Robust High-Order Implicit Shock Tracking for Complex High-Speed Flows

Charles Naudet (cnaudet@nd.edu), Tianci Huang (thuang5@nd.edu), and Matthew J. Zahr (mzahr@nd.edu)
Department of Aerospace and Mechanical Engineering; University of Notre Dame

Abstract

Shock tracking, as an alternative method to shock capturing, moves the computational mesh to align with the shock and has the potential to obtain high-order accuracy without extensive mesh refinement around the discontinuity. However, an aligned mesh is required to achieve such advantage, which is a challenge. In previous work, we introduced an implicit shock tracking framework that solves an optimization problem whose solution is a mesh that aligns with discontinuities and the corresponding flow solution, which does not require *a priori* knowledge of the discontinuities. Here, we present improvements to the robustness of the optimization solver and to the mesh motion that enable the solution of complex flows on high-order meshes. This is achieved by: (1) introducing shock-preserving, arbitrary-order element collapse in arbitrary dimensions to preserve solution features; (2) integrating solution re-initialization strategy for oscillatory elements to promote good mesh motion; and (3) developing a time-slab approach for solving space-time problems so that moving shocks can be tracked simultaneously in space and time. We demonstrate the robustness of the method using a suite of two- and three-dimensional flows.

Problem Formulation

- As introduced in [3] and improved in [4], we develop a PDE-constrained optimization framework to find solution to conservation laws with discontinuities, without the need to know or first solve for shock location.
- We seek for the discretized DG solution $\boldsymbol{u}$ along with the nodal positions of the computational mesh $\boldsymbol{x}$ that minimizes an objective function $f(\boldsymbol{u},\ \boldsymbol{x})$ while satisfing the discretized conservation law $\boldsymbol{r}(\boldsymbol{u},\ \boldsymbol{x}) = 0$

$$\underset{\boldsymbol{u},\,\boldsymbol{x}}{\text{minimize}} \quad f(\boldsymbol{u},\,\boldsymbol{x}) \qquad \text{subject to} \quad \boldsymbol{r}(\boldsymbol{u},\,\boldsymbol{x}) = 0,$$

where

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

- $f_{\text{err}}(\boldsymbol{u},\boldsymbol{x})$ is a measure of the DG solution error, $f_{\text{msh}}(\boldsymbol{x})$ penalizes the distortion of the mesh, and κ is a weight parameter.
- The error term of the objective function is the norm of the one-degree enriched DG residual $\boldsymbol{R}$

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

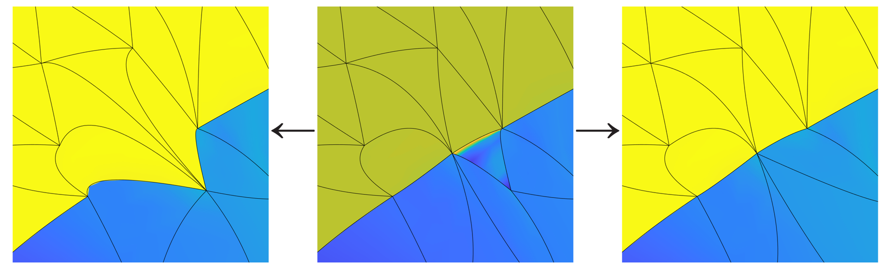
that is to use one degree higher polynomials to approximate equations (*test space*) than to approximate solutions (*trial space*). This adds additional constraints to the optimization: a solution $\boldsymbol{u}$ can satisfy $\boldsymbol{r}(\boldsymbol{u},\boldsymbol{x}) = \boldsymbol{0}$ with a non-aligned mesh but will likely not minimize $\boldsymbol{R}(\boldsymbol{u},\boldsymbol{x})$.
- f_{msh} prevents the mesh to be overly-askew, while f_{err} promotes mesh alignment as an aligned mesh should result in a much lower solution residual.

Mesh Operation

During the optimization, mesh nodes are being moved to track the shock. Elements near a shock will be squeezed towards the shock and result in small and askew elements, which are problematic as they can cause singularities and crash codes.

Shock-Aware Element Collapse

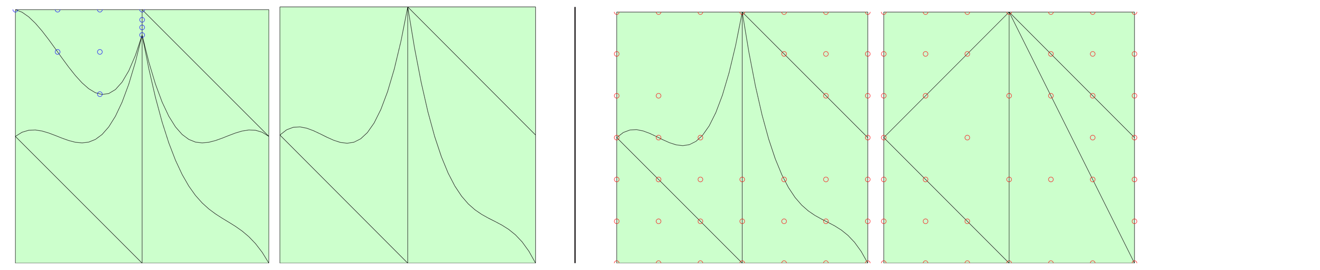
- Small elements are removed by collapsing their shortest edge. For high-order elements, the edge lengths are calculated based on the end-nodes.
- The edge collapse is performed by merging one end-node to the other, where the choice of the moving end-node ensures the mesh always conforms to the domain boundaries.
- Besides the boundary constraints, we add an additional constraint for node movement based on solution value jump to ensure nodes on a shock do not move away from the shock. This is based on the logic that if a node lies on a shock, then the solution values on the node should see a larger value jump than a node that is away from the shock.



Initial mesh (*middle*), after a non-preserving collapse (*left*) and after a shock-preserving collapse (*right*).

Element Smoothing

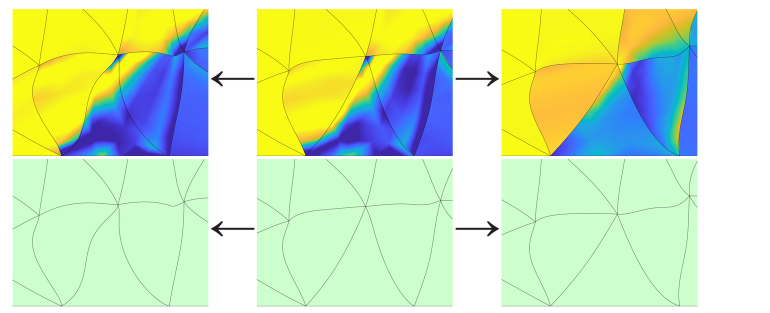
- While the mesh is being moved, and particularly after element collapses, elements can become singular or near-singular. This situation occurs more frequently when using high order meshes.
- For poorly shaped elements, we smooth the high order nodes by replacing the curved elements with their corresponding straight-sided version.



The left group of images shows a $p = 3$ element collapse (*from left to right*) where the collapsed element has its nodes labeled. The right group of images shows the smoothing process of the mesh after the collapse.

Solution Re-Initialization

- When using high-order approximations, the element-wise solution can become oscillatory, which leads to poor search directions for the mesh motion.
- We observe that a piecewise constant solution promotes good mesh motion, thus, we replace the element solution that is highly oscillatory with the solution average.



Example of oscillatory solution (*top*) and mesh (*bottom*). From middle to the left is after a step *without* re-initialization, while to the right is a step *with* re-initialization.

Space-time High-Order Tracking using Time Slabs

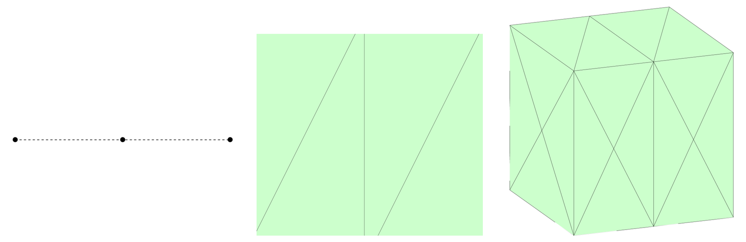
Resolving fine solution details for problems of higher than one-dimension over the entire space-time domain is very difficult. Our approach is to break the time-dimension into smaller bite-sized chunks, or "slabs", as seen in [1] and [2]. We then solve for the solution on each individual time slab, starting with the slab at the initial time, and stack each one on top of the previous.

Time Slab Requirements

- *Inter-Slab Conformity:* For N time slabs, the domain is defined as $\chi_n = \Omega \times [t_n, t_{n+1}]$, where χ_n is the space-time domain for the nth time slab, Ω is the spatial domain, and t_n and t_{n+1} are the initial and end times of the nth slab, respectively. The boundary at the top of the nth slab $\chi_n|_{t_{n+1}}$ must conform to the boundary at the bottom of the $n+1$th slab $\chi_{n+1}|_{t_{n+1}}$ in both topology and solution space, for $n = 1, \dots, N-1$.
- *Initial Condition for each Time Slab:* The topology and solution space from the top boundary of a previous slab $\chi_{n-1}|_{t_n}$ is held fixed as a boundary condition for the bottom of the nth time slab on the boundary defined as $\chi_n|_{t_n}$ so that features in the solution can be preserved and tracked between time slabs.

Extrusion as a Method for Time Slabs

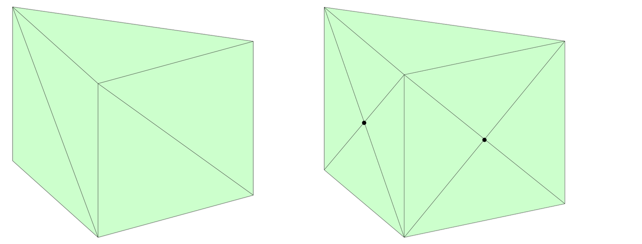
- One approach to create time slabs is to take the spatial domain Ω and to *extrude* it into the space $[t_n, t_{n+1}]$. This is done by taking a tensor product of the space and time domains $\chi_n = \Omega \bigotimes [t_n, t_{n+1}]$.
- Extruding an M-dimensional simplex with $F_M = M + 1$ faces into a new dimension creates a $(M+1)$-wedge geometry, with $2 + F_M$ faces, where two faces are simplices and the remaining F_M faces are M-dimensional wedge geometries. Wedge geometries must be split or decomposed to form simplices.



1D elements (*left*), extrusion/splitting of 1D elements to 2D (*middle*), extrusion/splitting of 2D elements to 3D (*right*).

Extrusion Considerations

- The decomposition of topology of the face of an element must conform to its neighboring element for extrusion. This is simple in two-dimensions because the faces of a 2D simplex are lines, thus, connecting existing vertices forms conforming simplices.
- The trivial decomposition of the above will not satisfy the conforming face rule for higher dimensions. We must add nodes to achieve a uniform face.



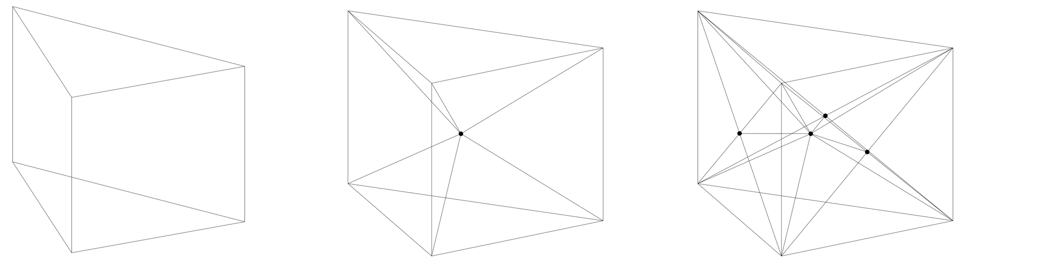
Extrusion/splitting of a 2D simplex to a 3D wedge: non-conforming (*left*), conforming (*right*).

Background and Motivation

- The problem of numerically finding the solution of conservation laws when the solution contains discontinuities or sharp gradients is difficult.
- Challenges arise within this situation as polynomial basis is used to approximate discontinuous features, which is likely to exhibit Gibbs' phenomena.
- Numerical dissipation can be added to smooth out the sharp features to make the approximation feasible, but can require large amount of refinement.
- The intrinsic inter-element solution discontinuity of the discontinuous Galerkin (DG) methods provides advantages for approximating discontinuous features with a mesh that aligns with such features.

Extrusion and Splitting in Four Dimensions

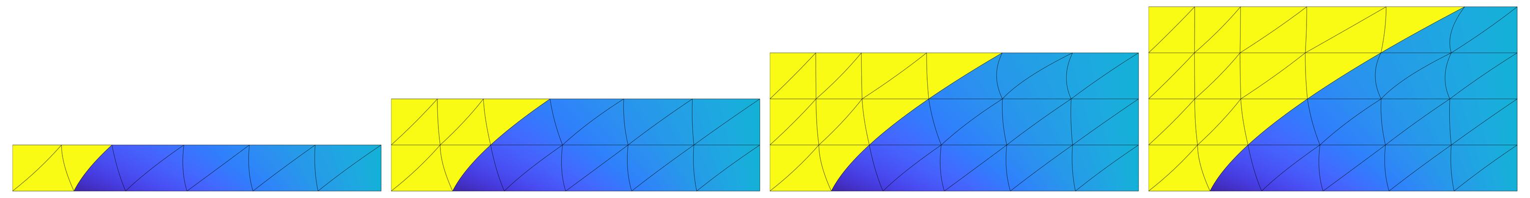
- Using properties of extrusion that we have previously defined, we formalize a strategy to extrude into 4D space and split wedges into simplices so that the topology of the 3-D faces between elements conform.
- Upon extruding a 3-simplex into 4D space there are six faces (two 3-simplices and four 3D wedges). We look at one 3D wedge face for simplicity, as the process will apply to each. We first add one node to the center of the 3D wedge and define two 3-simplex caps and three pyramids. We then add a node to the center of each of the 2D faces to decompose each pyramid into 3-simplices. Once each 3D wedge is axisymmetrically decomposed into 3-simplices, we define edges from each vertex to a new vertex in the center of the original 4D wedge. This defines 4-simplices from the 3-simplices while each face of the original 4D wedge is axisymmetrically decomposed for conformity.
- This means that each 4-simplex will be split into 58 4-simplices (2 |simplex caps| + 4 |3D wedge faces| · 14 |simplices decomposed per wedge face|). This is a general procedure that works in up to four dimensions.



Decomposition of 3D wedge face of a 4D wedge: Original 3D wedge (*left*), decomposition into two 3D simplices (*center*), and decomposition of remaining pyramid faces into 3D simplices (*right*).

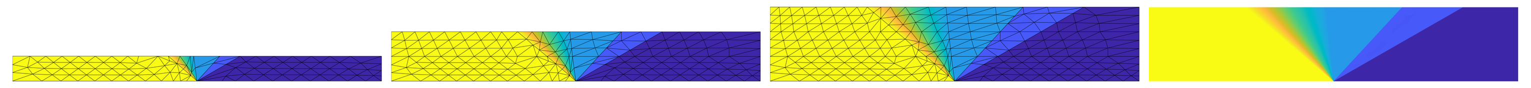
Numerical Example: Space-Time With Time-Slabs

1D Space-Time Burger's Equation



Burger's equation in 1D space and time for four time slabs, stacked on each other.

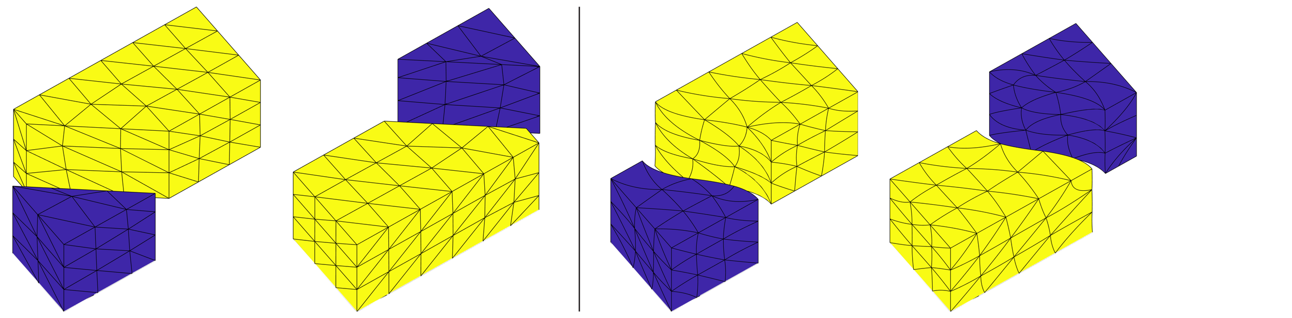
1D Space-Time Sod's Shock Tube



Sod's shock tube in 1D space and time for three time slabs, stacked on each other and the solution without the mesh (*right*).

Numerical Example: Linear Advection

- In this example, we simulate linear advection with piecewise constant solution, and we consider (1) a straight shock surface and (2) a trigonometric surface.

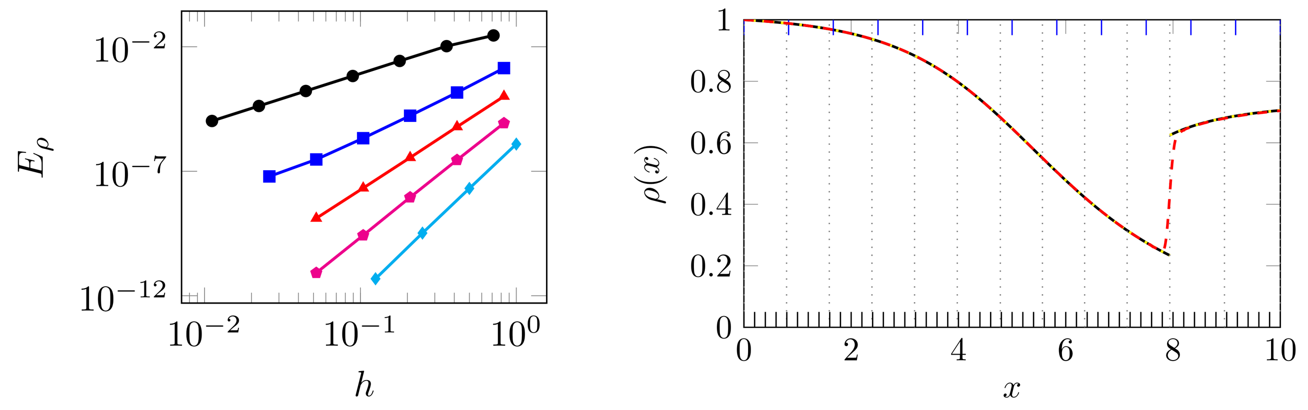


3D advection with straight shock (*left*) and curved shock (*right*) after tracking. Mesh is pulled apart from the shock surface.

Numerical Example: Euler Equations

Inviscid, compressible flow through area variation

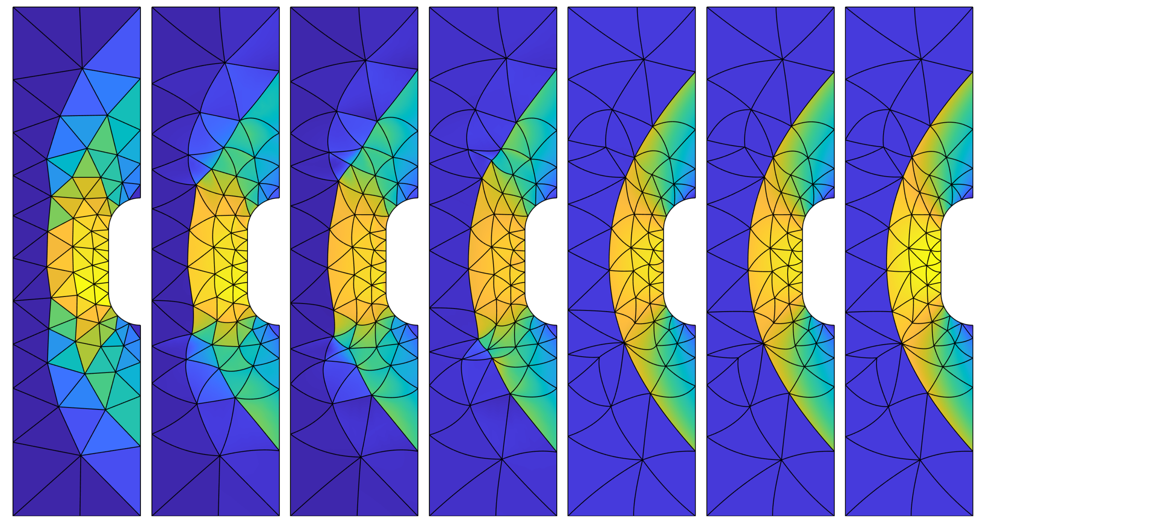
- We demonstrate our framework's optimal convergence rates with a $h-$ and $p-$convergence study.



1D Euler nozzle convergence study with $p = 1, ..., 5$ where h is the mesh size and E_ρ is the solution error (*left*), and the shock tracking solution comparing to shock capturing solution (*right*).

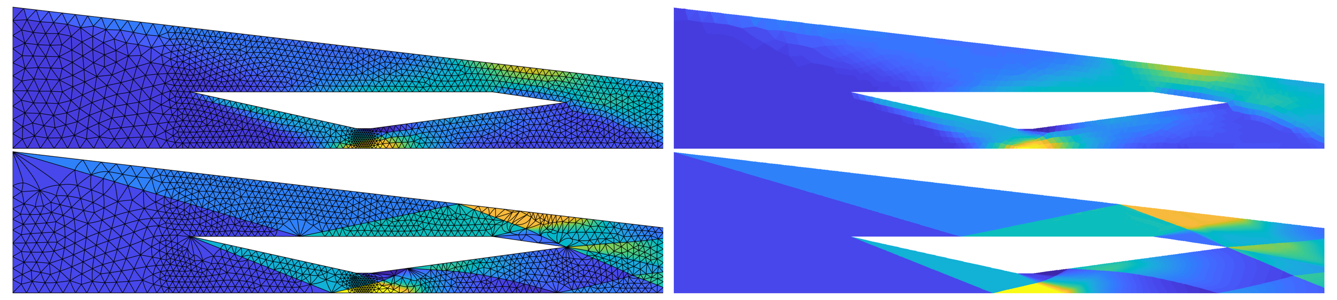
Inviscid, compressible flow over a smooth square

- We simulate supersonic flow ($M = 4$) over a smooth square using $p = 2$ solution and $q = 2$ mesh. From left to right are: the initial guess, the solution at $2, 4, 6, 9, 12, 15$ iterations. The initial guess is the $p = 0$ solution to the fixed-mesh.



Inviscid, compressible flow through scramjet intake

- The geometry here is the upper symmetric half of a scramjet intake, where the left boundary is the inlet. We simulate a $M = 5$ supersonic flow with $p = 2$ solution and $q = 2$ mesh.
- The first row shows the initial guess, and the second row shows the result after tracking.



Conclusions

We have extended our previous framework to deal with more complex flow problems, and we have greatly improved robustness. We have also developed a mesh extrusion strategy for space-time problems to solve for solutions in space and time simultaneously.

References

[1] Andrew Corrigan, Andrew D. Kercher, and David A. Kessler. A moving discontinuous galerkin finite element method for flows with interfaces. *International Journal for Numerical Methods in Fluids*, 89(9):362–406, 2019.

[2] Andrew T. Corrigan, Andrew Kercher, and David A. Kessler. *The Moving Discontinuous Galerkin Method with Interface Condition Enforcement for Unsteady Three-Dimensional Flows.*

[3] Matthew J. Zahr and Per-Olof Persson. An optimization-based approach for high-order accurate discretization of conservation laws with discontinuous solutions. *Journal of Computational Physics*, 365:105 – 134, 2018.

[4] Matthew J. Zahr, Andrew Shi, and Per-Olof Persson. Implicit shock tracking using an optimization-based high-order discontinuous Galerkin method. *Journal of Computational Physics*, 410, 2020.

Acknowledgments

This material is based upon work supported by AFOSR, award number FA9550-20-1-0236.