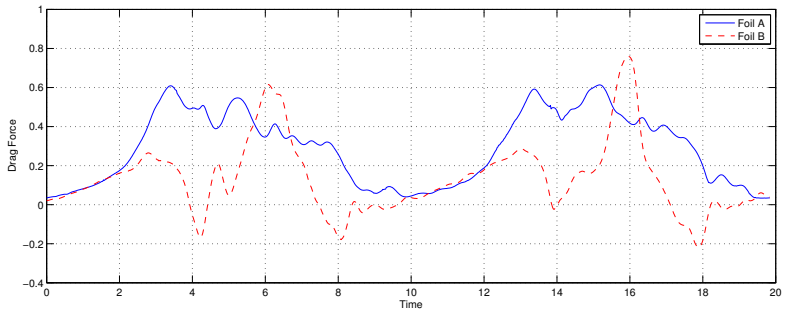




Example: Tandem Foils

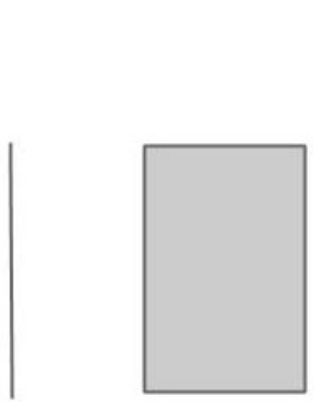




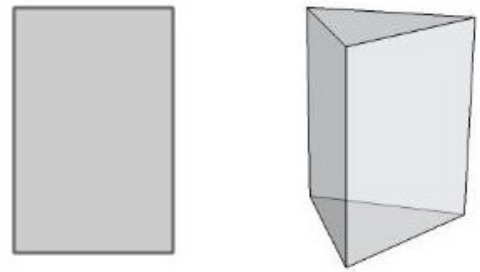


Fully unstructured 4D space-time mesh generation

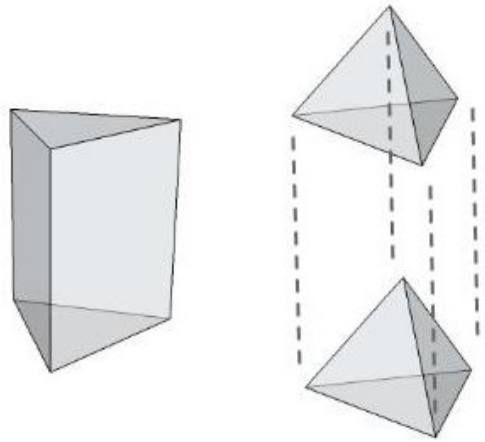
- 4D is difficult to visualize, and we extend concepts such as ‘Prisms’ and ‘Lateral Faces’ using combinatorial notions



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

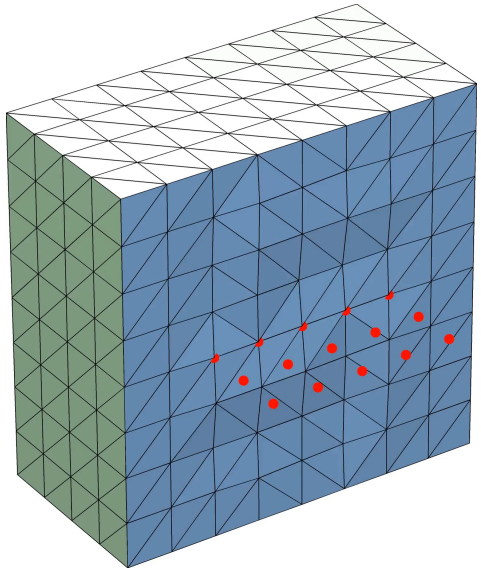
- The following result can be shown for obtaining valid simplex triangulations of a simple 4D prism:

Theorem

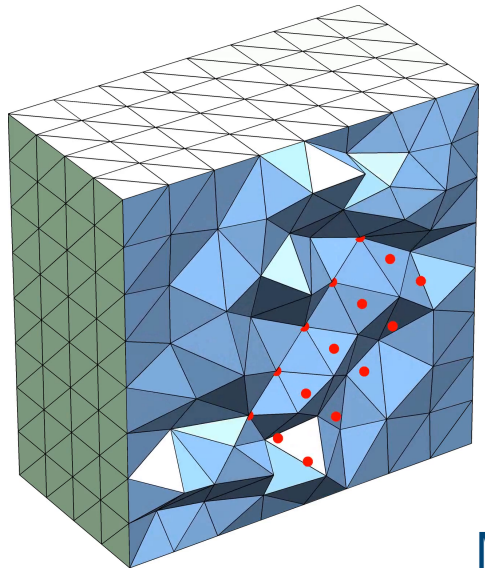
If the 4D prism mesh is constructed purely by simple 4D-prisms, the indexing approach can always triangulate the prism mesh into a valid simplex mesh. More precisely, in each simple 4D-prism, if we have the ordered vertices $\{p_{(1)}^{V,t}, p_{(2)}^{V,t}, p_{(3)}^{V,t}, p_{(4)}^{V,t}\}$ *with* $I_{(1)}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

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

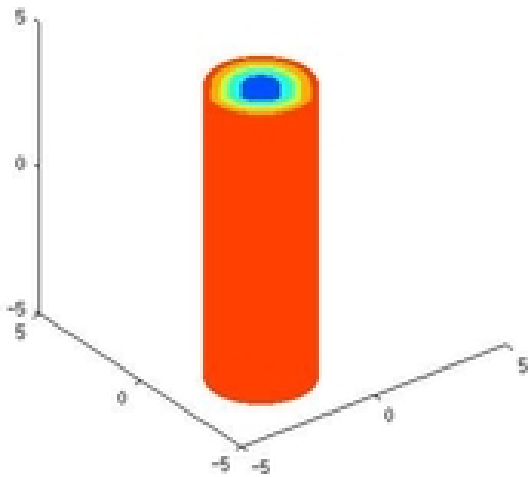
Extruded 3D Euler vortex convergence test



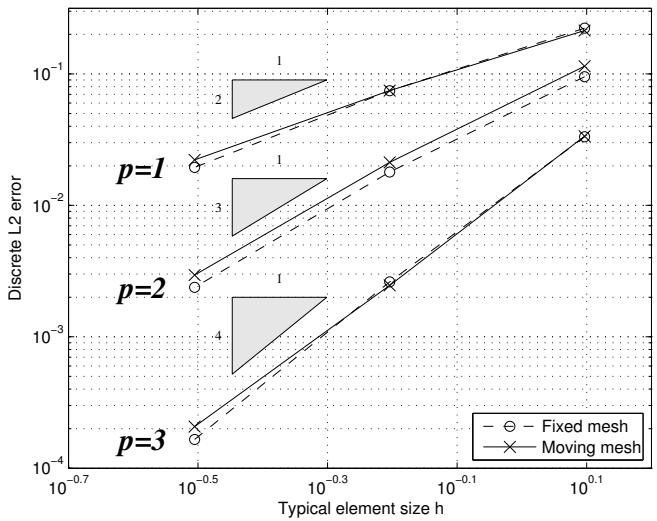
Initial mesh at $t = 0$



Moving mesh at



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