

High-Order Methods on Unstructured Meshes for Fluid and Solid Mechanics

Per-Olof Persson

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

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

Argyris Lecture
SimTech, University of Stuttgart

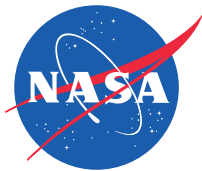


November 27, 2024



Acknowledgements

This work has been funded by multiple organizations for 15 years



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



Outline

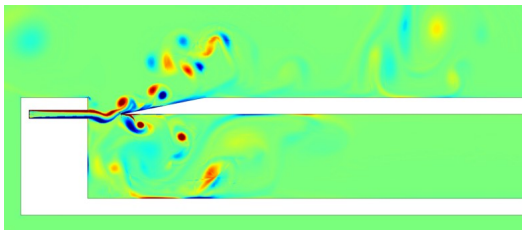
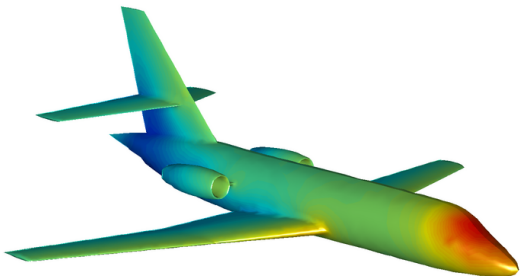
- 1 Introduction and Motivation
- 2 Unstructured Mesh Generation
 - The DistMesh Mesh Generator
 - Deep Reinforcement learning for Mesh Improvement
- 3 High-Order Methods for solving PDEs
 - The Discontinuous Galerkin Method
 - Sparse Discretizations and Efficient Solvers
- 4 Methods for Deforming Domains
 - Multiphysics and Fluid-Structure Interaction
 - Unstructured Mesh Space-Time Methods
- 5 PDE-Constrained Optimization
 - Optimization-Based High-Order Shock Tracking

Outline

- 1 Introduction and Motivation
- 2 Unstructured Mesh Generation
 - The DistMesh Mesh Generator
 - Deep Reinforcement learning for Mesh Improvement
- 3 High-Order Methods for solving PDEs
 - The Discontinuous Galerkin Method
 - Sparse Discretizations and Efficient Solvers
- 4 Methods for Deforming Domains
 - Multiphysics and Fluid-Structure Interaction
 - Unstructured Mesh Space-Time Methods
- 5 PDE-Constrained Optimization
 - Optimization-Based High-Order Shock Tracking

Numerical Simulations

- Constructing mathematical models and numerical solution techniques and using computers to analyze and solve scientific and engineering problems
- Numerical simulations help reconstruct and understand known events, or predict future situations
- Highly interdisciplinary field – Mathematics, computer science, physics, engineering



Modeling using Partial Differential Equations

- Almost all physical phenomena can be described by PDEs
- Example (Fluid dynamics): The Navier-Stokes equations

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{f}, \quad \nabla \cdot \mathbf{u} = 0$$

- Example (Elastodynamics): Navier's equations

$$\nabla \cdot \boldsymbol{\sigma}^T + \mathbf{b} = \rho \frac{\partial^2 \mathbf{u}}{\partial t^2}, \quad \boldsymbol{\varepsilon} = \frac{1}{2} (\nabla \mathbf{u} + \nabla \mathbf{u}^T), \quad \boldsymbol{\sigma} = 2\mu \boldsymbol{\varepsilon} + \lambda \text{tr}(\boldsymbol{\varepsilon}) \mathbf{1}.$$

- Example (Electromagnetics): Maxwell's equations

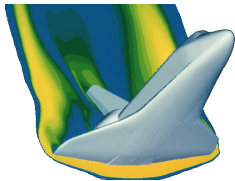
$$\begin{aligned} \nabla \cdot \mathbf{E} &= \frac{\rho}{\varepsilon_0} & \nabla \cdot \mathbf{B} &= 0 \\ \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} & \nabla \times \mathbf{B} &= \mu_0 \mathbf{J} + \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} \end{aligned}$$

- Also need to specify the physical region, boundary conditions, material parameters

Application Areas

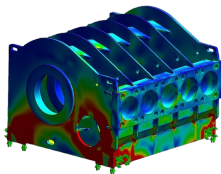
- **Computational Fluid Dynamics (CFD)**

- **Aerospace:** Optimizing aircraft aerodynamics.
- **Environment:** Simulating weather & pollutant dispersion.
- **Biomedical:** Modeling blood and airflow in the body.



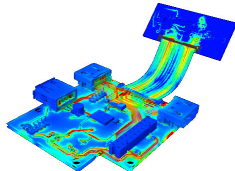
- **Elasticity**

- **Civil Engineering:** Stress analysis in structures.
- **Material Science:** Evaluating mechanical properties.
- **Automotive:** Crashworthiness and tire design.



- **Electromagnetics**

- **Telecom:** Designing antennas and 5G systems.
- **Power Systems:** Electric field analysis in transformers.
- **Electronics:** Minimizing electromagnetic interference.

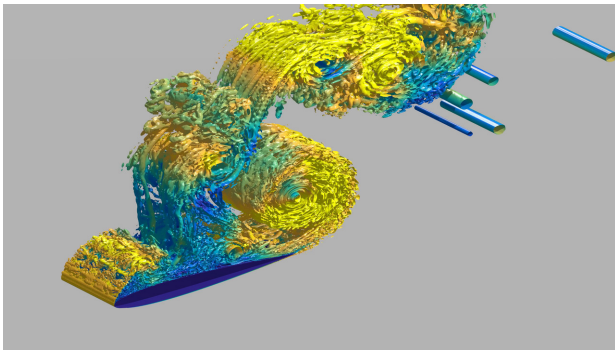
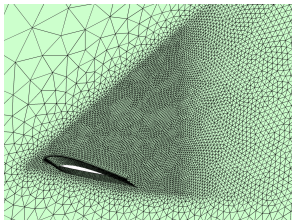


The Numerical Simulation Process

- Mathematical modeling
 - Define regions, PDE-based models, boundary conditions, etc
- Geometry modeling
 - Describe the regions using Computer Aided Design (CAD)
- **Mesh generation**
 - Discretize the regions using simple shapes such as triangles and tetrahedra
- **Numerical solution**
 - Approximate PDEs numerically and solve using efficient algorithms
- Post-processing/Visualization
 - Compute desired quantities, generate images and animations that illustrate the solution
- *All components used repeatedly in the optimization process*

Real-world applications: Large Eddy Simulation

- Turbulent flow problems are inherently difficult due to interactions between scales
- Large Eddy Simulation (LES) and high-order methods are widely believed to be part of the future state-of-the-art simulation tools
- Flow over an SD7003 airfoil at Reynolds number 100,000 and 30° angle of attack

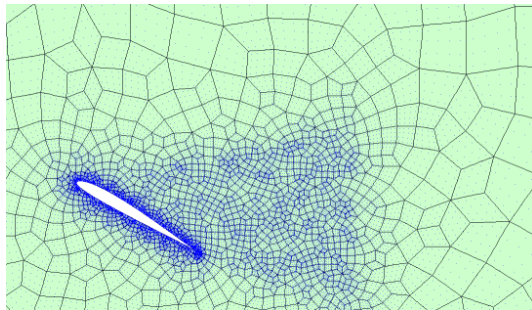
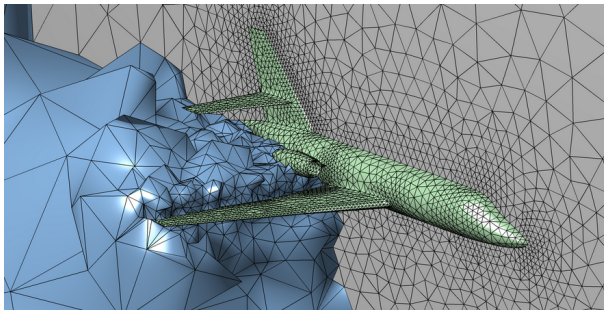


Outline

- 1 Introduction and Motivation
- 2 Unstructured Mesh Generation**
 - The DistMesh Mesh Generator
 - Deep Reinforcement learning for Mesh Improvement
- 3 High-Order Methods for solving PDEs
 - The Discontinuous Galerkin Method
 - Sparse Discretizations and Efficient Solvers
- 4 Methods for Deforming Domains
 - Multiphysics and Fluid-Structure Interaction
 - Unstructured Mesh Space-Time Methods
- 5 PDE-Constrained Optimization
 - Optimization-Based High-Order Shock Tracking

Why Unstructured Meshes?

- Complex *geometries* need flexible element topologies
- Complex *solution fields* need spatially variable resolution
- Automated mesh generators for CAD geometries produce unstructured elements
- Most real-world simulations use numerical methods for unstructured elements



Outline

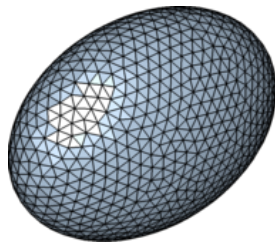
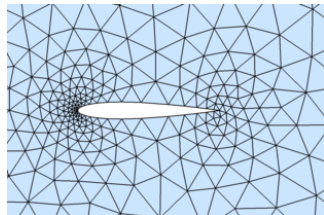
- 1 Introduction and Motivation
- 2 **Unstructured Mesh Generation**
 - **The DistMesh Mesh Generator**
 - Deep Reinforcement learning for Mesh Improvement
- 3 High-Order Methods for solving PDEs
 - The Discontinuous Galerkin Method
 - Sparse Discretizations and Efficient Solvers
- 4 Methods for Deforming Domains
 - Multiphysics and Fluid-Structure Interaction
 - Unstructured Mesh Space-Time Methods
- 5 PDE-Constrained Optimization
 - Optimization-Based High-Order Shock Tracking

The DistMesh Mesh Generator

- The *DistMesh* algorithm [Persson, Ph.D. thesis, '05]
- Spring-based non-linear force analogy for mesh motion

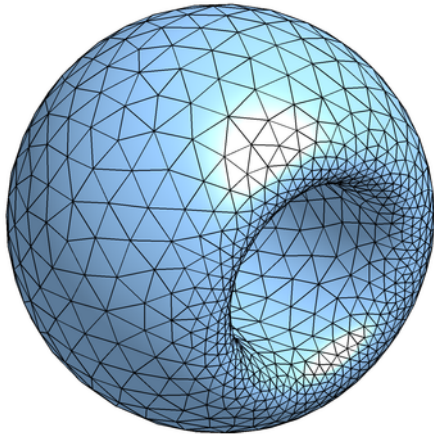
$$\mathbf{p}^{(n+1)} = \mathbf{p}^{(n)} + \delta \sum_i \mathbf{F}_i, \quad |\mathbf{F}_i(l)| = \begin{cases} k(l - l_0) & \text{if } l \geq l_0, \\ 0 & \text{if } l < l_0, \end{cases}$$

- Project boundary nodes using *implicit geometry* $\phi(\mathbf{x})$
- Local element connectivity operations
- Important 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, Julia, Mathematica, Octave



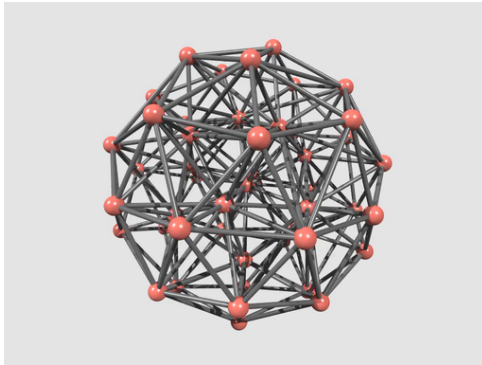
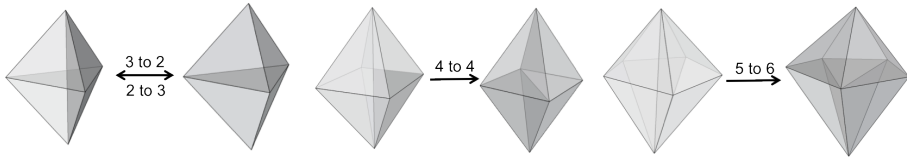
Live Mesh Demo

The DistMesh Mesh Generator on Surfaces



Element flips and DistMesh in 3D

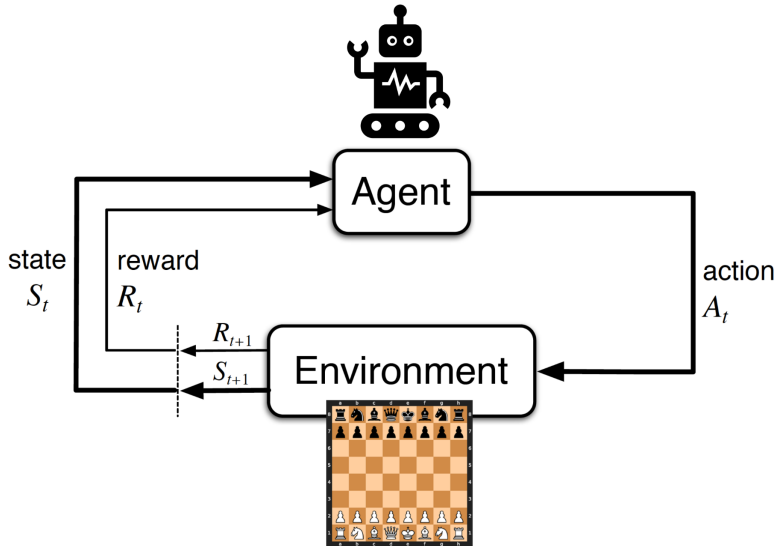
- Local element flips for 3D tetrahedra:



Outline

- 1 Introduction and Motivation
- 2 Unstructured Mesh Generation**
 - The DistMesh Mesh Generator
 - Deep Reinforcement learning for Mesh Improvement**
- 3 High-Order Methods for solving PDEs
 - The Discontinuous Galerkin Method
 - Sparse Discretizations and Efficient Solvers
- 4 Methods for Deforming Domains
 - Multiphysics and Fluid-Structure Interaction
 - Unstructured Mesh Space-Time Methods
- 5 PDE-Constrained Optimization
 - Optimization-Based High-Order Shock Tracking

Basic idea of reinforcement learning



Reinforcement Learning, Solutions Methods

Finite state-space



Finite action-space

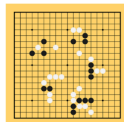


Tabular methods

Iterative methods with
provable convergence



10^{45} states



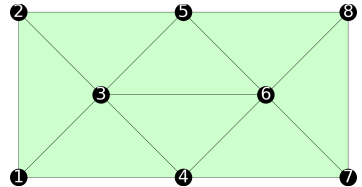
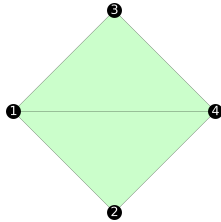
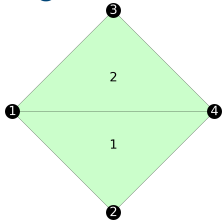
10^{170} states

Sampling based methods

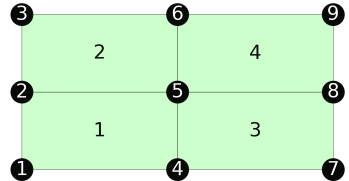
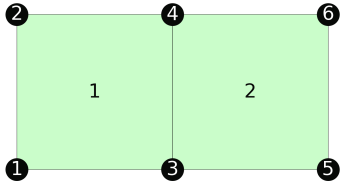
- Monte Carlo Tree Search
- Deep RL

Local mesh editing operations

Triangles:



Quadrilaterals:



Objective: minimize vertex irregularity

Given:

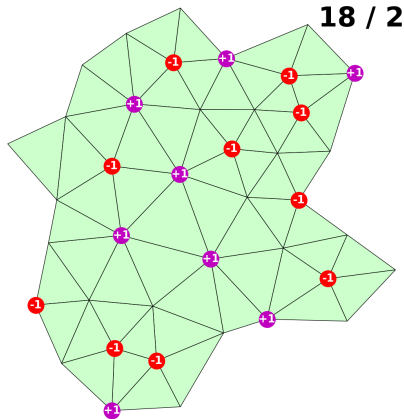
- Mesh m
- Desired degree of vertices d^*
- Define $\Delta_i = d_i - d_i^*$

minimize $s = \sum_i |\Delta_i|$

Note that:

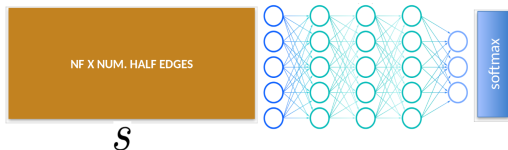
- $s^* = \left| \sum_i \Delta_i \right| \leq \sum_i |\Delta_i| = s$
- s^* invariant under mesh edits

$\implies s^*$ is a bound on the best possible improved mesh $\implies$ use for a normalized optimality score.

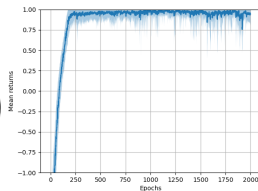


Deep Reinforcement Learning for Block Meshing

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



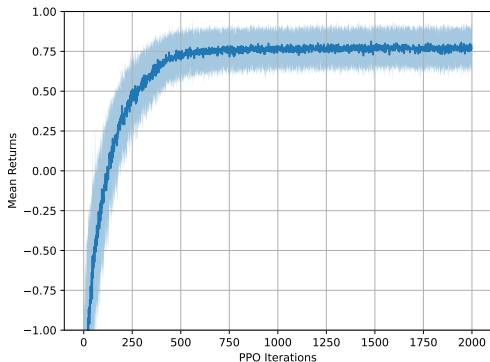
$$\pi(a|s)$$



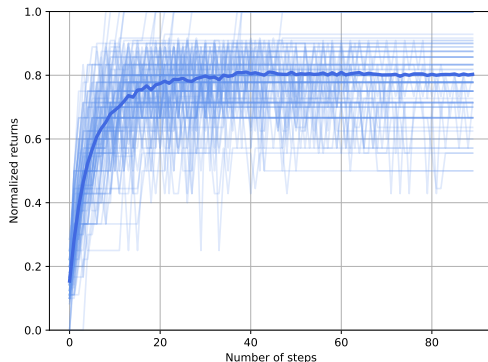
- Narayanan, Pan, Persson. *Learning topological operations on meshes with application to block decomposition of polygons*. Computer-Aided Design, Vol. 175, pp. 103744 (2024).

- Narayanan, *Machine Learning Methods to Optimize the Geometry and Topology of Meshes*. Ph.D. thesis, University of California, Berkeley, May 2024.

Results: Triangular Meshes



Average performance over training history

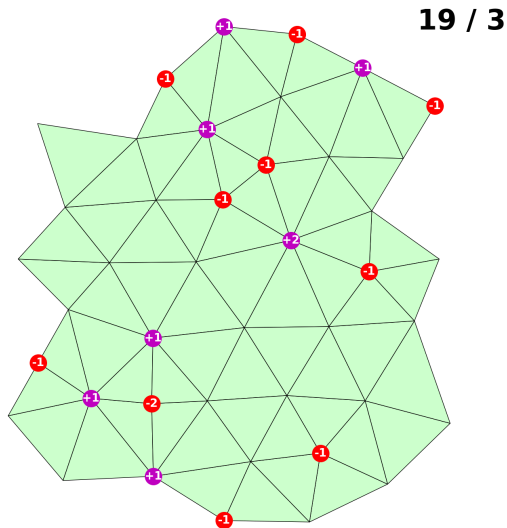


Evaluating the trained agent on multiple rollouts

Performance of the triangle mesh agent over the training history.

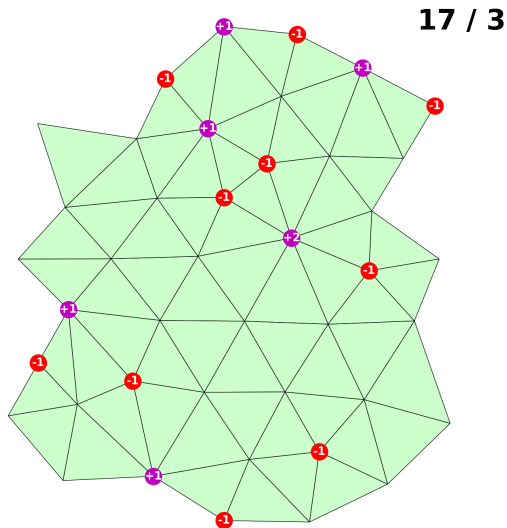
Results: Triangular Meshing

Triangular meshing
Example 1
Step 0 (out of 27)



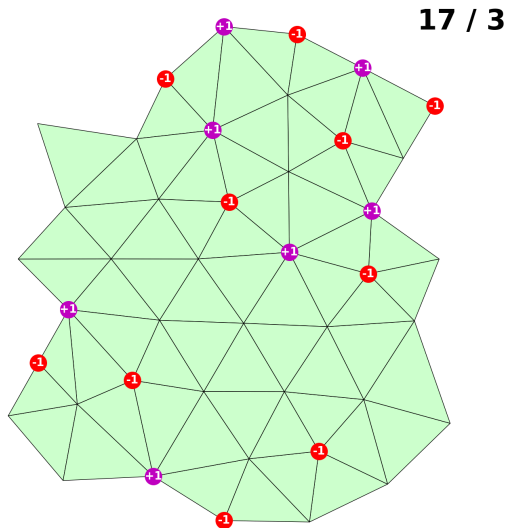
Results: Triangular Meshing

Triangular meshing
Example 1
Step 1 (out of 27)



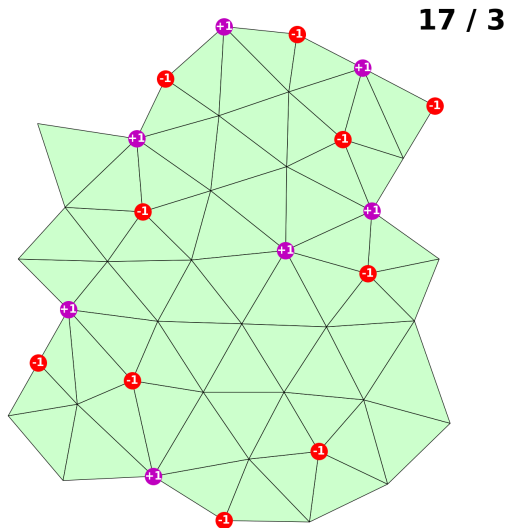
Results: Triangular Meshing

Triangular meshing
Example 1
Step 2 (out of 27)



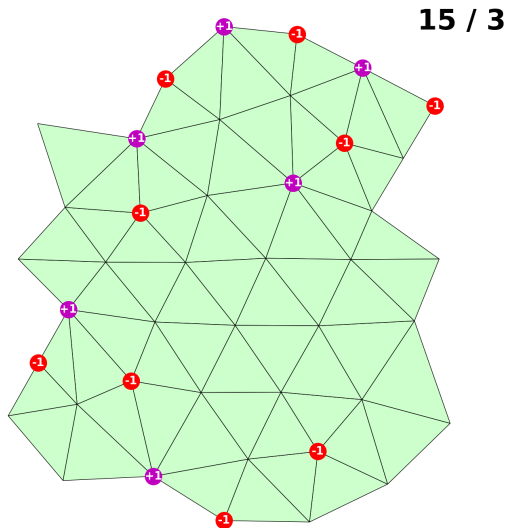
Results: Triangular Meshing

Triangular meshing
Example 1
Step 3 (out of 27)



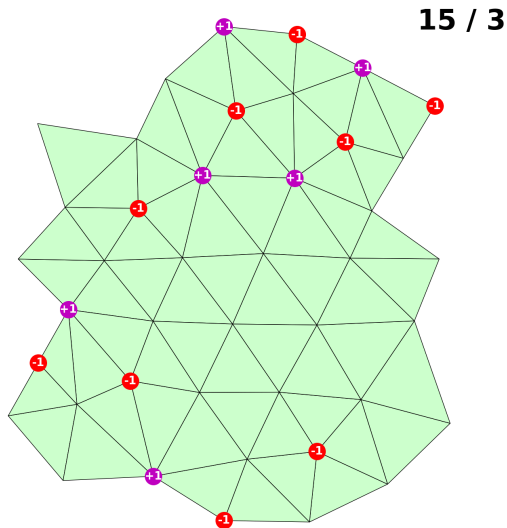
Results: Triangular Meshing

Triangular meshing
Example 1
Step 4 (out of 27)



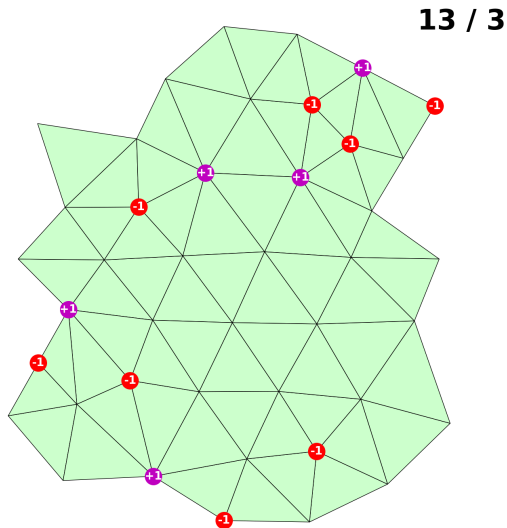
Results: Triangular Meshing

Triangular meshing
Example 1
Step 5 (out of 27)



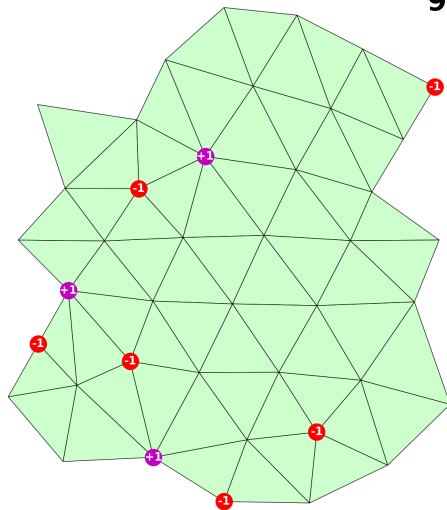
Results: Triangular Meshing

Triangular meshing
Example 1
Step 6 (out of 27)



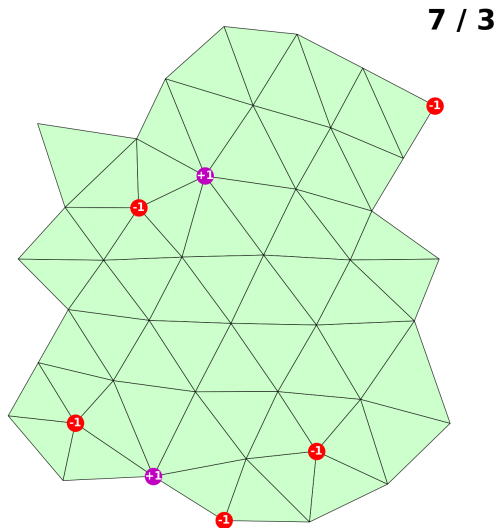
Results: Triangular Meshing

Triangular meshing
Example 1
Step 7 (out of 27)



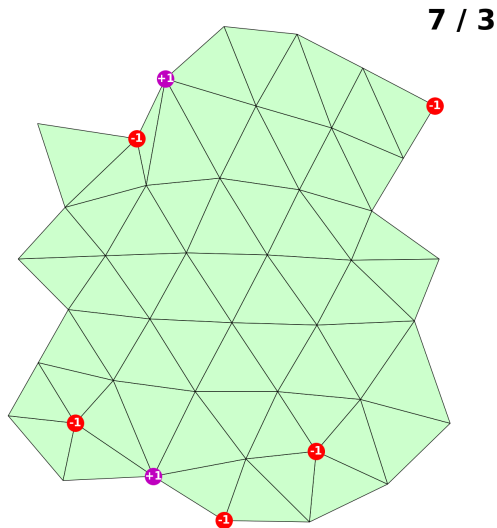
Results: Triangular Meshing

Triangular meshing
Example 1
Step 8 (out of 27)



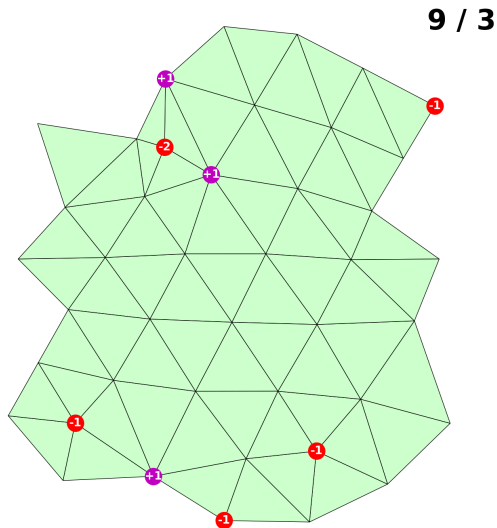
Results: Triangular Meshing

Triangular meshing
Example 1
Step 9 (out of 27)



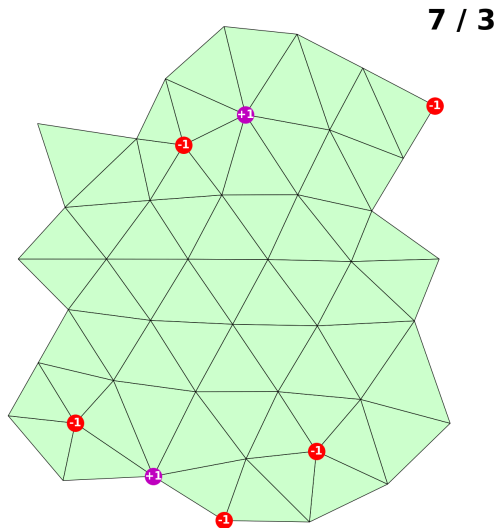
Results: Triangular Meshing

Triangular meshing
Example 1
Step 10 (out of 27)



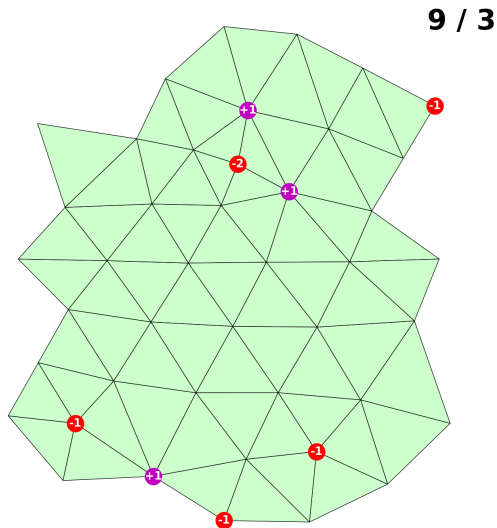
Results: Triangular Meshing

Triangular meshing
Example 1
Step 11 (out of 27)



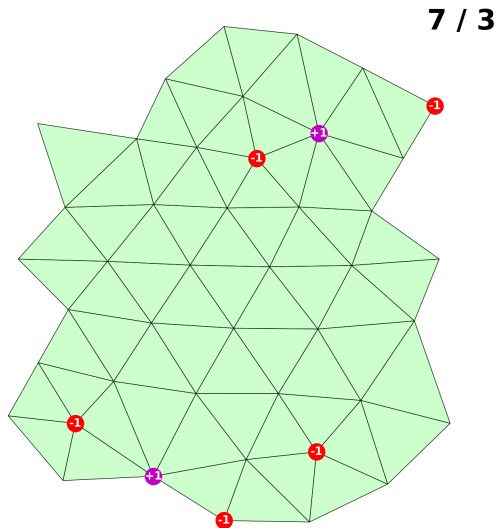
Results: Triangular Meshing

Triangular meshing
Example 1
Step 12 (out of 27)



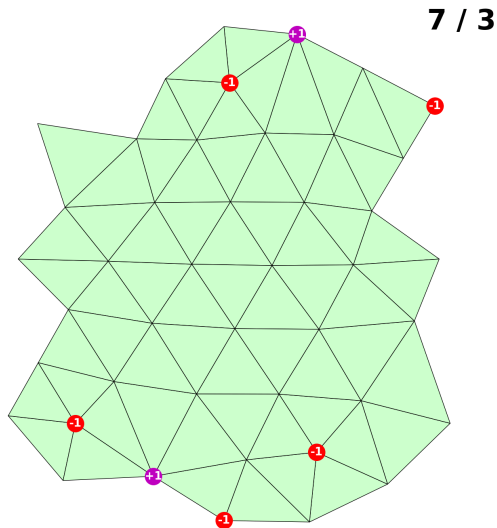
Results: Triangular Meshing

Triangular meshing
Example 1
Step 13 (out of 27)



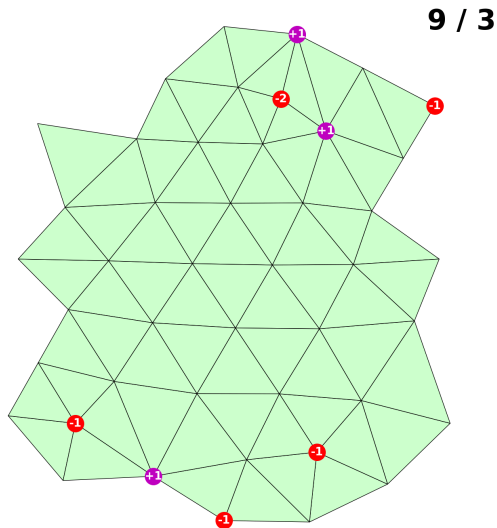
Results: Triangular Meshing

Triangular meshing
Example 1
Step 14 (out of 27)



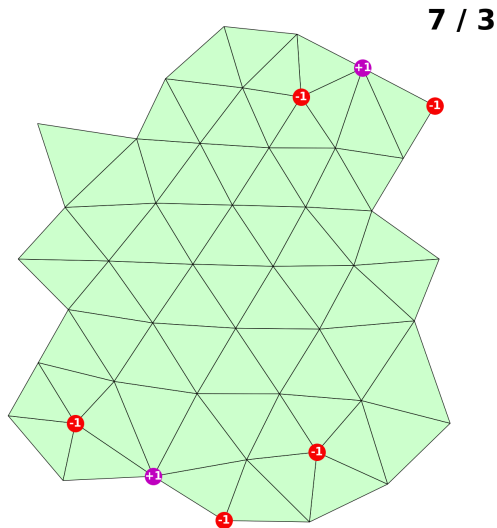
Results: Triangular Meshing

Triangular meshing
Example 1
Step 15 (out of 27)



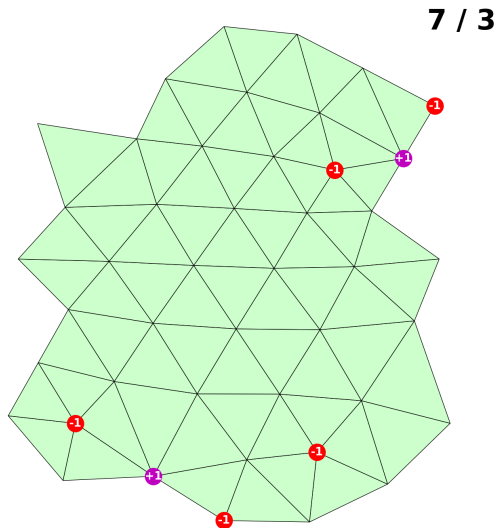
Results: Triangular Meshing

Triangular meshing
Example 1
Step 16 (out of 27)



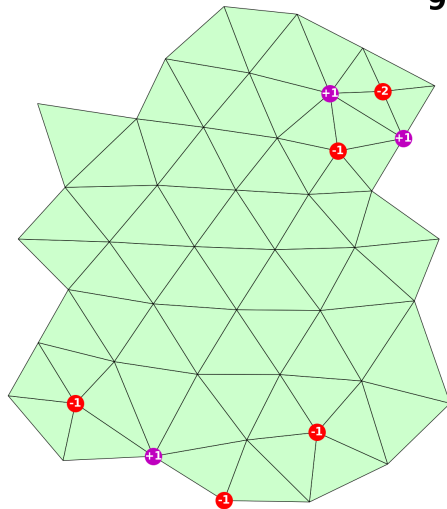
Results: Triangular Meshing

Triangular meshing
Example 1
Step 17 (out of 27)



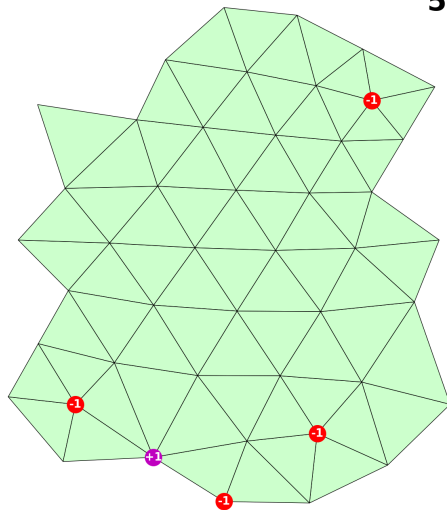
Results: Triangular Meshing

Triangular meshing
Example 1
Step 18 (out of 27)



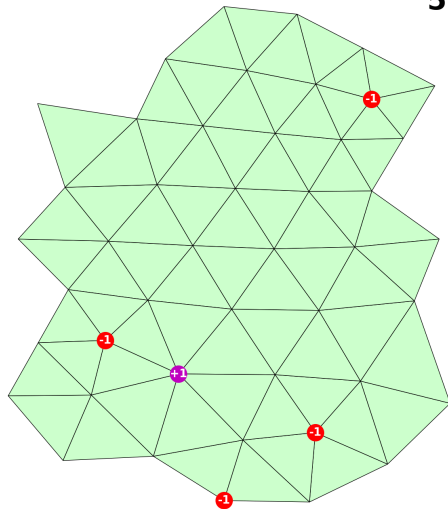
Results: Triangular Meshing

Triangular meshing
Example 1
Step 19 (out of 27)



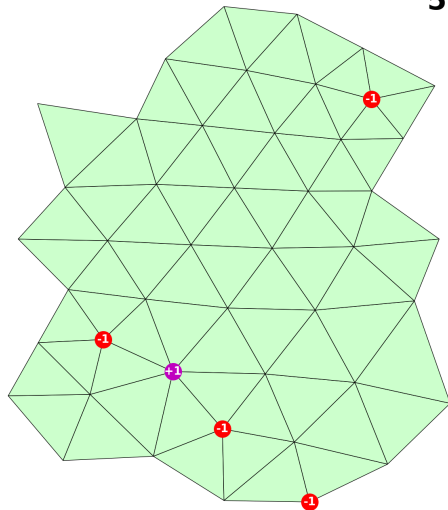
Results: Triangular Meshing

Triangular meshing
Example 1
Step 20 (out of 27)



Results: Triangular Meshing

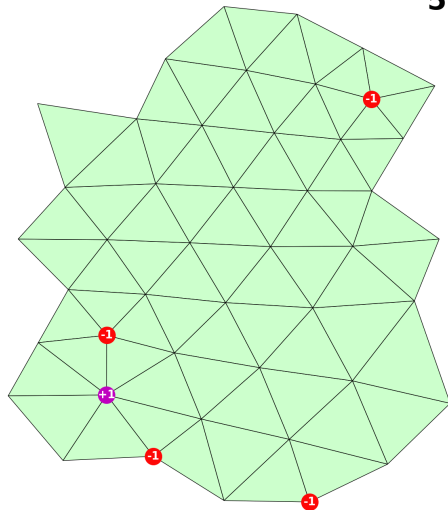
Triangular meshing
Example 1
Step 21 (out of 27)



5 / 3

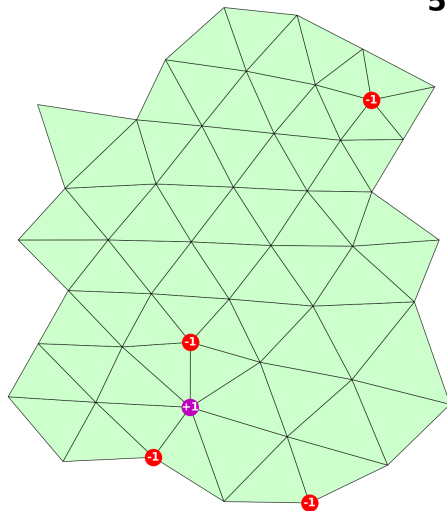
Results: Triangular Meshing

Triangular meshing
Example 1
Step 22 (out of 27)



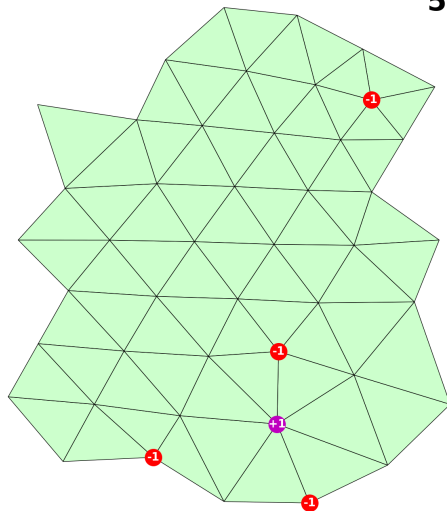
Results: Triangular Meshing

Triangular meshing
Example 1
Step 23 (out of 27)



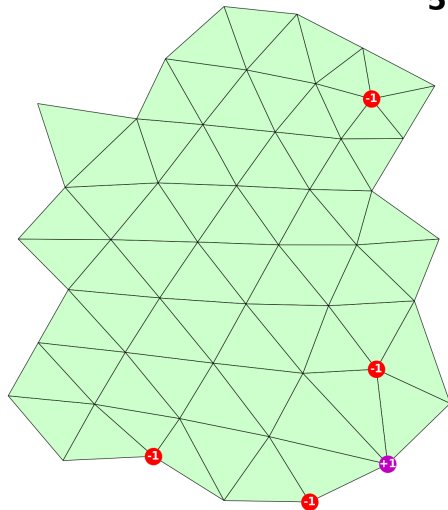
Results: Triangular Meshing

Triangular meshing
Example 1
Step 24 (out of 27)



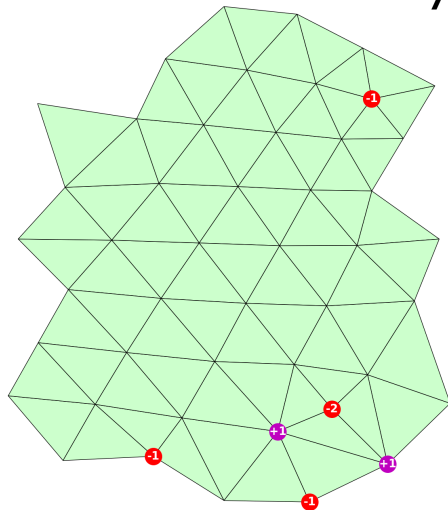
Results: Triangular Meshing

Triangular meshing
Example 1
Step 25 (out of 27)



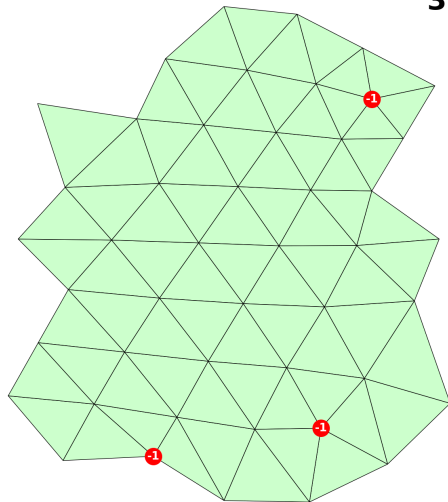
Results: Triangular Meshing

Triangular meshing
Example 1
Step 26 (out of 27)



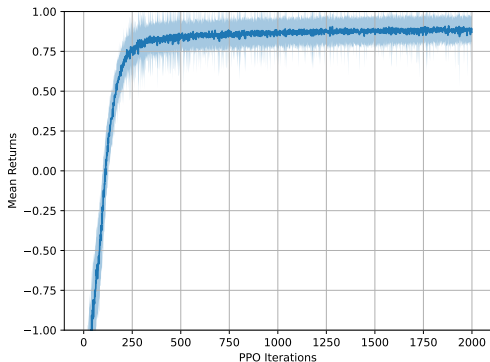
Results: Triangular Meshing

Triangular meshing
Example 1
Step 27 (out of 27)

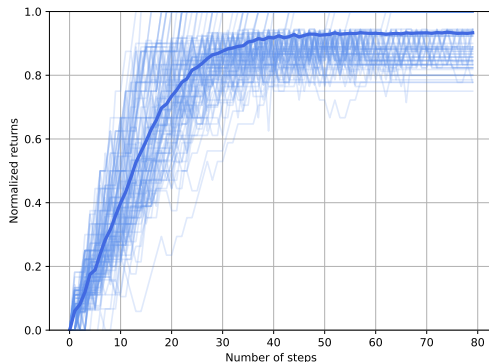


3 / 3

Results: Quadrilateral Meshes



Average performance over training history

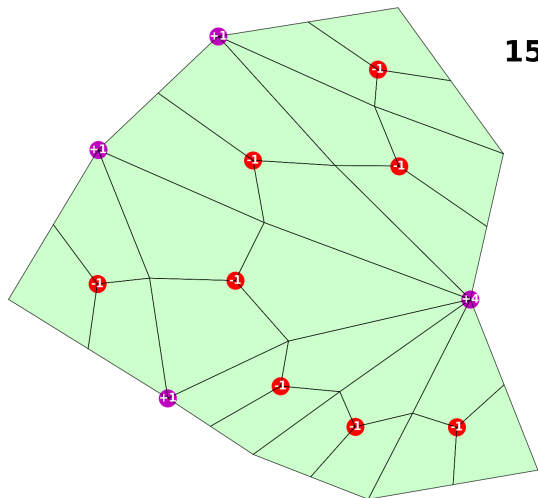


Evaluating the trained agent on multiple rollouts

Performance of the quadrilateral mesh agent over the training history.

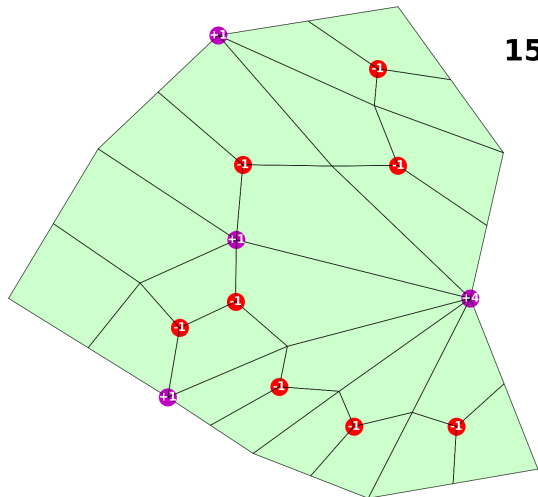
Results: Quadrilateral block meshing

Block mesh decomposition
Example 1
Step 0 (out of 19)



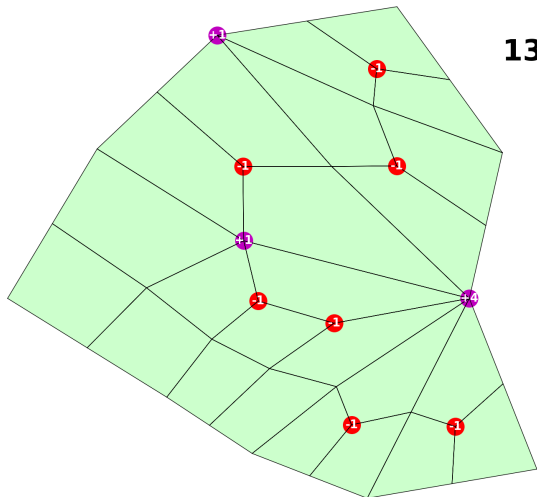
Results: Quadrilateral block meshing

Block mesh decomposition
Example 1
Step 1 (out of 19)



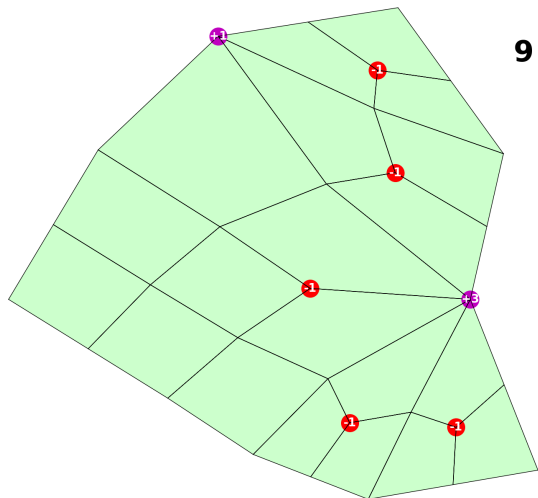
Results: Quadrilateral block meshing

Block mesh decomposition
Example 1
Step 2 (out of 19)



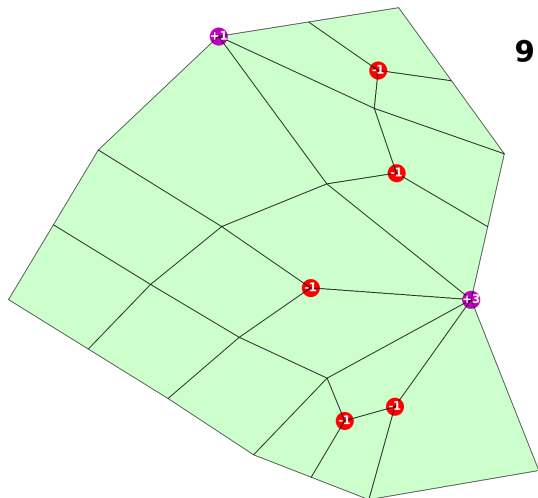
Results: Quadrilateral block meshing

Block mesh decomposition
Example 1
Step 3 (out of 19)



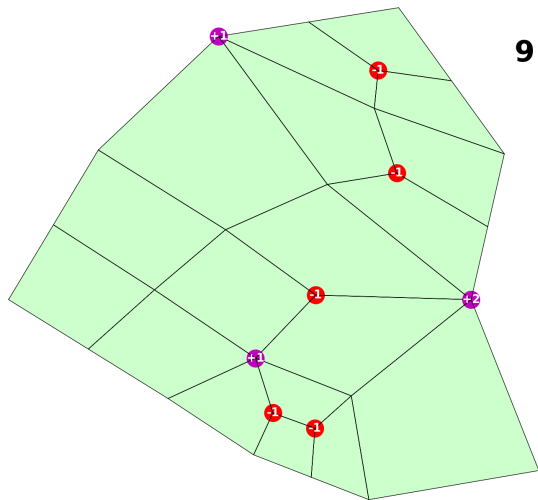
Results: Quadrilateral block meshing

Block mesh decomposition
Example 1
Step 4 (out of 19)



Results: Quadrilateral block meshing

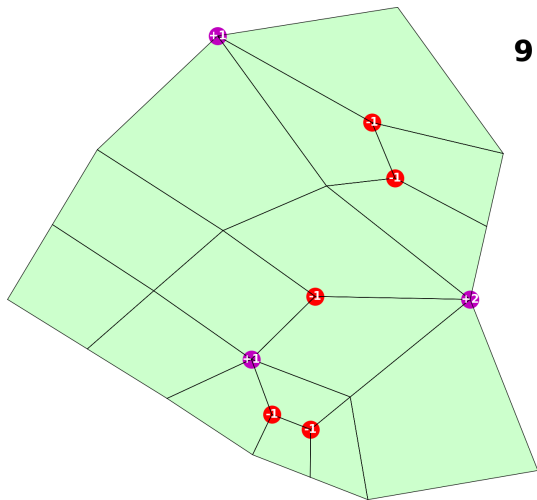
Block mesh decomposition
Example 1
Step 5 (out of 19)



9 / 1

Results: Quadrilateral block meshing

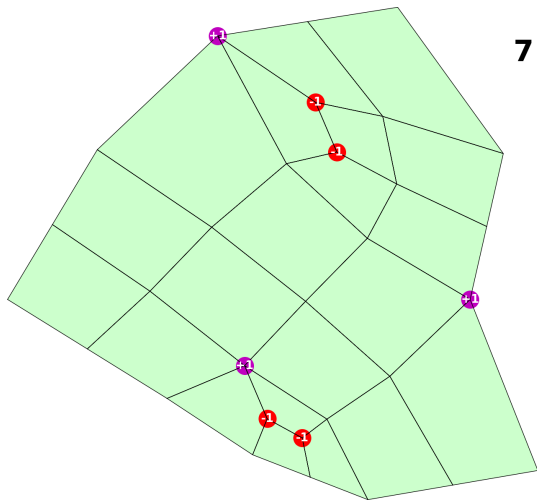
Block mesh decomposition
Example 1
Step 6 (out of 19)



9 / 1

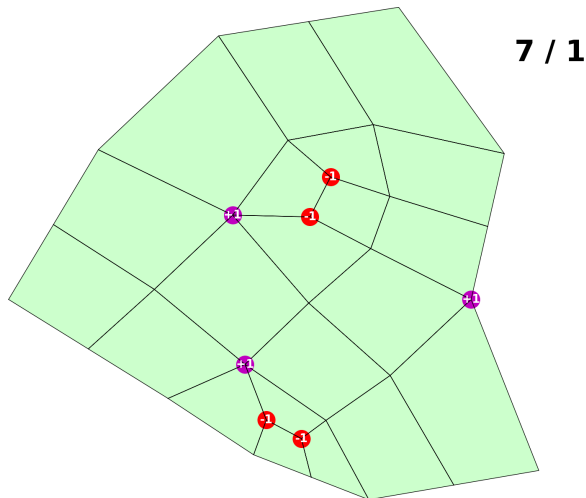
Results: Quadrilateral block meshing

Block mesh decomposition
Example 1
Step 7 (out of 19)



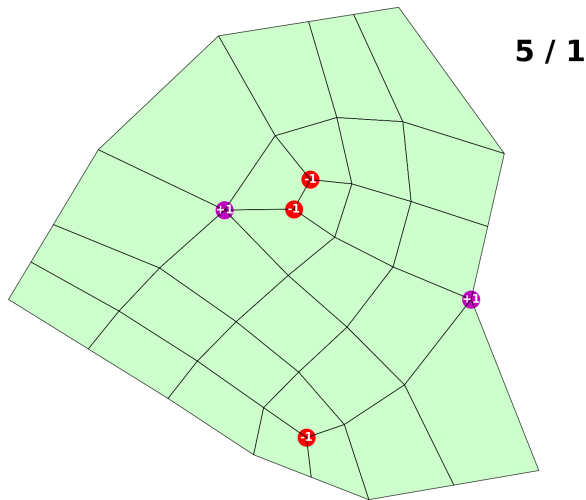
Results: Quadrilateral block meshing

Block mesh decomposition
Example 1
Step 8 (out of 19)



Results: Quadrilateral block meshing

Block mesh decomposition
Example 1
Step 9 (out of 19)

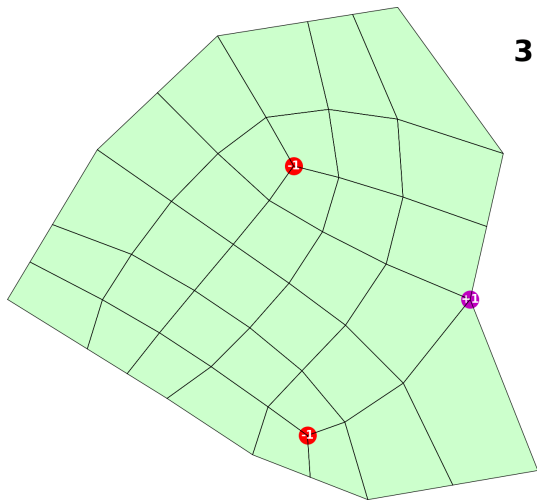


Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 10 (out of 19)



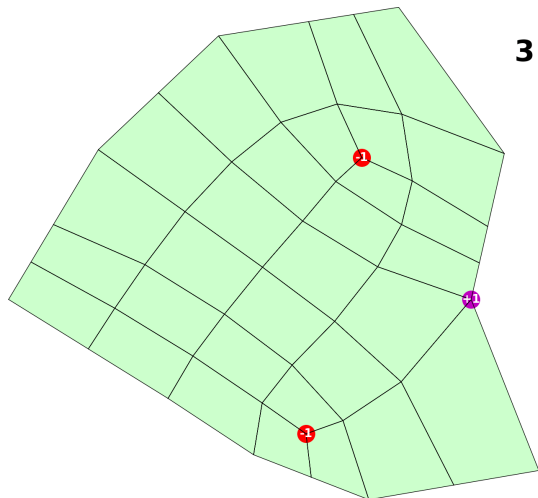
3 / 1

Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 11 (out of 19)



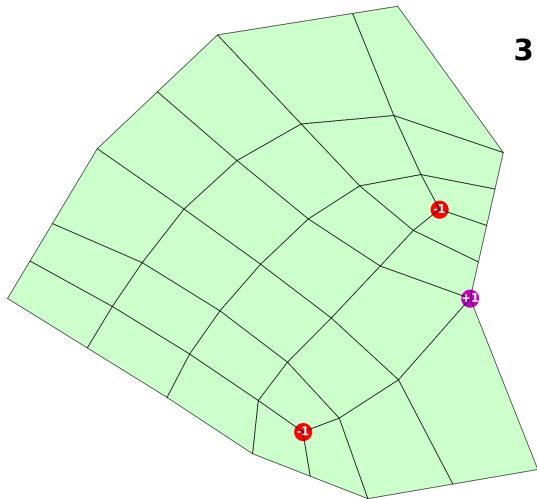
3 / 1

Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 12 (out of 19)

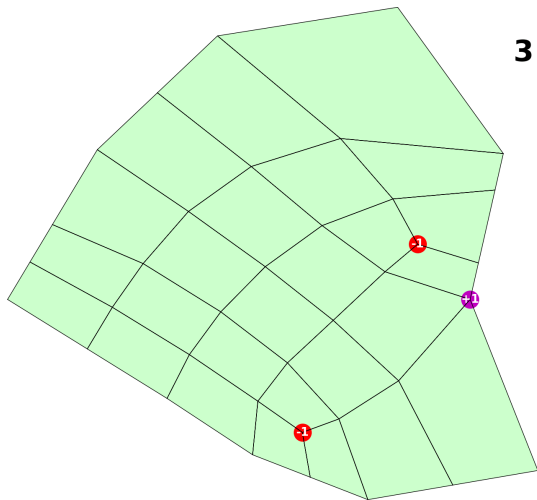


Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 13 (out of 19)



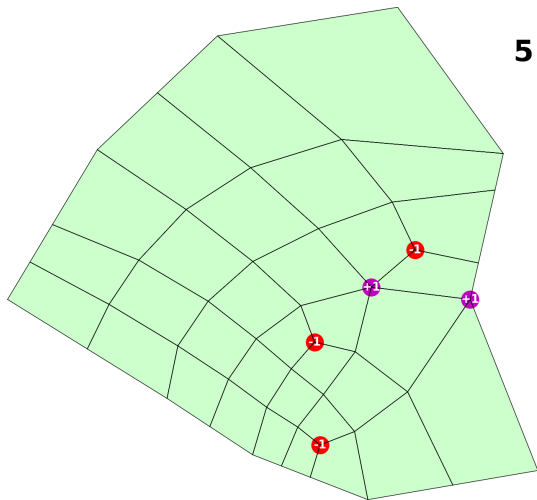
3 / 1

Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 14 (out of 19)

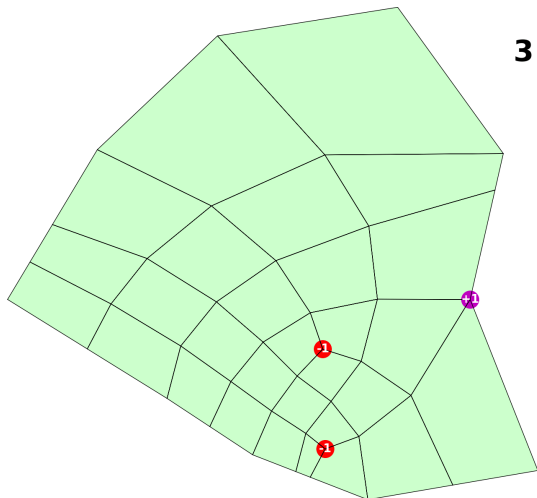


Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 15 (out of 19)

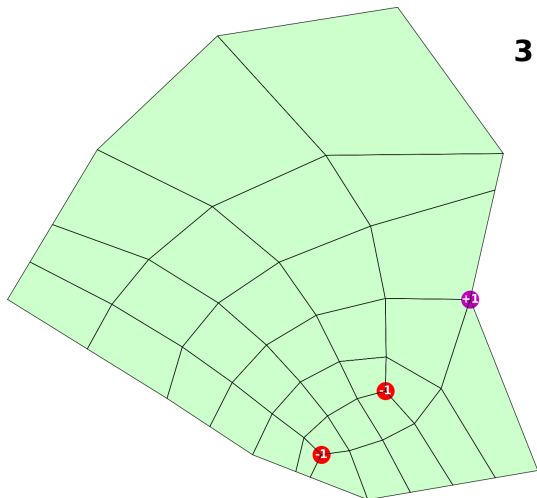


Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 16 (out of 19)



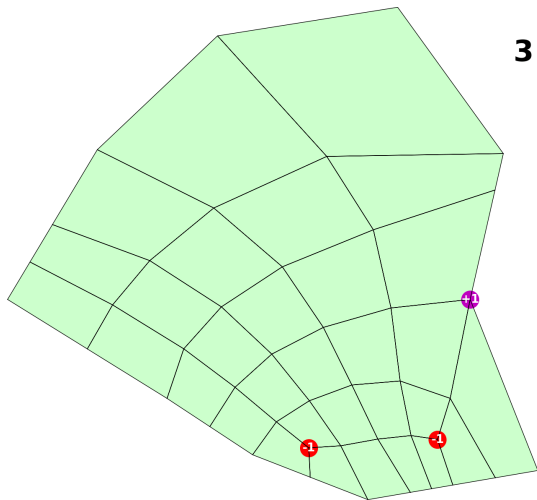
3 / 1

Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 17 (out of 19)

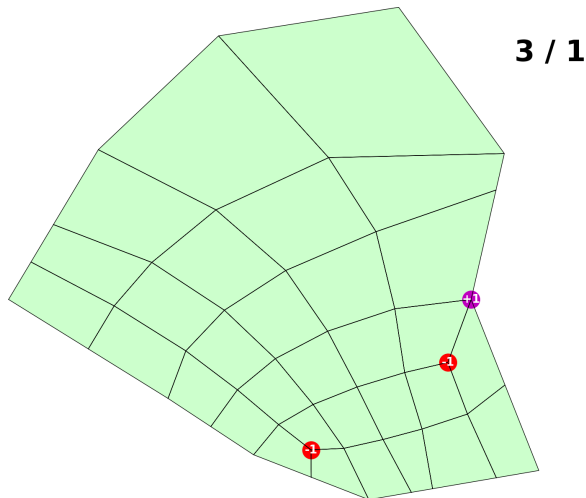


Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 18 (out of 19)

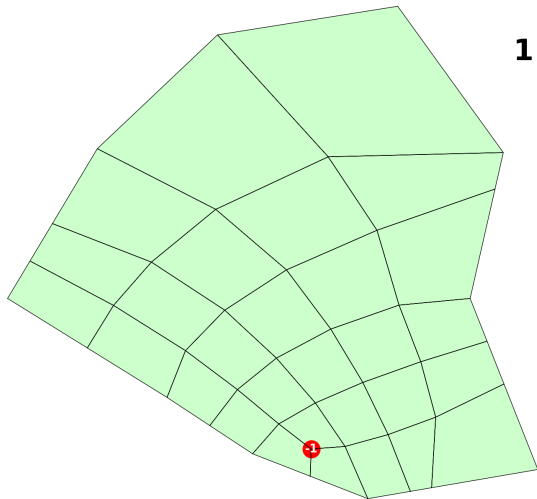


Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 19 (out of 19)

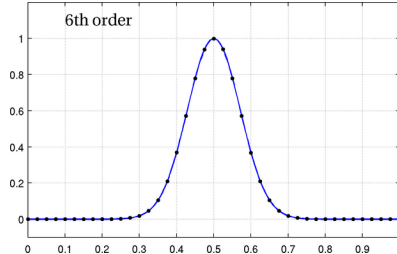
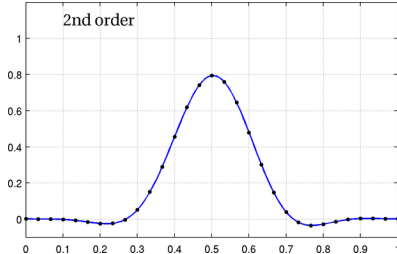
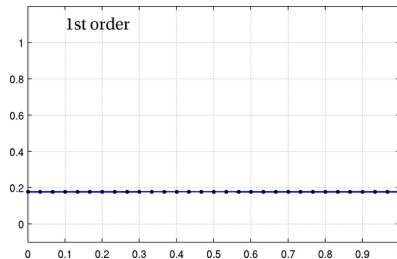
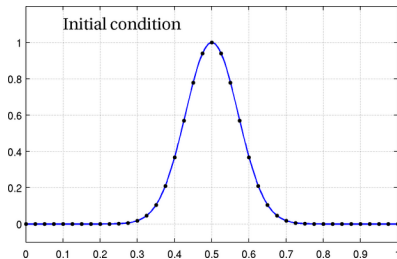


Outline

- 1 Introduction and Motivation
- 2 Unstructured Mesh Generation
 - The DistMesh Mesh Generator
 - Deep Reinforcement learning for Mesh Improvement
- 3 High-Order Methods for solving PDEs**
 - The Discontinuous Galerkin Method
 - Sparse Discretizations and Efficient Solvers
- 4 Methods for Deforming Domains
 - Multiphysics and Fluid-Structure Interaction
 - Unstructured Mesh Space-Time Methods
- 5 PDE-Constrained Optimization
 - Optimization-Based High-Order Shock Tracking

Why high-order accurate methods?

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



Outline

- 1 Introduction and Motivation
- 2 Unstructured Mesh Generation
 - The DistMesh Mesh Generator
 - Deep Reinforcement learning for Mesh Improvement
- 3 High-Order Methods for solving PDEs**
 - The Discontinuous Galerkin Method**
 - Sparse Discretizations and Efficient Solvers
- 4 Methods for Deforming Domains
 - Multiphysics and Fluid-Structure Interaction
 - Unstructured Mesh Space-Time Methods
- 5 PDE-Constrained Optimization
 - Optimization-Based High-Order Shock Tracking

The Finite Difference Method (FDM)

- Consider linear convection: $u_t + u_x = 0$ for $x \in [0, 1]$, $u(0) = u(1)$
- Approximate u_x point-wise using difference formulas:

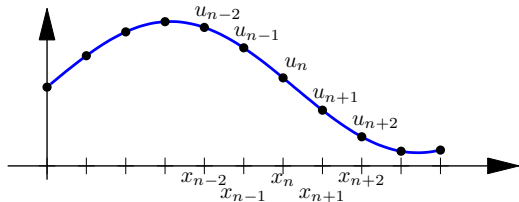
$$\frac{d}{dx}u(x_n) \approx \frac{u_{n+1} - u_{n-1}}{2\Delta x}$$

or high-order:

$$\frac{d}{dx}u(x_n) \approx \frac{u_{n+2} - 8u_{n+1} + 8u_{n-1} - u_{n-2}}{12\Delta x}$$

or one-sided (e.g. for stability, “upwinding”):

$$\frac{d}{dx}u(x_n) \approx \frac{3u_n - 16u_{n-1} + 36u_{n-2} - 48u_{n-3} + 25u_{n-4}}{25\Delta x}$$

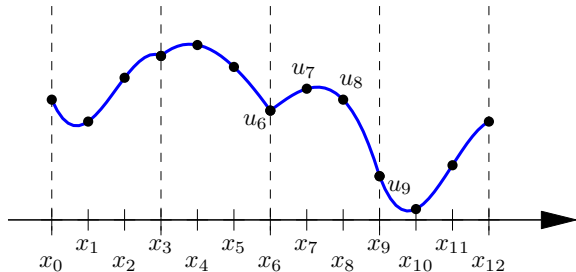


- Simple, efficient, flexible
- Needs *structured neighborhood* of nodes – hard to generalize to unstructured grids in 2-D and 3-D

The Finite Element Method (FEM)

- Discretize domain into *elements* (intervals)
- Seek approximate solution in space of piecewise polynomials $\hat{X}$
- Impose equation weakly: Seek $\hat{u} \in \hat{X}$ such that for all $v \in \hat{X}$:

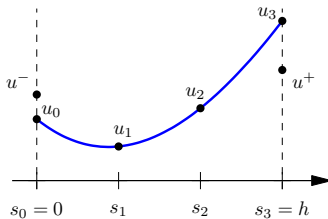
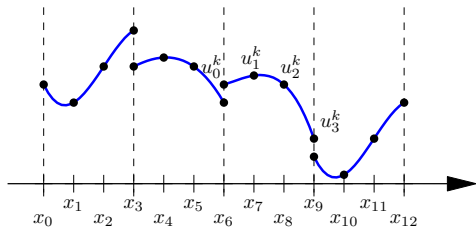
$$\begin{aligned} & \int_0^1 (\hat{u}_t + \hat{u}_x) v \, dx \\ &= \int_0^1 \hat{u}_t v \, dx + \int_0^1 \hat{u}_x v \, dx \\ &= \int_0^1 \hat{u}_t v \, dx - \int_0^1 \hat{u} v_x \, dx = 0 \end{aligned}$$



- Leads to semi-discrete system $M\mathbf{u}_t + K\mathbf{u} = 0$, with element-wise local M, K matrices
- Unclear how to stabilize by upwinding (but some techniques exist, e.g. SUPG)

The Discontinuous Galerkin Method

- Do not enforce continuity – allow “jumps” between elements



- Galerkin formulation for single element $\kappa = [0, h]$: For all $v \in P^p(\kappa)$,

$$\begin{aligned} \int_0^h (\hat{u}_t + \hat{u}_x) v \, dx &= \int_0^h \hat{u}_t v \, dx + \int_0^h \hat{u}_x v \, dx \\ &= \int_0^h \hat{u}_t v \, dx - \int_0^h \hat{u} v_x \, dx + \mathcal{U}(u^+, u_p) v(h) - \mathcal{U}(u_0, u^-) v(0) \end{aligned}$$

- Numerical flux function $\mathcal{U}(u_R, u_L)$ allows for stabilization by high-order upwinding, e.g.

$$\mathcal{U}(u_R, u_L) = u_L$$

The Discontinuous Galerkin Method

- Generalization to any system of conservation laws $\mathbf{u}_t + \nabla \cdot \mathcal{F}_i(\mathbf{u}) = 0$, in any dimension and any element shape:

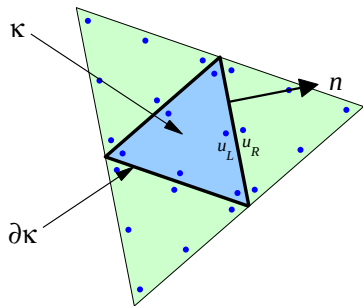
$$\int_{\kappa} [(\mathbf{u}_h)_t] \mathbf{v}_h d\mathbf{x} - \int_{\kappa} \mathcal{F}_i(\mathbf{u}_h) \nabla \mathbf{v}_h d\mathbf{x} + \int_{\partial\kappa} \hat{\mathcal{F}}_i(\mathbf{u}_h^+, \mathbf{u}_h^-, \hat{\mathbf{n}}) \mathbf{v}_h^+ ds = 0$$

with numerical flux function $\hat{\mathcal{F}}_i(\mathbf{u}_L, \mathbf{u}_R, \hat{\mathbf{n}})$.

- Seek approximate solution in space of element-wise polynomials:

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

- Global problem: Find $\mathbf{u}_h \in \mathcal{V}_h^p$ such that this weighted residual is zero for all $\mathbf{v}_h \in \mathcal{V}_h^p$
- Error = $\mathcal{O}(h^{p+1})$ for smooth solutions

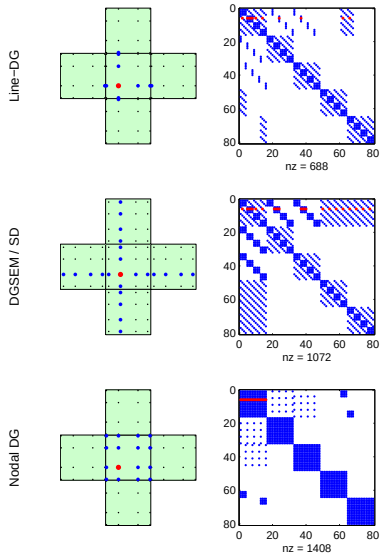


Outline

- 1 Introduction and Motivation
- 2 Unstructured Mesh Generation
 - The DistMesh Mesh Generator
 - Deep Reinforcement learning for Mesh Improvement
- 3 High-Order Methods for solving PDEs**
 - The Discontinuous Galerkin Method
 - Sparse Discretizations and Efficient Solvers**
- 4 Methods for Deforming Domains
 - Multiphysics and Fluid-Structure Interaction
 - Unstructured Mesh Space-Time Methods
- 5 PDE-Constrained Optimization
 - Optimization-Based High-Order Shock Tracking

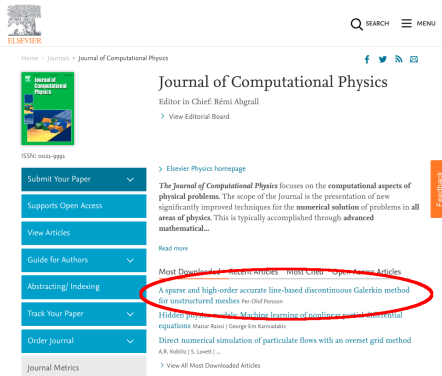
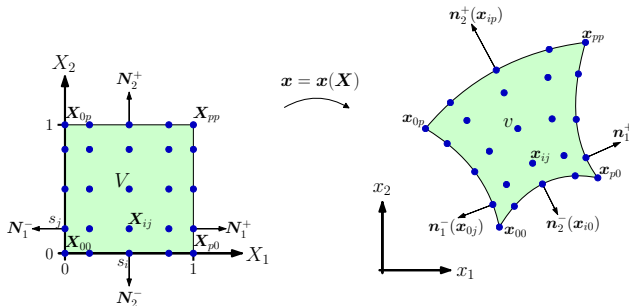
Sparse high-order discretizations, the Line-DG method

- High-order methods with drastically improved sparsity can be designed
- For example, the **Line-DG method** (Persson 2012) achieves an optimal sparsity pattern without under-integration
- This directly speeds up explicit schemes and matrix-vector computations
- **The main issue is how to precondition** – standard preconditioners such as element-based Jacobi / ILU destroy sparsity, but they are required for performance
- ***Goal: Develop efficient sparsity preserving approximate block preconditioners***



The Line-DG method

- The Line-DG method – map from reference element, but use 1D DG along each coordinate direction
- Optimal sparsity, correctly integrated along lines



One of the most downloaded
JCP papers (August 2018)

A Face Upwinded Spectral Element Method (FUSE) for Conservation Laws

Scientific Achievement

A new stabilization scheme for high-order continuous Spectral Element Methods which is provable convergent up to any order.

Significance and Impact

The work has the potential to drastically improve the performance of high-order methods, which are widely believed to be required for accurate predictions of turbulent flows and problems with waves and non-linear interactions.

Technical Approach

- Most stabilized schemes for fluids and other conservation laws are based on discontinuous formulations (e.g., the discontinuous Galerkin method)
- A remarkably simple way to stabilize continuous methods: Inspired by finite difference methods, choose the full upwind stencil only for face nodes
- Provably high-order convergent for a non-standard node distribution
- In addition, a line-based sparsity patterns bring the Jacobian cost from $\mathcal{O}(p^D)$ to $\mathcal{O}(pD)$, for polynomial degree p in D dimensions

PI(s)/Facility Lead(s): Per-Olof Persson, LBNL Math Group

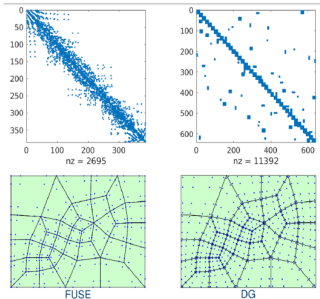
ASCR Program: Base Math

ASCR PM: Steven Lee

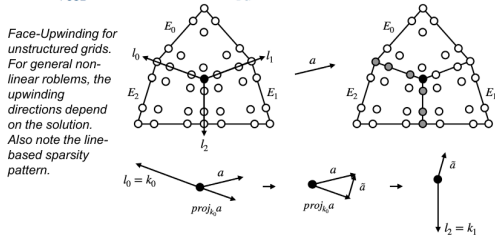
Publication(s) for this work:

Y. Pan, P.-O. Persson, "A Face-Upwinded Spectral Element Method on Unstructured Quadrilateral Meshes," *Journal of Computational Physics* (in review)

Y. Pan, P.-O. Persson, "A Stabilized Face-Upwinded High-Order Method for Incompressible Flows," *Proc. of 2023 AIAA AVIATION*, June 2023.

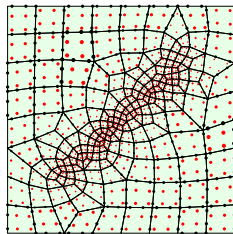
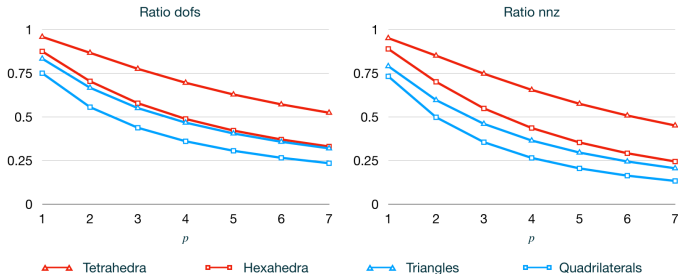


Sparsity pattern for FUSE vs DG, at polynomial degree 3. Due to continuous fields and line-based sparsity patterns, the Jacobian matrices are more than 4 times cheaper. This effect increases in 3D and for higher degrees.

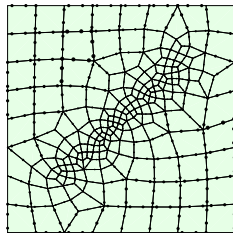


Half-Closed Discontinuous Galerkin Methods

- Nodes placed only on subset of boundaries
- Same operator sparsity as with standard closed nodes
- Gauss-Radau nodes gives extra degree of quadrature accuracy, and often an extra degree of super-convergence
- Static condensation for DG, same sparsity as CG
- 4x-5x smaller Jacobians + 2x faster multigrid convergence



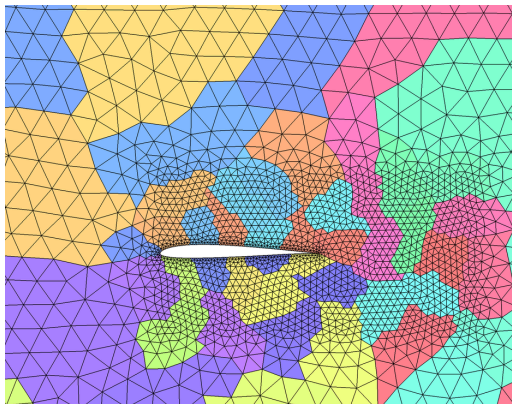
Half-closed DG nodes



After static condensation

Efficient parallel Newton-Krylov Solvers

- Parallel iterative implicit solvers required for real-world CFD
- *hp*-multigrid and Block-ILU(0) preconditioners with Minimum Discarded Fill (MDF) element ordering [Persson/Peraire 2008]
- Distributed parallel solvers developed in [Persson '09]
- IMEX schemes for geometrically induced stiffness [Persson 2011]
- Approximate tensor-product preconditioners for very high order DG methods [Pazner, Persson 2017,2018]
- Agglomeration-Based Multigrid Solvers [Pan, Persson 2021]

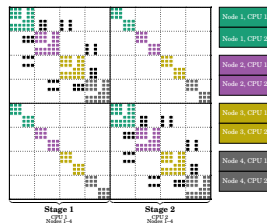
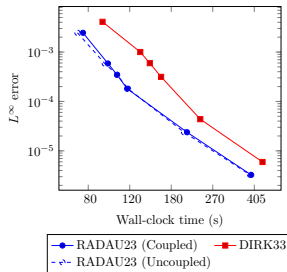


Stage-Parallel Implicit Runge-Kutta Methods

- s -stage Implicit Runge-Kutta (IRK) Method for $M \frac{\partial u}{\partial t} = f(u)$:

$$Mk_i = f \left(t_0 + \Delta t c_i, u_0 + \Delta t \sum_{j=1}^s a_{ij} k_j \right), \quad u_1 = u_0 + \Delta t \sum_{i=1}^s b_i k_i$$

- Change variable and solve for stage solutions $W = (A \otimes I_n)K$ for increased sparsity
- Precondition by stage-uncoupled shifted block ILU(0)
- Perfect stage parallelization by allocating high communication to shared memory



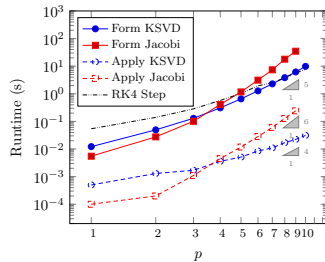
Approximate tensor-product preconditioners

- Develop *implicit* high-order methods with *linear complexity* in the polynomial degree p per DOF, that is, $\mathcal{O}(p^{d+1})$, similar to the Jacobian matrices in the Line-DG scheme
- For element-wise inverses, find best approximation with Kronecker-SVDs:

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

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

- Use Schur factorization for efficient inversion (similar to matrix diagonalization)
- Asymptotically superior, lower run-time already at $p > 3$



W. Pazner and P.-O. Persson, *Approximate tensor-product preconditioners for very high order discontinuous Galerkin methods*, J. Comput. Phys., Vol. 354, pp. 344-369, February 2018.

W. Pazner and P.-O. Persson, *Interior penalty tensor-product preconditioners for high-order discontinuous Galerkin discretizations*. Proc. of the 2018 AIAA Aerospace Sciences Meeting, January 2018. AIAA-2018-1093.

W. Pazner and P.-O. Persson, *High-Order DNS and LES Simulations Using an Implicit Tensor-Product Discontinuous Galerkin Method*. Proc. of the 23rd AIAA Computational Fluid Dynamics Conference, June 2017. AIAA-2017-3948. **(Winner, AIAA CFD 2017 Student Paper Competition)**

Outline

- 1 Introduction and Motivation
- 2 Unstructured Mesh Generation
 - The DistMesh Mesh Generator
 - Deep Reinforcement learning for Mesh Improvement
- 3 High-Order Methods for solving PDEs
 - The Discontinuous Galerkin Method
 - Sparse Discretizations and Efficient Solvers
- 4 Methods for Deforming Domains**
 - Multiphysics and Fluid-Structure Interaction
 - Unstructured Mesh Space-Time Methods
- 5 PDE-Constrained Optimization
 - Optimization-Based High-Order Shock Tracking

ALE Formulation for Deforming Domains

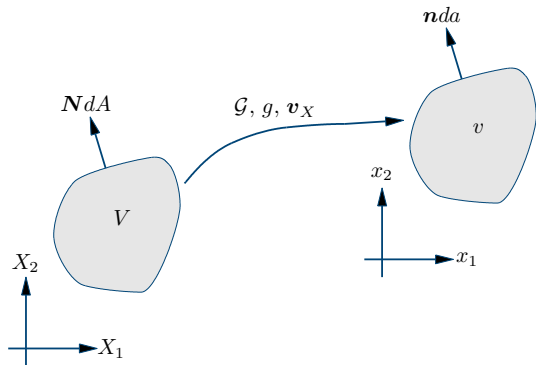
- Use mapping-based ALE formulation for moving domains [Visbal/Gaitonde '02], [Persson/Bonet/Peraire '09]
- Map from reference domain V to physical deformable domain $v(t)$
- Introduce the *mapping deformation gradient* $\mathbf{G}$ and the *mapping velocity* $\mathbf{v}_X$ as

$$\mathbf{G} = \nabla_X \mathcal{G}$$

$$\mathbf{v}_X = \left. \frac{\partial \mathcal{G}}{\partial t} \right|_X$$

and set $g = \det(\mathbf{G})$

- Transform equations to account for the motion



Transformed Equations

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

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

can be written in the reference configuration V as

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

where

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

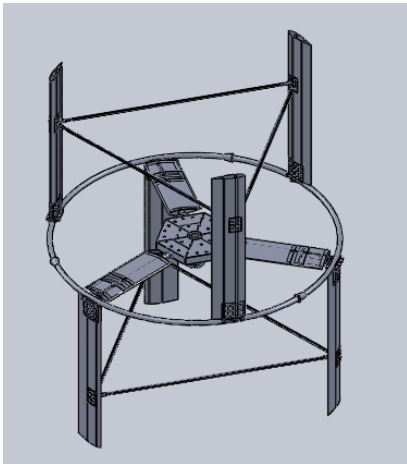
and

$$\nabla_x \mathbf{U}_x = \nabla_X (g^{-1} \mathbf{U}_X) \mathbf{G}^{-T} = (g^{-1} \nabla_X \mathbf{U}_X - \mathbf{U}_X \nabla_X (g^{-1})) \mathbf{G}^{-T}$$

- Details in [Persson/Bonet/Peraire '09], including how to satisfy the so-called Geometric Conservation Law (GCL)

Vertical Axis Wind Turbines

- Experimental design by G. Dahlbacka (LBNL) and collaborators
- 3kW unit, CAD design (left) and assembled unit (right)

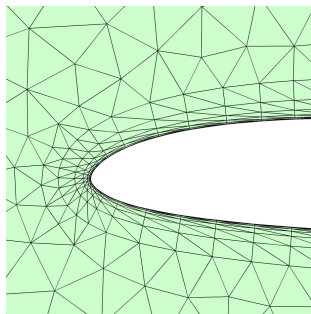
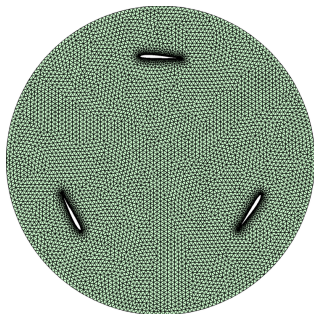


VAWT – Mathematical Model and Discretization

- Preliminary 2D simulation, using vertical symmetry
- Solve the Navier-Stokes equations in a rotating frame:

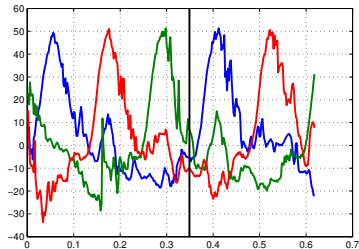
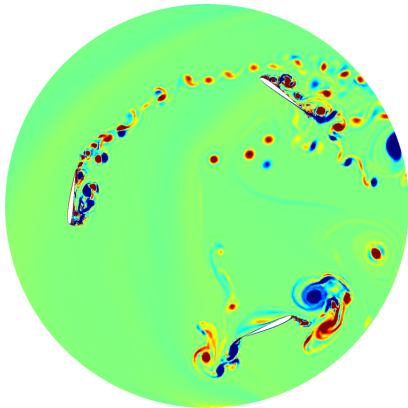
$$\mathcal{G}(X, Y, t) = \begin{bmatrix} \cos \omega t & -\sin \omega t \\ \sin \omega t & \cos \omega t \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

- Hybrid boundary layer/unstructured mesh, element degree $p = 3$



VAWT – Numerical Results

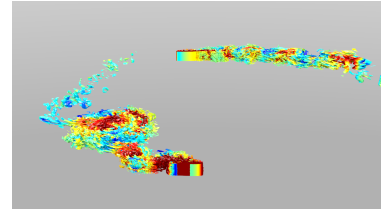
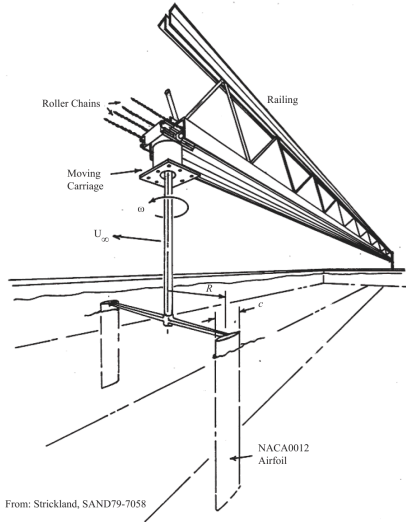
- Simulation with freestream wind speed 12 m/s (horizontally, from the left) and wing tip speed ratio 1.5
- Visualization by z -vorticity in rotating frame



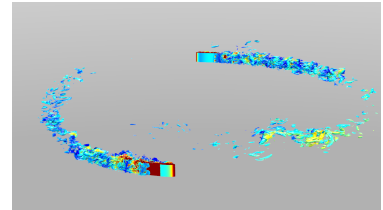
Moment on each blade vs. time

Sandia National Laboratories' Low Re VAWT

Sandia National Labs tow tank experiment from 1979



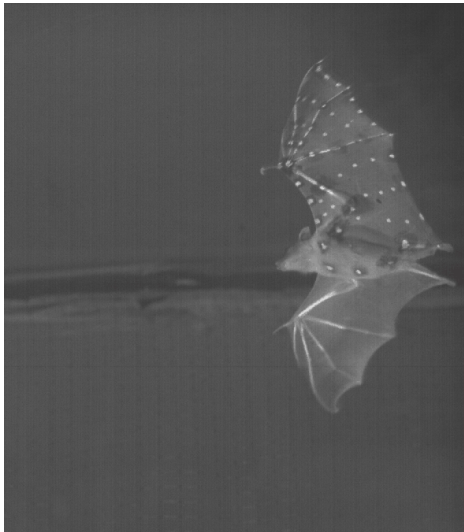
TSR = 2.5



TSR = 5.0

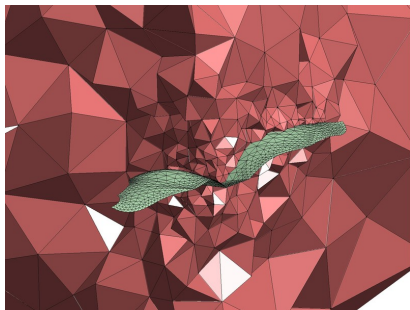
Bio-Inspiration for Flapping Wing MAVs

- Develop high-order accurate simulation capabilities that capture the complex physics in flapping flight
- Use the computational tools for increased understanding and to design optimized flapping kinematics

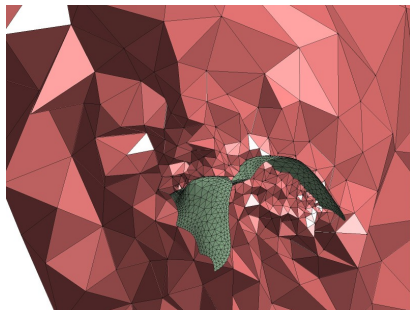


Domain Mapping

- Highly complex wing motion from measured data
- Construct mapping $\mathcal{G}(X, t)$ *numerically* by nonlinear solid mechanics approach [Persson '09]
- A reference mesh (left) is deformed elastically to smoothly align with the prescribed wing motion (right)
- Grid velocity $\mathbf{v}_X = \frac{\partial \mathcal{G}}{\partial t} \Big|_X$ defined consistently with DIRK scheme



$\mathcal{G}(X, t)$

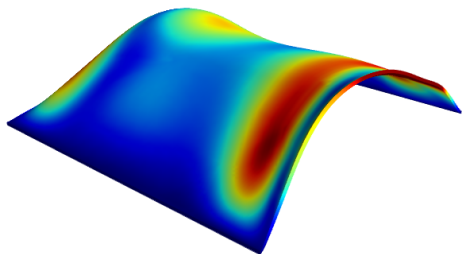
A curved arrow pointing from the reference mesh on the left to the deformed mesh on the right, indicating the mapping function $\mathcal{G}(X, t)$.

Outline

- 1 Introduction and Motivation
- 2 Unstructured Mesh Generation
 - The DistMesh Mesh Generator
 - Deep Reinforcement learning for Mesh Improvement
- 3 High-Order Methods for solving PDEs
 - The Discontinuous Galerkin Method
 - Sparse Discretizations and Efficient Solvers
- 4 Methods for Deforming Domains**
 - Multiphysics and Fluid-Structure Interaction**
 - Unstructured Mesh Space-Time Methods
- 5 PDE-Constrained Optimization
 - Optimization-Based High-Order Shock Tracking

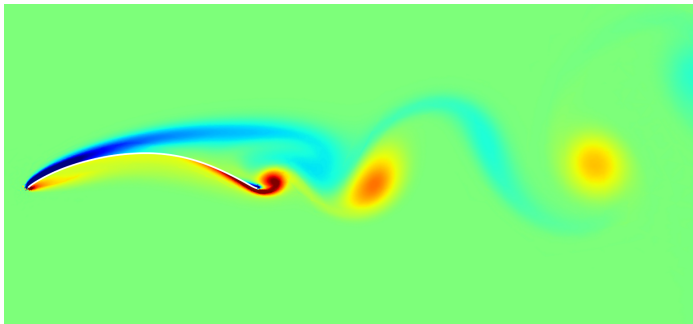
Fluid-Structure Interaction with IMEX Schemes

- Many important problems require predictions of fluid-structure interaction (FSI):
 - Oscillatory interactions in engineering systems (e.g. aircraft, turbines, and bridges) can lead to failure
 - The blood flow in arteries and artificial heart valves is highly dependent on structural interactions
- Requirements on numerical solvers:
 - High-order accuracy, to capture non-linear interactions and multiscale phenomena
 - Unstructured meshes, for complex geometries and adaptivity



Full vs. weak coupling

- Two main numerical approaches for the coupling:
 - Fully coupled (monolithic): Solve the fluid/structure equations simultaneously. Accurate, but requires specialized codes and solvers are often slow.
 - Weakly coupled (partitioned): Use standard solvers for fluid/structure and apply a separate coupling scheme, often together with subiterations. Efficient and simple, but issues with accuracy and stability.



Lagrangian Solid Dynamics FEM Discretization

- Map from reference domain V to physical domain $v(t)$

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} , \quad J = \det \mathbf{F} , \quad \mathbf{v}(\mathbf{X}, t) = \frac{\partial \mathbf{x}}{\partial t} , \quad \mathbf{p} = \rho_0 \mathbf{v}$$

- Volumetric non-linear elasticity:

$$\frac{\partial \mathbf{p}}{\partial t} - \nabla \cdot \mathbf{P} = \mathbf{b} \quad \text{in } \Omega$$

$$\mathbf{P}(\mathbf{F}) \cdot \mathbf{N} = \mathbf{t} \quad \text{on } \Gamma_N$$

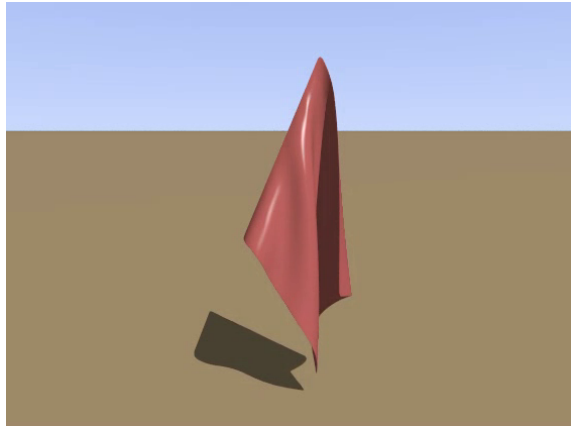
$$\mathbf{x} = \mathbf{x}_D \quad \text{on } \Gamma_D$$

with *first Piola-Kirchhoff stress tensor* $\mathbf{P}(\mathbf{F})$ given by hyperelastic neo-Hookean constitutive model

- Standard high-order FEM discretization; write as first order system.
- Volumetric modeling of thin structures $\implies$ stiff nonlinear systems
- However, direct solvers scale well due to 2-D nature of the mesh

Large Deformation Solid Dynamics

- The dynamic deformation of a rubber-like material



Partitioned FSI using IMEX schemes

- [Froehle & Persson 2013, 2014], [Huang, Persson, Zahr 2019]
- IMEX schemes can be used to derive accurate partitioning methods for fully coupled FSI problems
- Treat a *predicted* traction $\tilde{\mathbf{t}}$ explicitly and everything else implicitly:

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

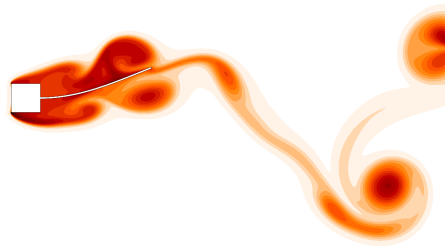
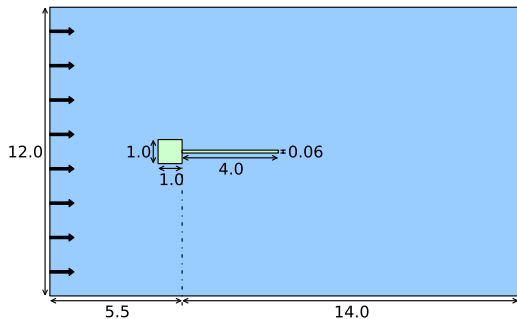
- Use Runge-Kutta based predictor [Van Zuijlen & Bijl 2005]

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

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

Verification: Cantilever System

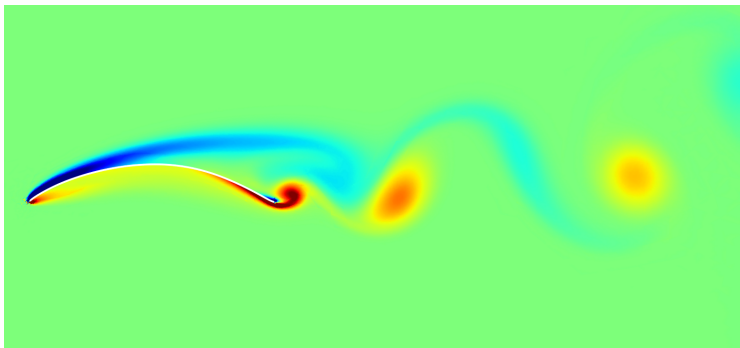
- Standard FSI benchmark problem. [Wall & Ramm 1998]
- Flow around elastic cantilever behind a square bluff body



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

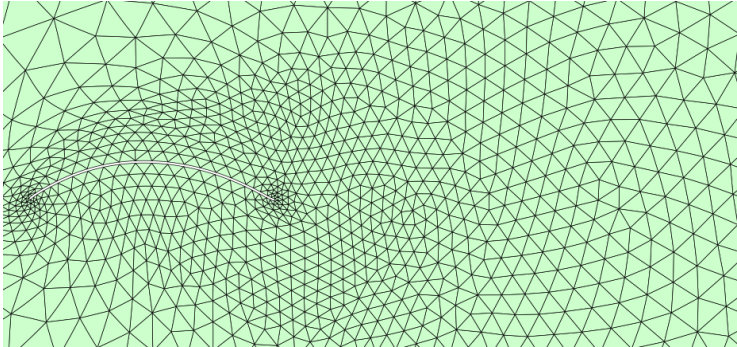
Flow around membrane, 2-D

- Volumetric modeling of Lagrangian Neo-Hookean membrane
- Membrane ends are held fixed but allowed to rotate
- Angle of attack 20° , Reynolds number 2,000
- Implicit schemes handle complex behavior with large time-steps
- Low membrane stiffness



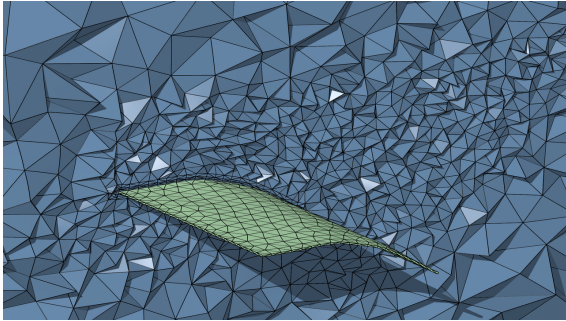
Flow around membrane, 2-D

- Mesh motion

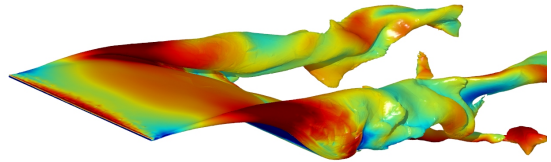


Flow around Membrane, 3-D

- 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



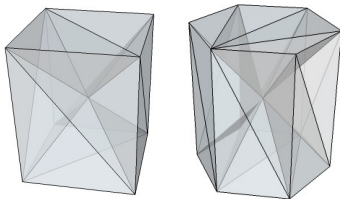
Flow field

Outline

- 1 Introduction and Motivation
- 2 Unstructured Mesh Generation
 - The DistMesh Mesh Generator
 - Deep Reinforcement learning for Mesh Improvement
- 3 High-Order Methods for solving PDEs
 - The Discontinuous Galerkin Method
 - Sparse Discretizations and Efficient Solvers
- 4 Methods for Deforming Domains**
 - Multiphysics and Fluid-Structure Interaction
 - Unstructured Mesh Space-Time Methods**
- 5 PDE-Constrained Optimization
 - Optimization-Based High-Order Shock Tracking

Unstructured mesh methods for large deformation problems

- Limitations with ALE for very large domain deformations
- Remeshing and solution transfer expensive and inaccurate
- ***Develop numerical schemes that are integrated with an iterative moving mesh generator***
- Various schemes:
 - Space-time DG formulations on fully unstructured 4D meshes
 - Combined ALE and local L_2 -projection DG method

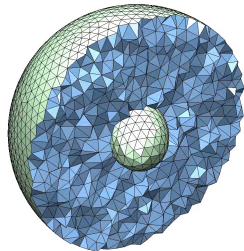
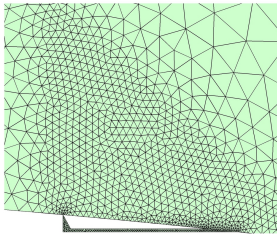
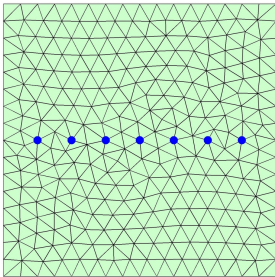


L. Wang and P.-O. Persson, *A high-order discontinuous Galerkin method with unstructured space-time meshes for two-dimensional compressible flows on domains with large deformations*. Comput. & Fluids, Vol. 118, pp. 53-68, September 2015.

L. Wang, *Discontinuous Galerkin Methods on Moving Domains with Large Deformations*. Ph.D. thesis, University of California, Berkeley. May 2015.

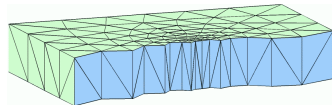
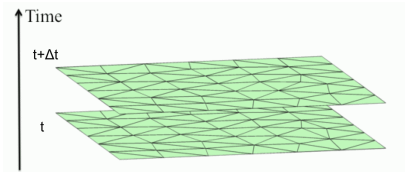
Moving Meshes

- Use the DistMesh mesh generation algorithm (Persson 2005) 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

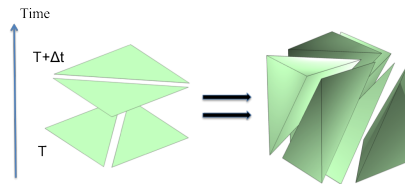
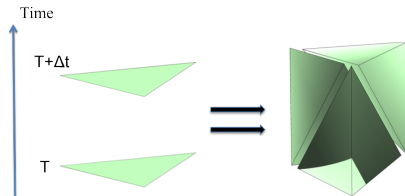


Space-time Mesh Generation

- Local mesh operations allows to space-time slab mesh generation



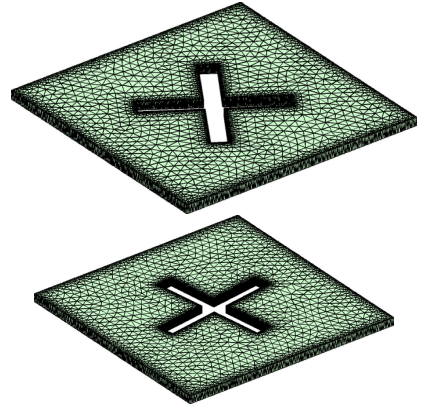
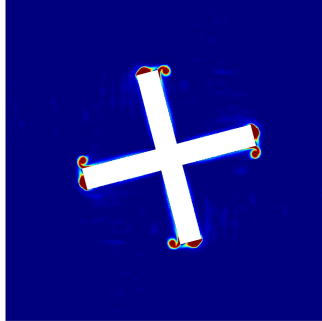
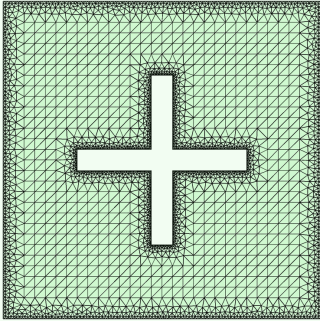
- Each local prism triangulation depends on the choice of the diagonals on lateral faces



- Can be extended to generation of 4D unstructured simplex meshes (3D + time)

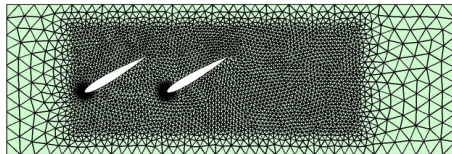
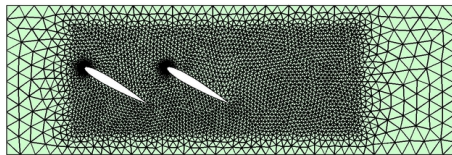
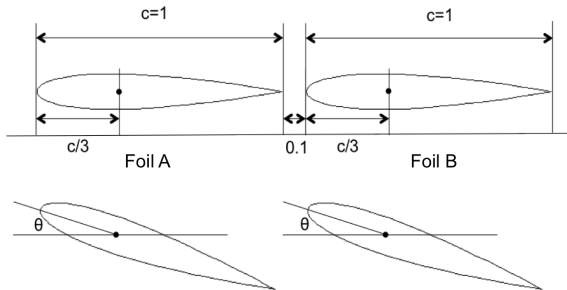
Example: Spinning Cross

- Flow around a spinning cross,
 $Re = 3000$, $Mach\ 0.2$, $p = 2$.

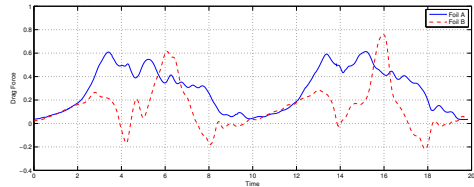
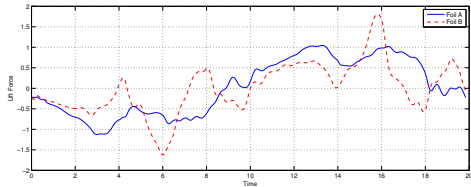
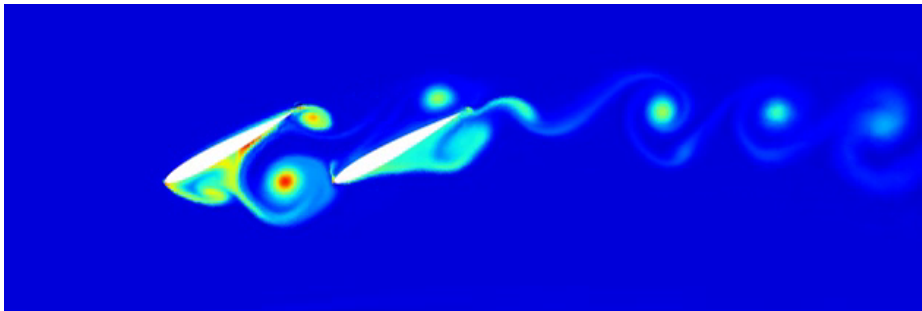


Example: Tandem Foils

- 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

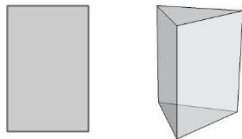


Fully unstructured 4D space-time mesh generation

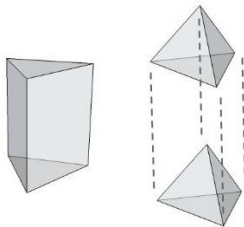
- 4D is difficult to visualize, and we extend concepts such as 'Prisms' and 'Lateral Faces' using combinatorial notions



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

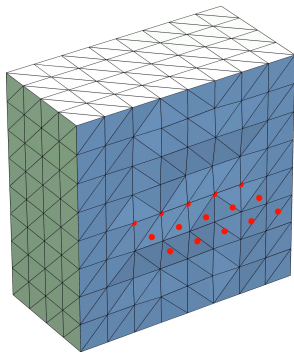
$$T_1 = \{p_{(1)}^{V,t}, p_{(1)}^{V,t+\Delta t}, p_{(2)}^{V,t+\Delta t}, p_{(3)}^{V,t+\Delta t}, p_{(4)}^{V,t+\Delta t}\},$$

$$T_2 = \{p_{(1)}^{V,t}, p_{(2)}^{V,t}, p_{(2)}^{V,t+\Delta t}, p_{(3)}^{V,t+\Delta t}, p_{(4)}^{V,t+\Delta t}\},$$

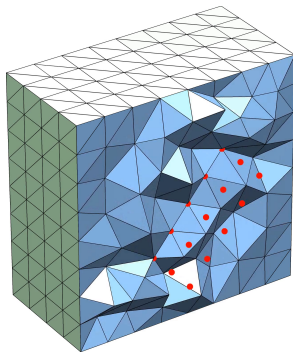
$$T_3 = \{p_{(1)}^{V,t}, p_{(2)}^{V,t}, p_{(3)}^{V,t}, p_{(3)}^{V,t+\Delta t}, p_{(4)}^{V,t+\Delta t}\},$$

$$T_4 = \{p_{(1)}^{V,t}, p_{(2)}^{V,t}, p_{(3)}^{V,t}, p_{(4)}^{V,t}, p_{(4)}^{V,t+\Delta t}\}$$

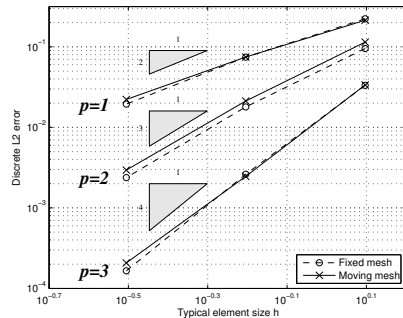
Extruded 3D Euler vortex convergence test



Initial mesh at $t = 0$



Moving mesh at $t = T$



Convergence Plot

Outline

- 1 Introduction and Motivation
- 2 Unstructured Mesh Generation
 - The DistMesh Mesh Generator
 - Deep Reinforcement learning for Mesh Improvement
- 3 High-Order Methods for solving PDEs
 - The Discontinuous Galerkin Method
 - Sparse Discretizations and Efficient Solvers
- 4 Methods for Deforming Domains
 - Multiphysics and Fluid-Structure Interaction
 - Unstructured Mesh Space-Time Methods
- 5 **PDE-Constrained Optimization**
 - Optimization-Based High-Order Shock Tracking

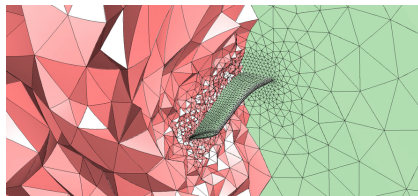
Discretization of PDE-Constrained Optimization

- Continuous PDE-constrained optimization problem

$$\underset{U, \mu}{\text{minimize}} \quad \mathcal{J}(U, \mu)$$

$$\text{subject to} \quad C(U, \mu) \leq 0$$

$$\frac{\partial U}{\partial t} + \nabla \cdot F(U, \nabla U) = 0 \quad \text{in } v(\mu, t)$$



- Solve using fully discrete adjoints and high-order spatial/temporal solvers

[Zahr, Persson 2017, 2018]

Energy = 1.45e-01, Thrust = -1.12e-01

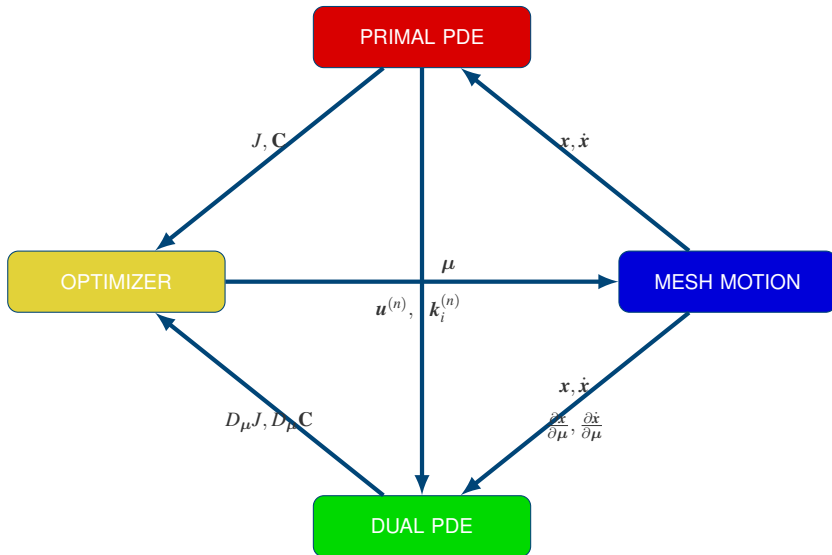


Energy = 3.14-01, Thrust = 0.0e+00



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 $F(\mathbf{u}^{(n)}, \mathbf{k}_i^{(n)}, \boldsymbol{\mu})$
 - Represents either the **objective** function or a **constraint**
- **Adjoint method**: indirect method for computing $D_{\boldsymbol{\mu}}F$ that require one linear evolution evaluation equation for each quantity of interest, F
 - Related to *backprop* in machine learning, or *reverse mode automatic differentiation* (*)
- **Linear** evolution equations solved **backward** in time
- **Primal** state $\mathbf{u}_i^{(n)}$ required at each stage of dual problem
- Gradient reconstruction via dual variables

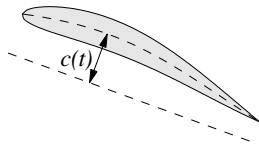
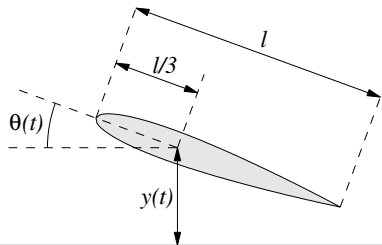
$$D_{\boldsymbol{\mu}}F = \frac{\partial F}{\partial \boldsymbol{\mu}} + \boldsymbol{\lambda}^{(0)T} \frac{\partial \mathbf{u}_0}{\partial \boldsymbol{\mu}} + \sum_{n=1}^{N_t} \Delta t_n \sum_{i=1}^s \boldsymbol{\kappa}_i^{(n)T} \frac{\partial \mathbf{r}}{\partial \boldsymbol{\mu}}(\mathbf{u}_i^{(n)},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

(*) F. Sapienza, J. Bolibar, F. Schäfer, B. Groenke, A. Pal, V. Boussange, P. Heimbach, G. Hooker, F. Pérez, P.-O. Persson, C. Rackauckas, *Differentiable Programming for Differential Equations: A Review*.
arXiv:2406.09699.

Energetically Optimal Flapping, Thrust Constraint

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

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

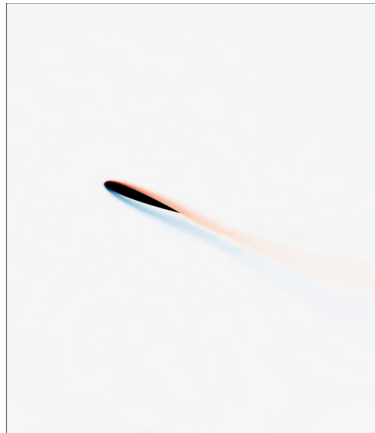


Airfoil schematic, kinematic description

Energetically optimal flapping vs. required thrust

Energy = 0.21935

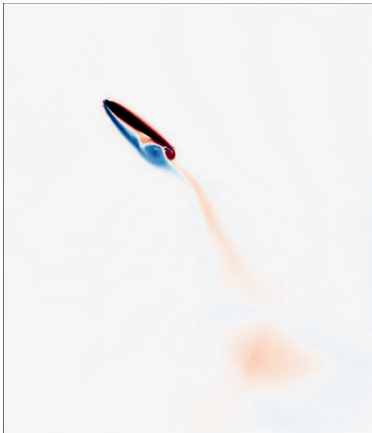
Thrust = 0.0000



Optimal $T_x = 0$

Energy = 3.00404

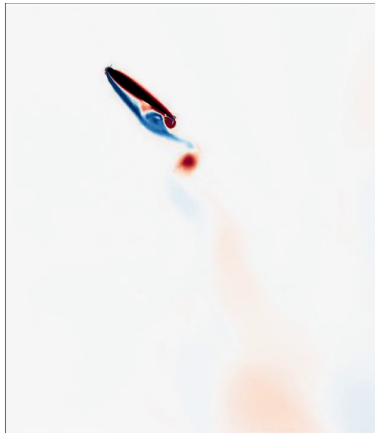
Thrust = 1.5000



Optimal $T_x = 1.5$

Energy = 6.2869

Thrust = 2.5000



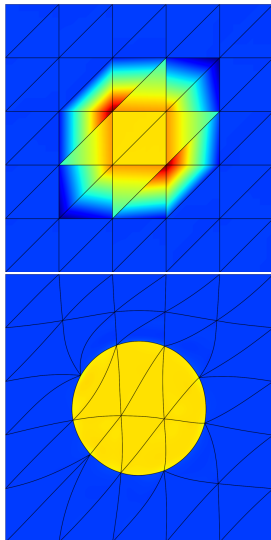
Optimal $T_x = 2.5$

Outline

- 1 Introduction and Motivation
- 2 Unstructured Mesh Generation
 - The DistMesh Mesh Generator
 - Deep Reinforcement learning for Mesh Improvement
- 3 High-Order Methods for solving PDEs
 - The Discontinuous Galerkin Method
 - Sparse Discretizations and Efficient Solvers
- 4 Methods for Deforming Domains
 - Multiphysics and Fluid-Structure Interaction
 - Unstructured Mesh Space-Time Methods
- 5 PDE-Constrained Optimization
 - Optimization-Based High-Order Shock Tracking

Optimization-Based High-Order Shock Tracking

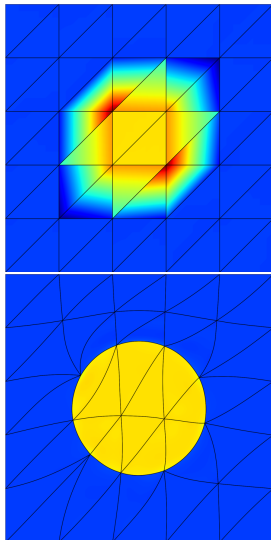
- Motivation: Use our localized moving mesh generators and DG solvers to align the discontinuities in the approximation space (element faces) with the discontinuities in the solution
- Difficult problem: Complex geometries of shock surfaces, cannot solve DG problem unless mesh is perfectly aligned
- Our solution: Use optimization framework, with ALE formulation and discrete adjoints for the sensitivities [Zahr,Persson JCP 2018], [Zahr et al, 2019, 2020, 2021, ...]
- Full space solver required for non-linear stability



High-order implicit shock tracking

Ingredients:

- Discontinuous Galerkin discretization: inter-element jumps, high-order
- Discontinuity-aligned mesh is the solution of an optimization problem constrained by the discrete PDE
 $\implies$ *implicit shock tracking*
- Full space solver that converges the solution and mesh simultaneously to ensure solution of PDE never required on non-aligned mesh



Discontinuous Galerkin discretization of conservation law

- Inviscid conservation law:

$$\nabla \cdot F(U) = 0 \quad \text{in } \Omega$$

- Element-wise weak form of conservation law, numerical flux function:

$$\int_{\partial K} \psi^+ \cdot \mathcal{H}(U^+, U^-, n) dS - \int_K F(U) : \nabla \psi dV = 0$$

- Introduce test space $\mathcal{V}_{h,p'}$ and trial space $\mathcal{V}_{h,p}$, where h is the element size and p/p' is the polynomial degree, to define the finite-dimensional residual

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

- Introduce basis for polynomial spaces to obtain discrete residuals

$$\mathbf{r}(\mathbf{u}, \mathbf{x}) \quad (p' = p), \quad \mathbf{R}(\mathbf{u}, \mathbf{x}) \quad (p' = p + 1),$$

where $\mathbf{u}$ is the discrete state vector and $\mathbf{x}$ are the coordinates of the mesh nodes.

In the special case $p' = p$, we have the governing DG equations: $\mathbf{r}(\mathbf{u}, \mathbf{x}) = \mathbf{0}$.

Implicit shock tracking: constrained optimization formulation

- Formulate as optimization problem constrained by the discrete PDE

$$\underset{\mathbf{u}, \mathbf{x}}{\text{minimize}} \quad f(\mathbf{u}, \mathbf{x})$$

$$\text{subject to} \quad \mathbf{r}(\mathbf{u}, \mathbf{x}) = \mathbf{0}.$$

- Define a tracking term by penalizing the DG residual in the *enriched test space*

$$f_{\text{err}}(\mathbf{u}, \mathbf{x}) := \frac{1}{2} \mathbf{R}(\mathbf{u}, \mathbf{x})^T \mathbf{R}(\mathbf{u}, \mathbf{x}).$$

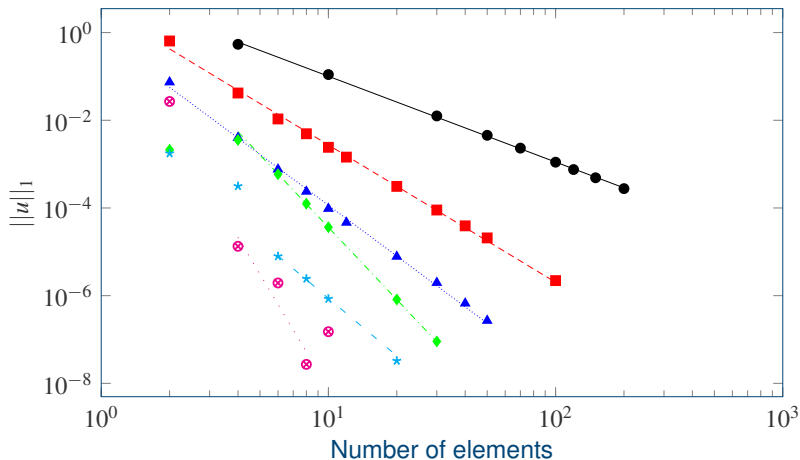
- Define a mesh quality term to account for the distortion of each high-order element

$$f_{\text{msh}}(\mathbf{x}) := \frac{1}{2} \mathbf{R}_{\text{msh}}(\mathbf{x})^T \mathbf{R}_{\text{msh}}(\mathbf{x}).$$

- Use an objective function that *balances* the two terms

$$f(\mathbf{u}, \mathbf{x}) = f_{\text{err}}(\mathbf{u}, \mathbf{x}) + \kappa^2 f_{\text{msh}}(\mathbf{x}).$$

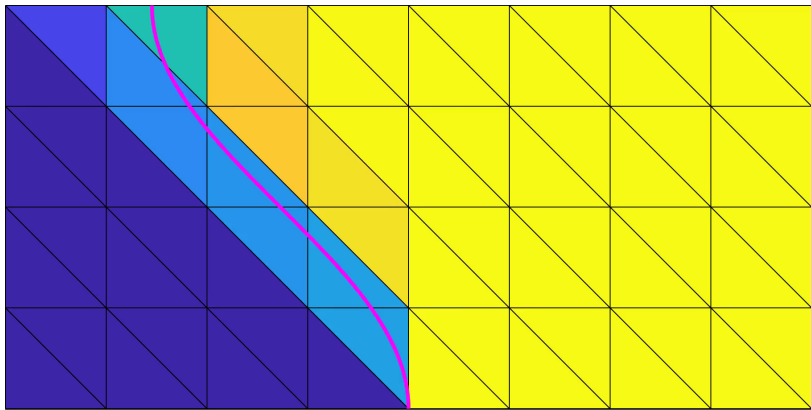
Burger's equation, $\mathcal{O}(h^{p+1})$ convergence rates



Burgers' equation with shock tracking: $p = 1$ (●), $p = 2$ (■), $p = 3$ (▲), $p = 4$ (◆), $p = 5$ (*), $p = 6$ (⊗).

Slopes: $\angle -1.95$ (—), $\angle -3.13$ (---), $\angle -3.85$ (.....), $\angle -5.47$ (-.-.-), $\angle -4.36$ (- - -), $\angle -8.67$ (· · · · ·).

Linear advection, trigonometric shock

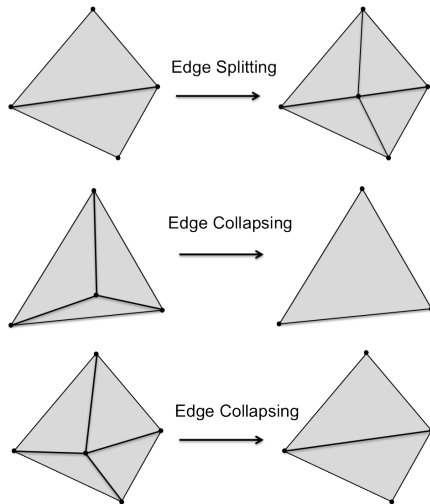
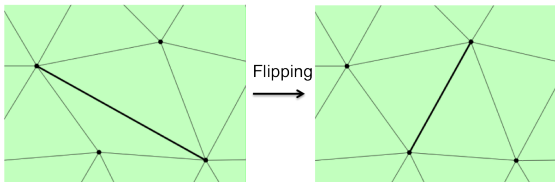


$p = 0$ space for solution, $q = 2$ space for mesh

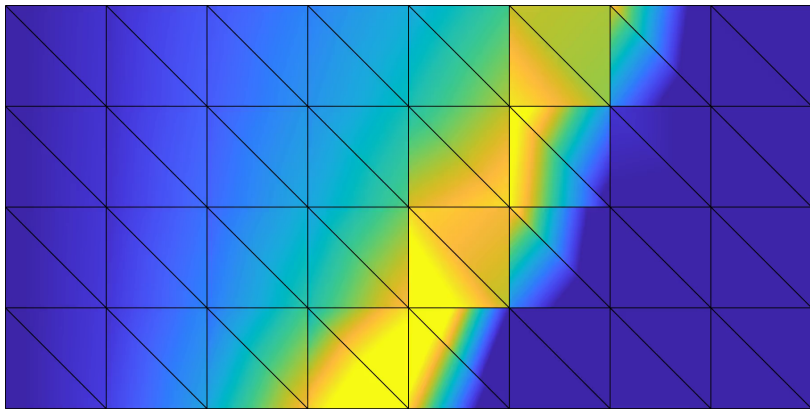
L_1 error = 1.15×10^{-3}

Mesh topology operations

For more general mesh deformations, mesh topology operations such as *edge flips*, *edge splits*, and *edge collapses* are required

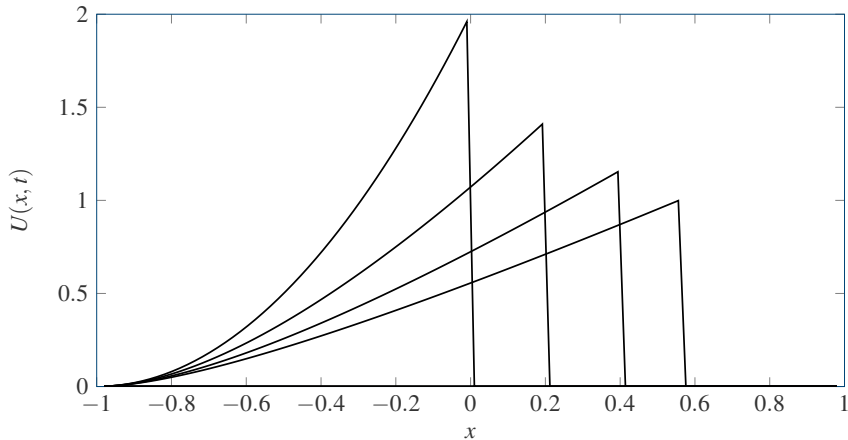


Inviscid Burgers' equation: space-time formulation



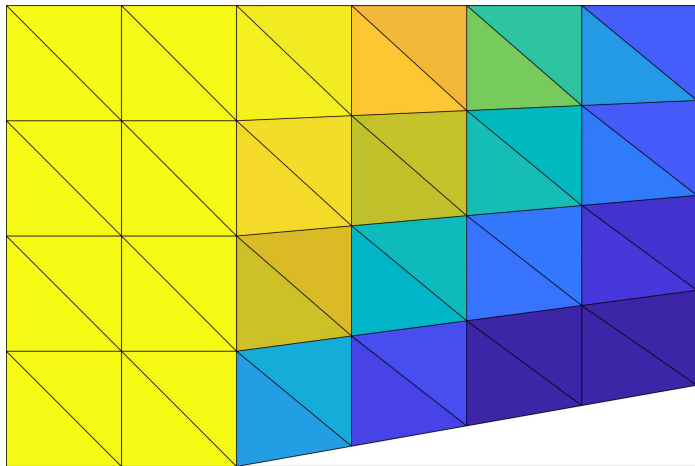
$p = 3$ space for solution, $q = 3$ space for mesh

Inviscid Burgers' equation: Temporal snapshots



Temporal snapshots of inviscid Burgers' equation from $p = 4$
implicit tracking simulation

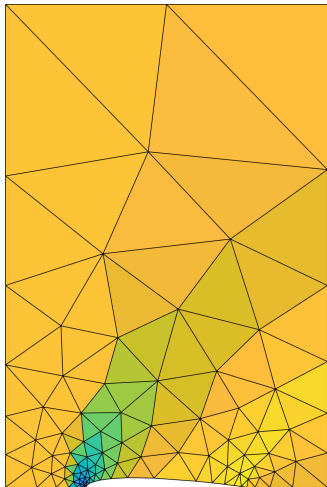
Supersonic flow past wedge ($M = 2$)



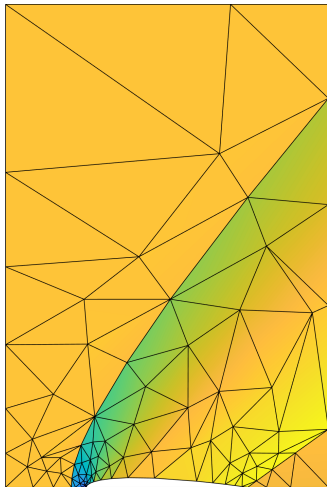
$p = 0$ space for solution, $q = 1$ space for mesh

L^2 stagnation enthalpy error: 7.94×10^{-10}

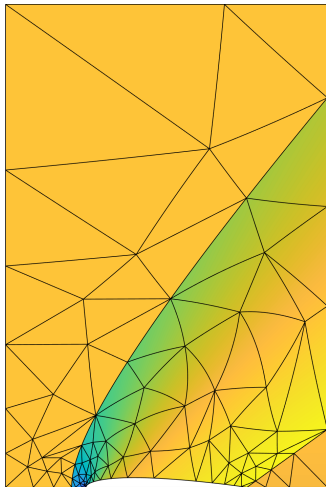
Supersonic flow past NACA0012 airfoil ($M = 1.5$)



Initialization



$p = 1$ tracking
 $e_H = 1.30 \times 10^{-3}$



$p = 2$ tracking
 $e_H = 6.73 \times 10^{-5}$

Supersonic flow past NACA0012 airfoil ($M = 1.5$)



Initialization

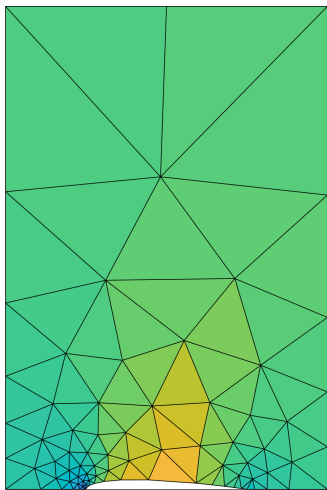


$p = 1$ tracking
 $e_H = 1.30 \times 10^{-3}$

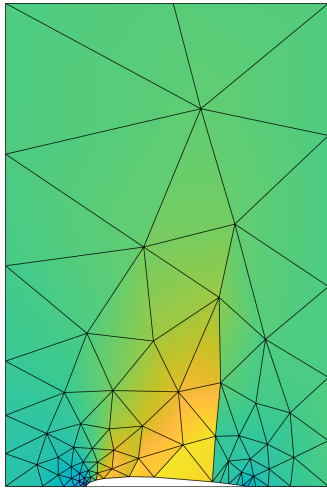


$p = 2$ tracking
 $e_H = 6.73 \times 10^{-5}$

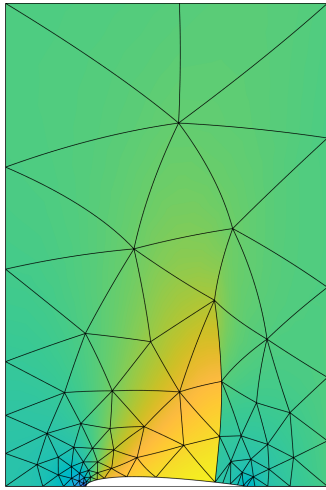
Transonic flow past NACA0012 airfoil ($M = 0.85$)



Initialization



$p = 1$ tracking



$p = 2$ tracking

Transonic flow past NACA0012 airfoil ($M = 0.85$)



Initialization



$p = 1$ tracking



$p = 2$ tracking

Summary

- High-order methods and unstructured meshes required for future simulation software
- New research to find low-cost numerical schemes and more efficient robust solvers
- Applications across engineering disciplines, related to turbulent flows, structures, nonlinear waves, and more
- High-order multiphysics solvers enable accurate fluid-structure interaction and deforming domain simulations.
- Papers and software at <http://persson.berkeley.edu>

