A Method for Bounding High-Order Functions + Recent Developments in High-Order Mesh Optimization

A Method for Bounding High-Order Finite Element Functions. arXiv:2504.11688
PDE-Constrained High-Order Mesh Optimization: arXiv:2507.01917

MFEM Community Workshop
10-11 September 2025

CASC
Center for Applied Scientific Computing

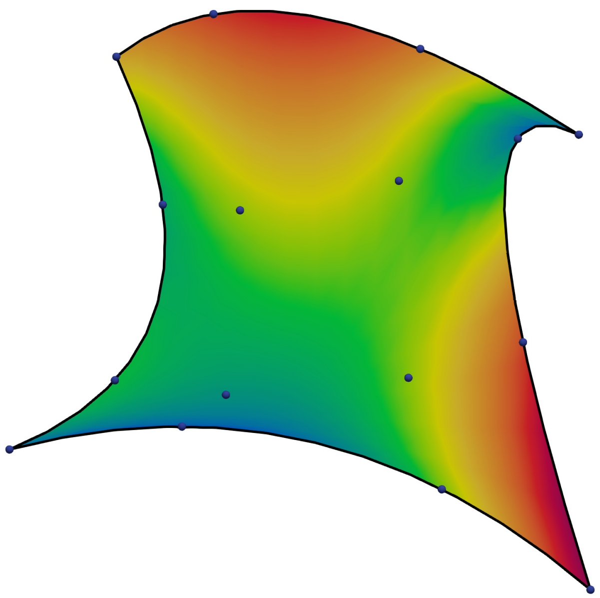
Ketan Mittal
(Bounding) Tarik Dzanic, Tzanio Kolev
(Meshing) Veselin Dobrev, Pat Knupp, Boyan Lazarov, Mathias Schmidt, Vladimir Tomov, Tzanio Kolev

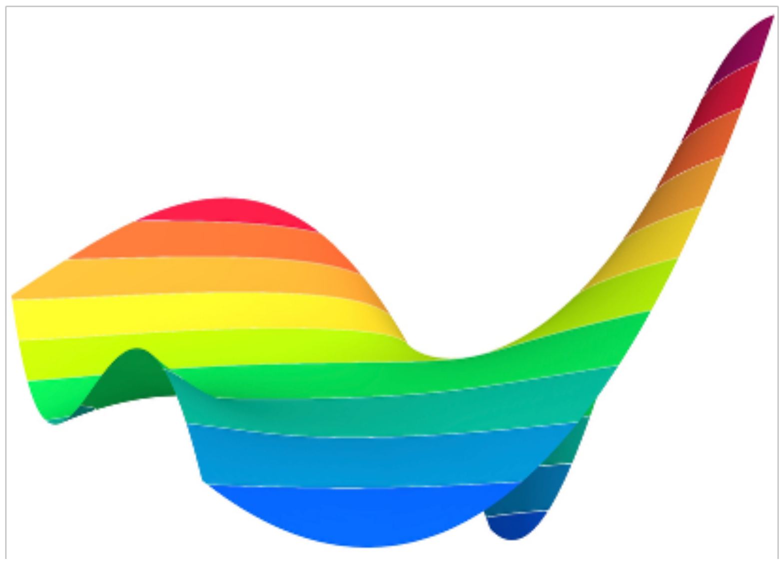
LLNL-PRES-2011018
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

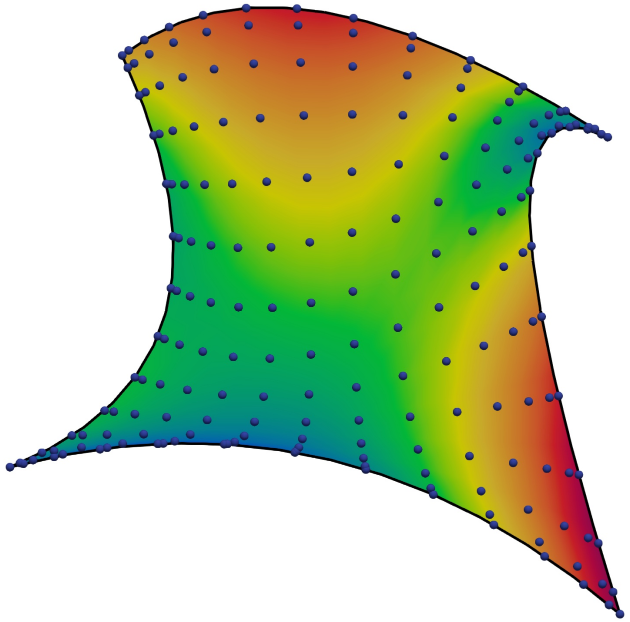
Motivation

- Computing extrema of high-order functions is not trivial.





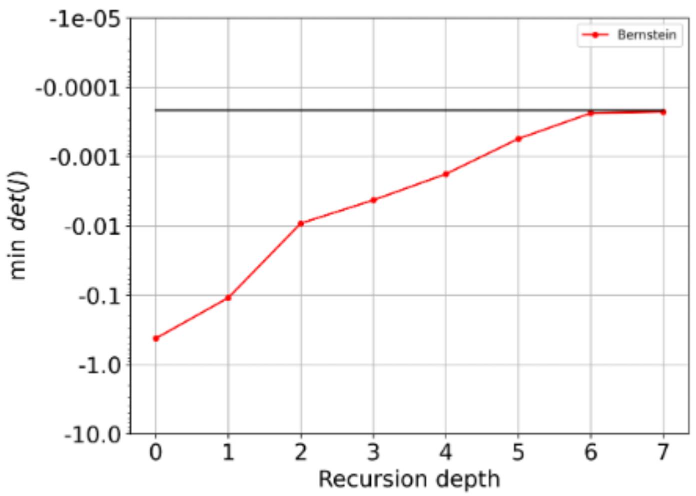
Cubic function for the determinant of the Jacobian for a quadratic element.



GLL quadrature points associated with a 26th order integration rule also fails to detect negative `det(J)`.

- Sampling is expensive and not robust.

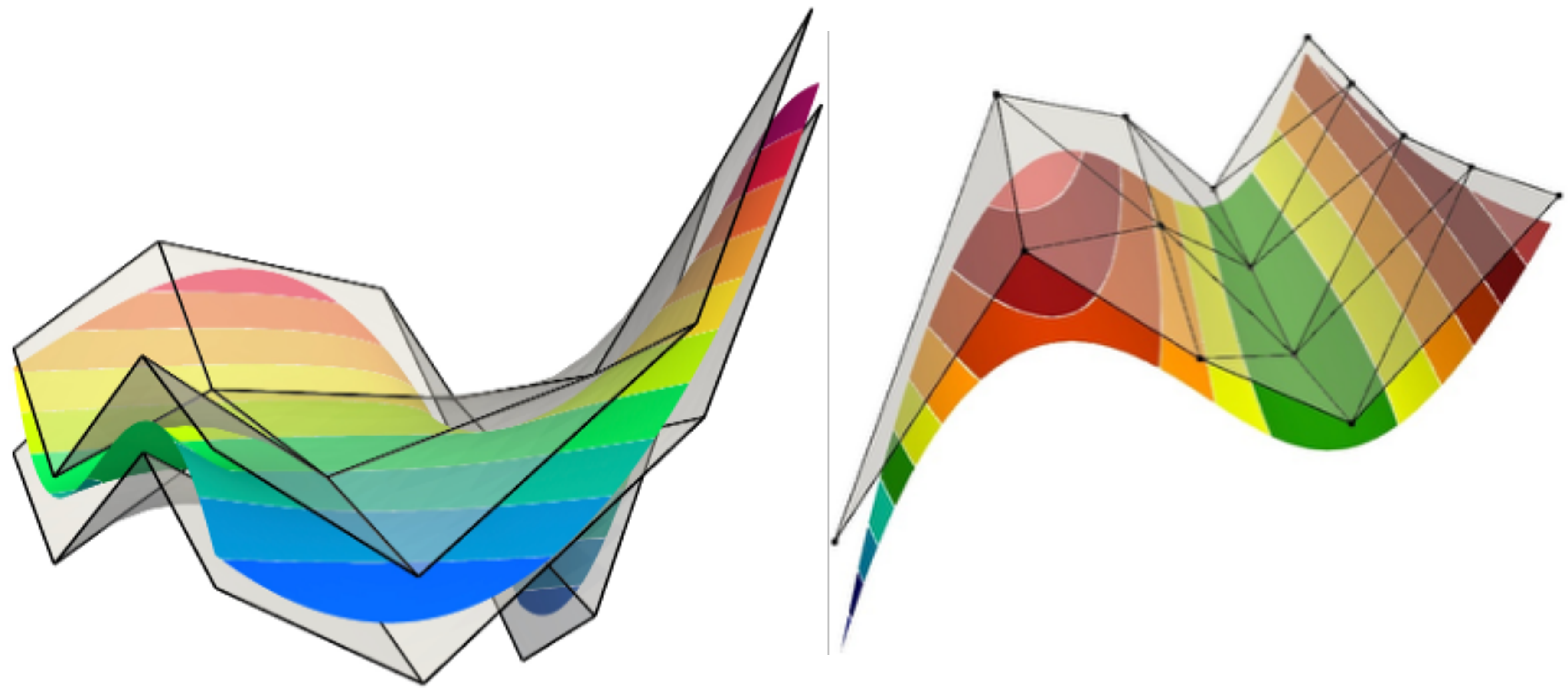
- Bernstein bases give us rather loose bounds. The minimum bound estimate for `det(J)` starts at -0.42 here.



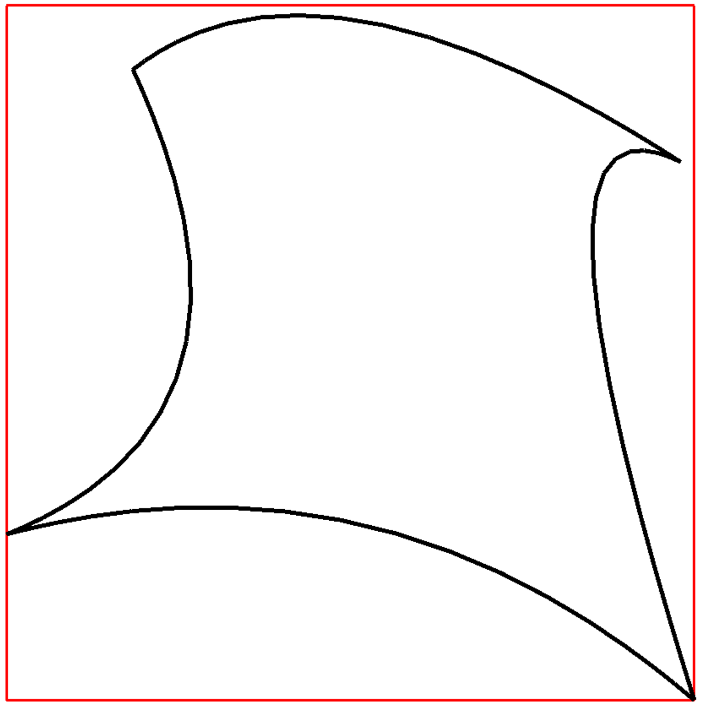
Minimum `det(J)` estimated using Bernstein coefficients.

Proposed Solution

- We construct piecewise linear envelope around a given function using a relatively cheap and robust technique with user-tunable compactness.
  - Based on technique developed by James Lottes in findpts/gslib.



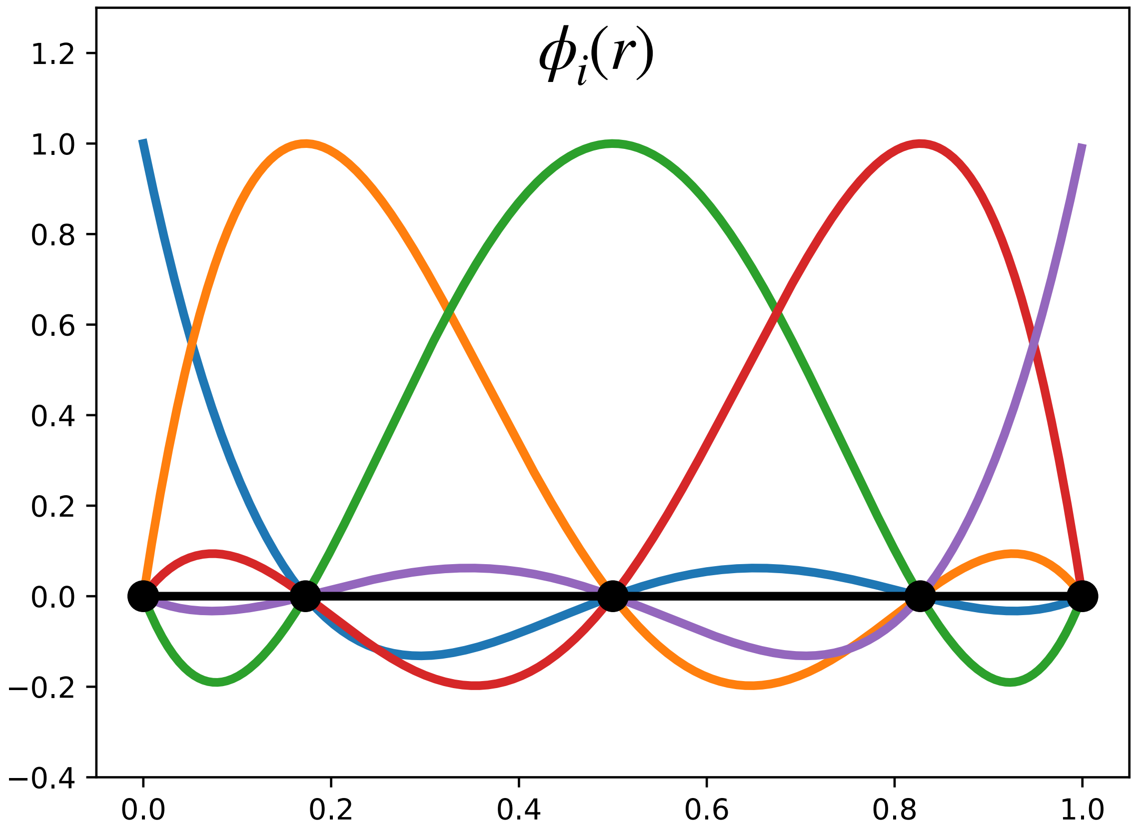
Piecewise linear bounds for a high-order function on a quadrilateral and triangle.



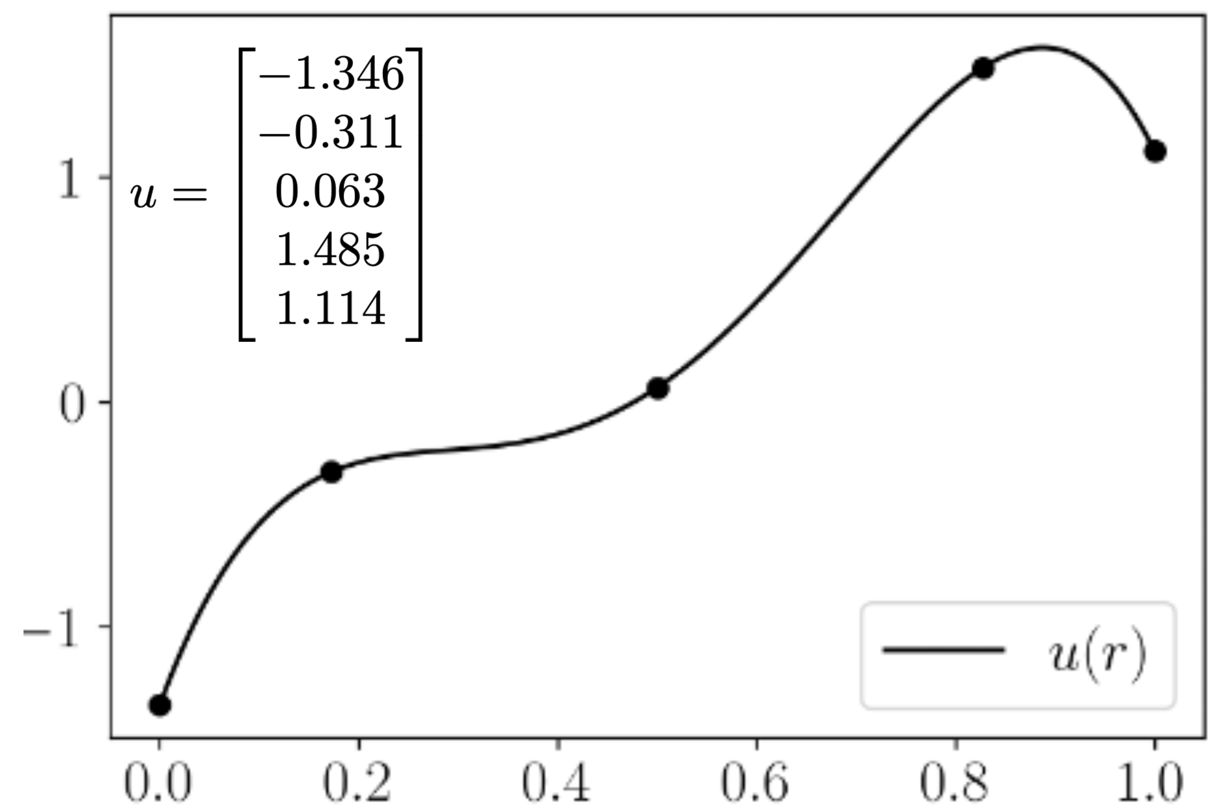
Bounding box around a quad.

High-Order Function Representation

$$u(r) = \sum_{i=1}^{N} u_i \phi_i(r)$$



4th-order Lagrange bases on N=5 GLL nodes.



4th-order function defined using Lagrange bases

Bounding a High-Order Function

- Use piecewise linear bounds of the bases to bound the function.

$$\underline{\phi}_{i,\boldsymbol{\eta},\mathbf{q}}(r) \leq \phi_i(r) \leq \overline{\phi}_{i,\boldsymbol{\eta},\mathbf{q}}(r)$$

$$\underline{q}_{ij} := \underline{\phi}_i(\eta_j) \qquad \bar{q}_{ij} := \bar{\phi}_i(\eta_j)$$

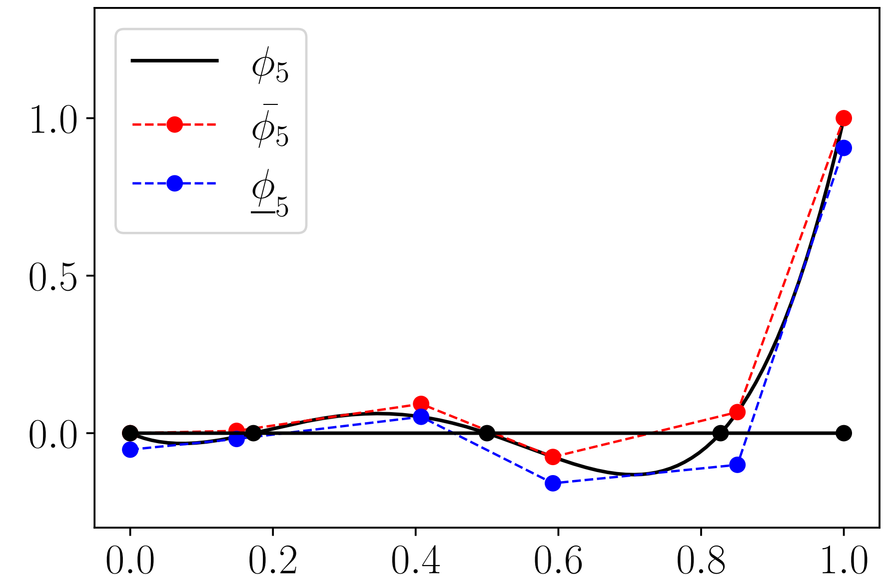
$$u(r) = \sum_{i=1}^{N} u_i \phi_i(r)$$

$$\bar{u}(\eta_j) = \sum_{i=1}^{N} \max\{u_i \underline{q}_{ij}, u_i \bar{q}_{ij}\}$$

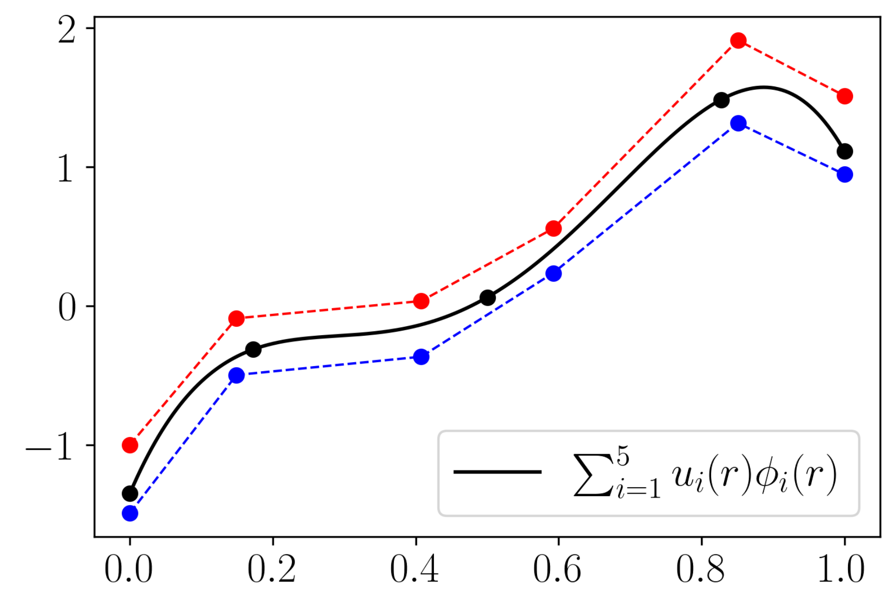
$$\underline{u}(\eta_j) = \sum_{i=1}^{N} \min\{u_i \underline{q}_{ij}, u_i \bar{q}_{ij}\}$$

$$\underline{u} \leq u \leq \overline{u}$$

Cost is $\mathcal{O}(N \cdot M)$



$$u = \begin{bmatrix} -1.346 \\ -0.311 \\ 0.063 \\ 1.485 \\ 1.114 \end{bmatrix}$$



Generalization of the Bounding Approach

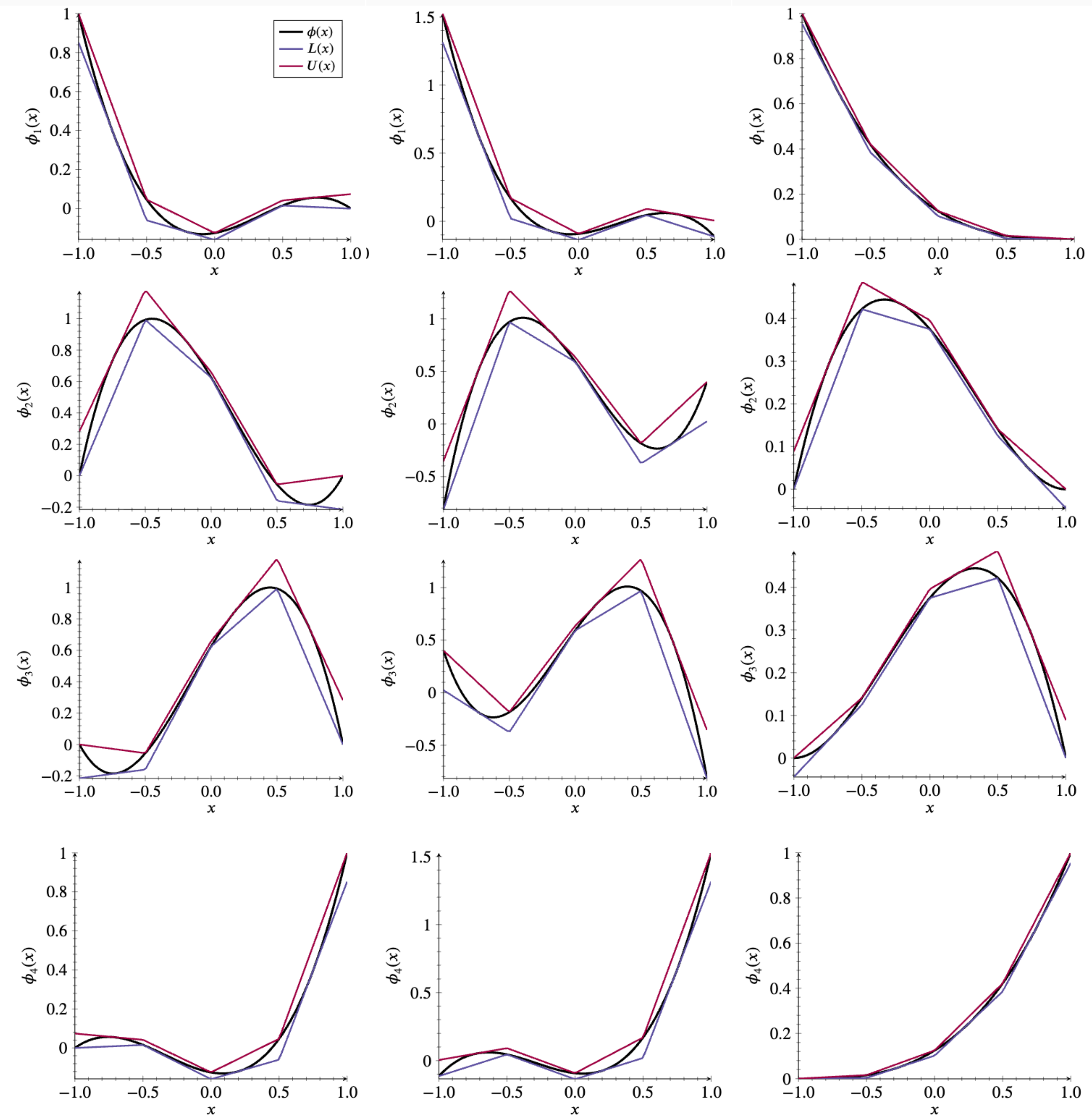
- Works for different bases.
- Works for different element types in higher dimensions.
- Lower compute cost for tensor-product bases:

$$u(r) = \sum_{i=1}^{N} \sum_{j=1}^{N} u_{ij} \; \phi_j(s) \; \phi_i(r)$$

$$u(r) = \sum_{i=1}^{N} \underbrace{\sum_{j=1}^{N} u_{ij} \; \phi_j(s)}_{v_i} \; \phi_i(r)$$

$$v_i \in [\underline{v}_i, \overline{v}_i]$$

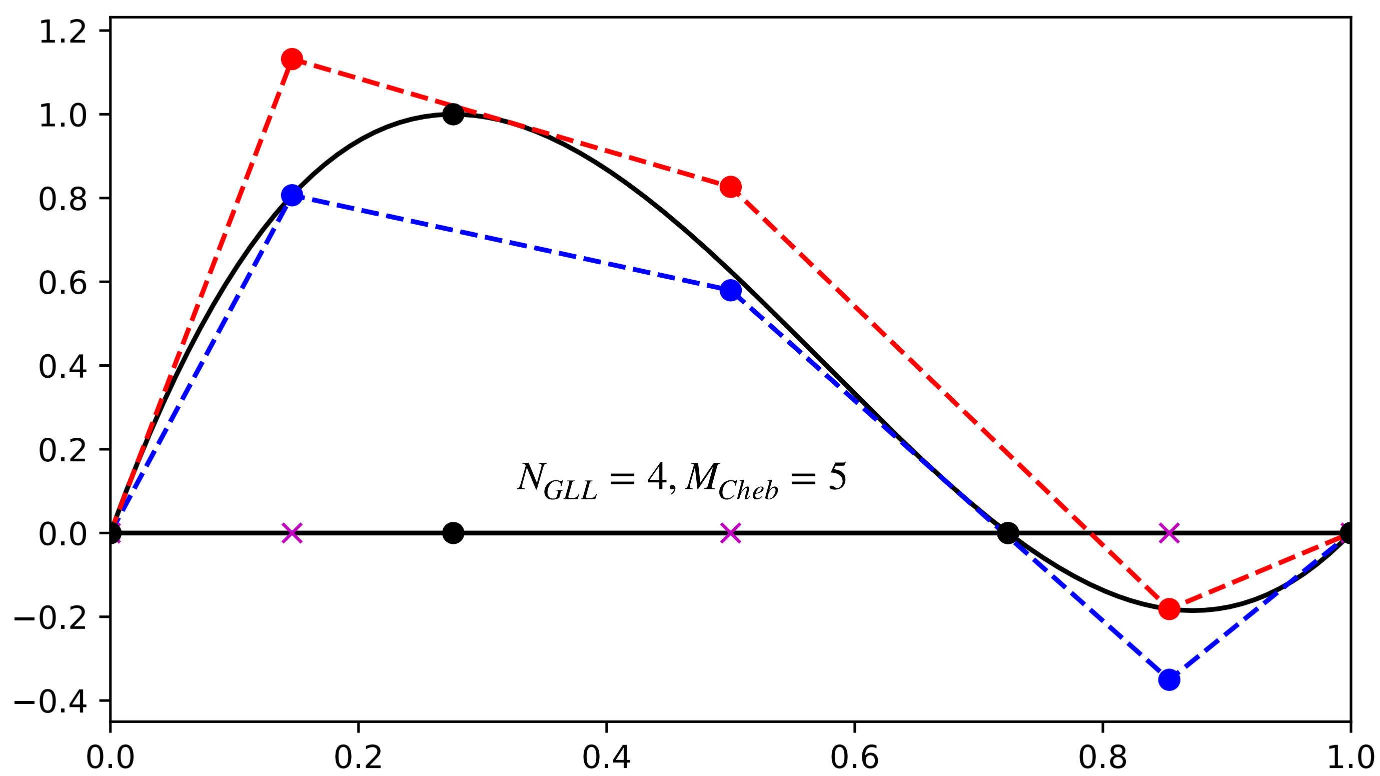
Cost is $\mathcal{O}(N^D \cdot M + N \cdot M^D) \approx \mathcal{O}(N^{D+1})$



(a) Gauss–Lobatto basis (b) Gauss–Legendre basis (c) Bernstein basis

Computing Piecewise Linear Bounds of Bases

- Simple numerical recipe using bases and their derivatives.

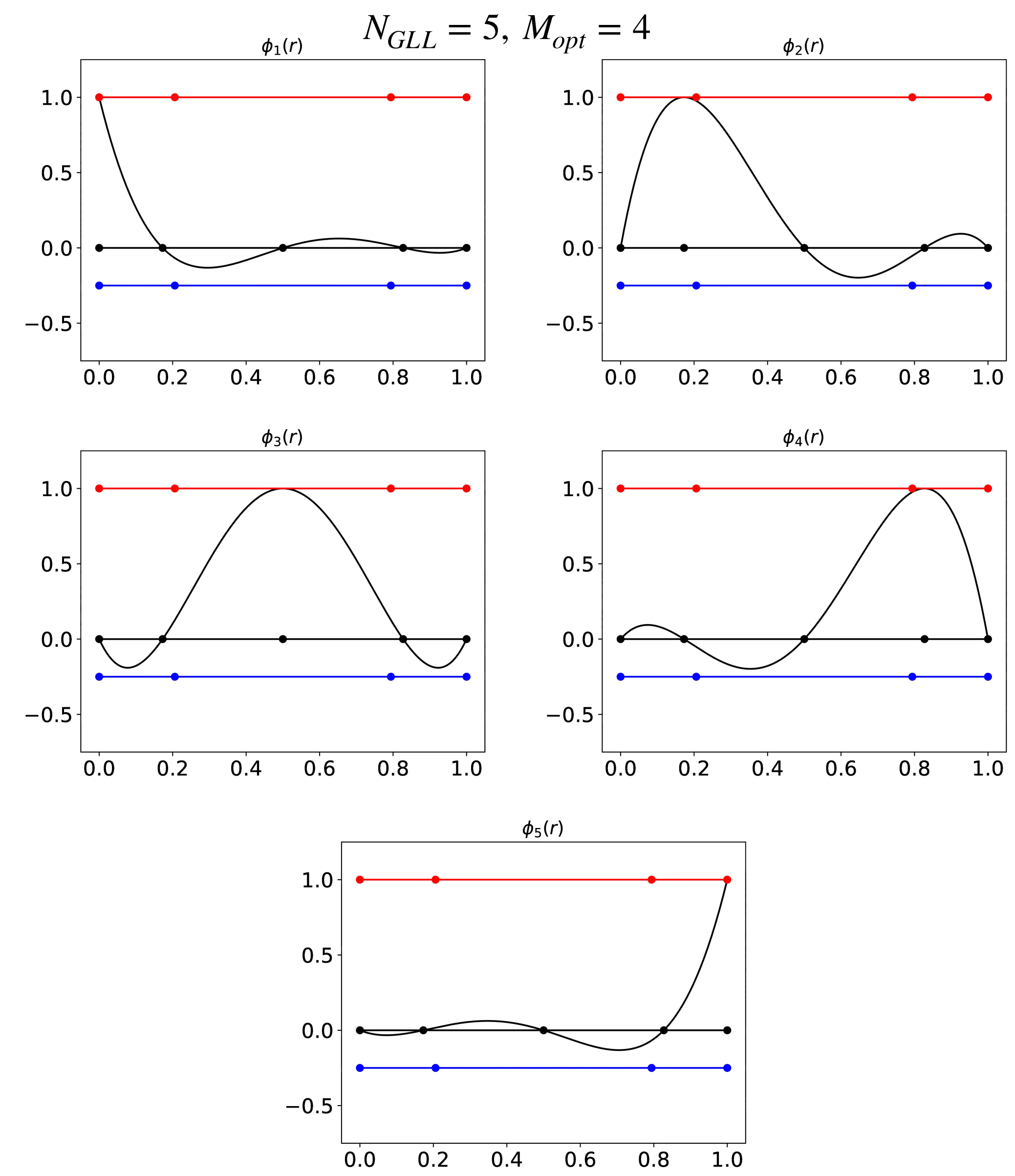


N_{GLL}	M_{Cheb}	$M_{\text{GL+End}}$	M_{GLL}
3	4	4	6
4	7	6	9
5	9	7	12
6	10	8	15
7	12	9	18
8	14	11	21
9	16	12	23
10	17	13	26

General Field Evaluation in High-Order Meshes on GPUs, Computers & Fluids (2025).

Computing Piecewise Linear Bounds of Bases

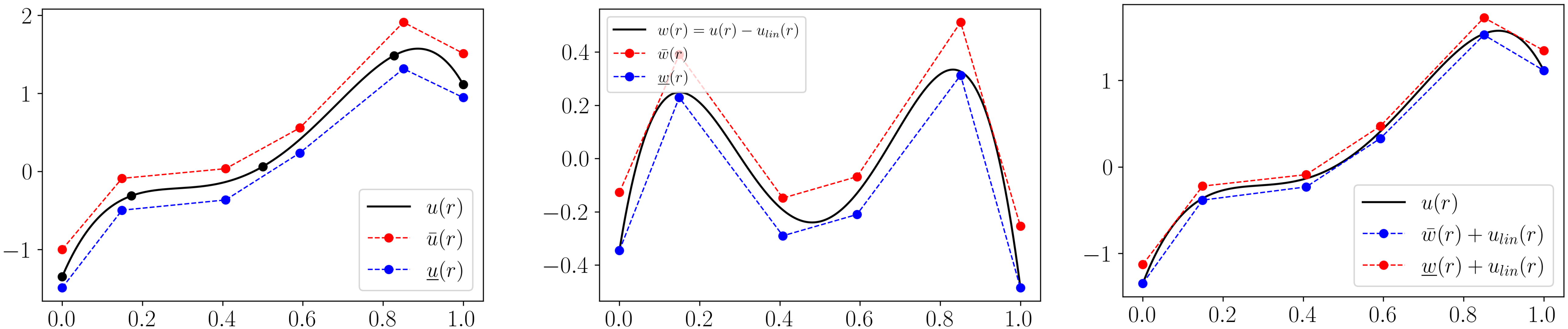
- Optimization-based approach
  - Works for any number of control points
  - Run offline once and store the bounding matrices for (N, M) pairs.



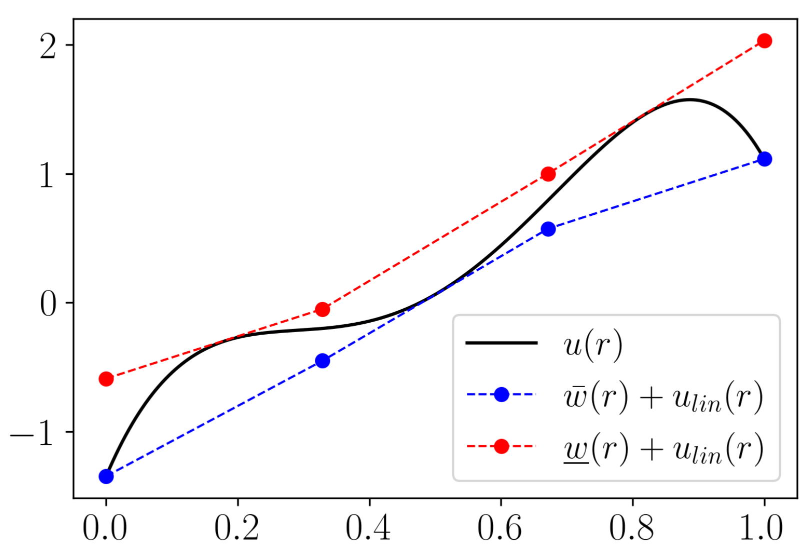
A method for bounding high-order finite element functions: Applications to mesh validity and bounds-preserving limiters, arXiv: 2504.11688.

Effectiveness of Bounding

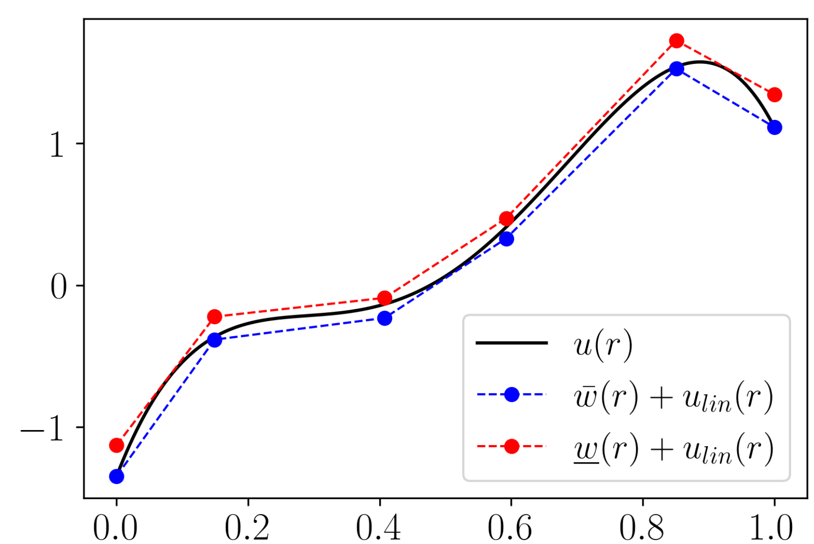
- Linear fit offset to increase effectiveness



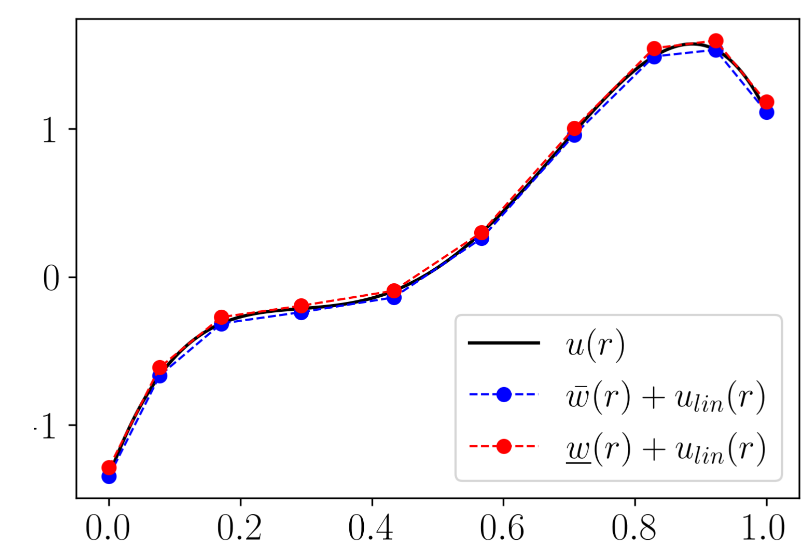
- User tunable compactness



M=4

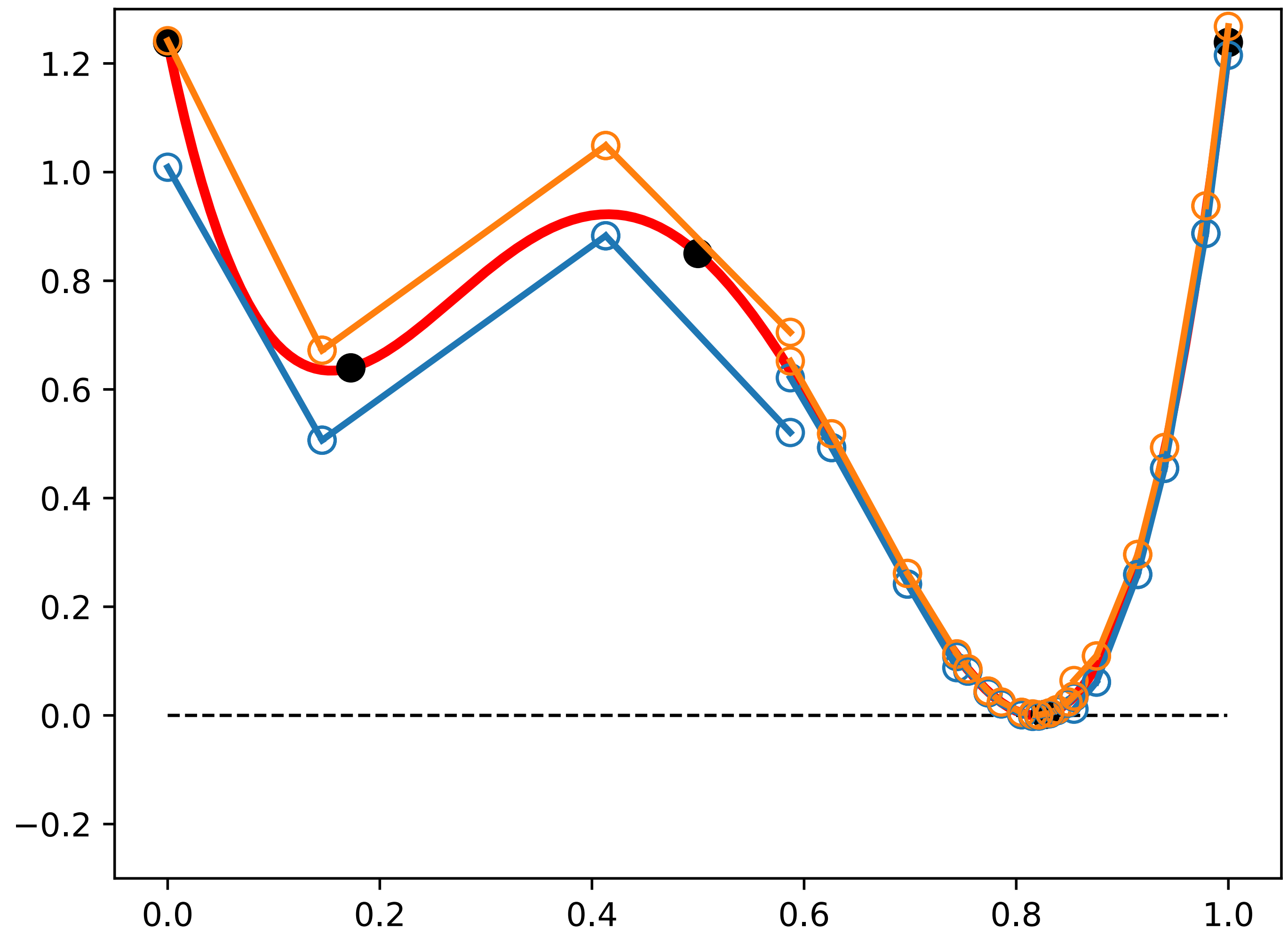


M=6



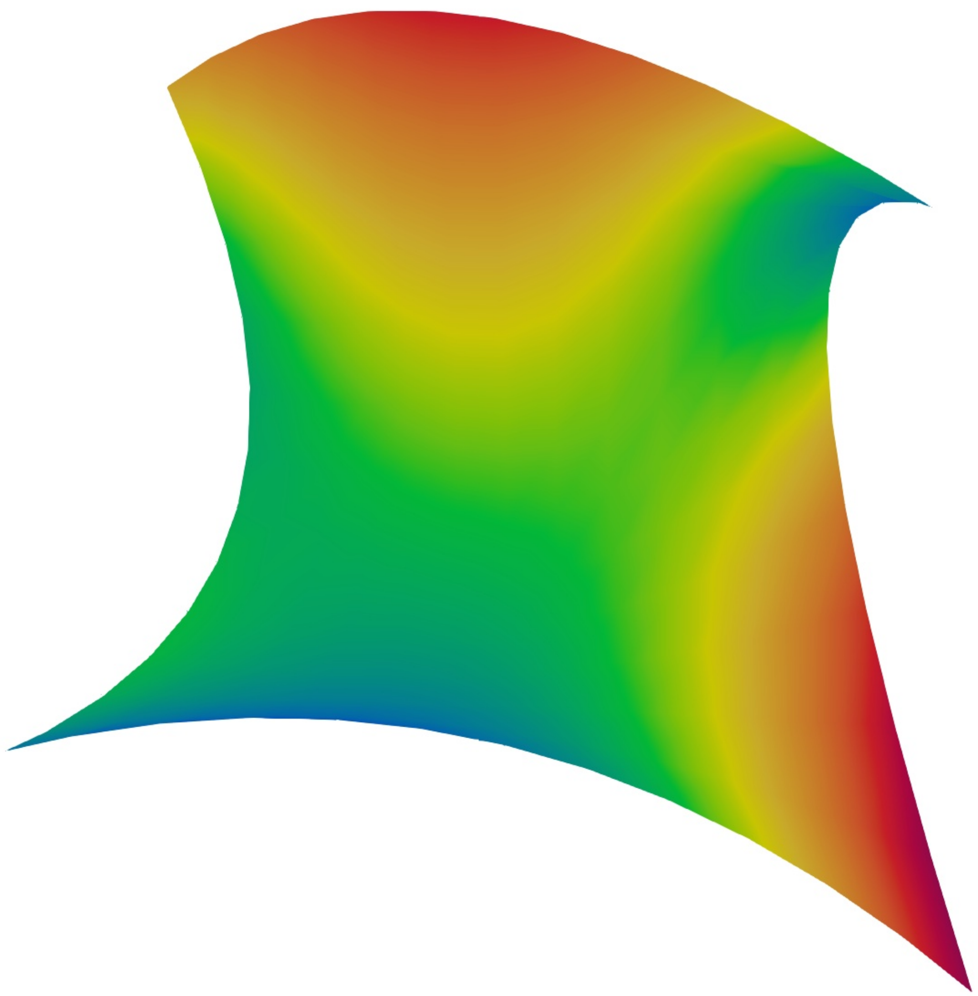
M=10

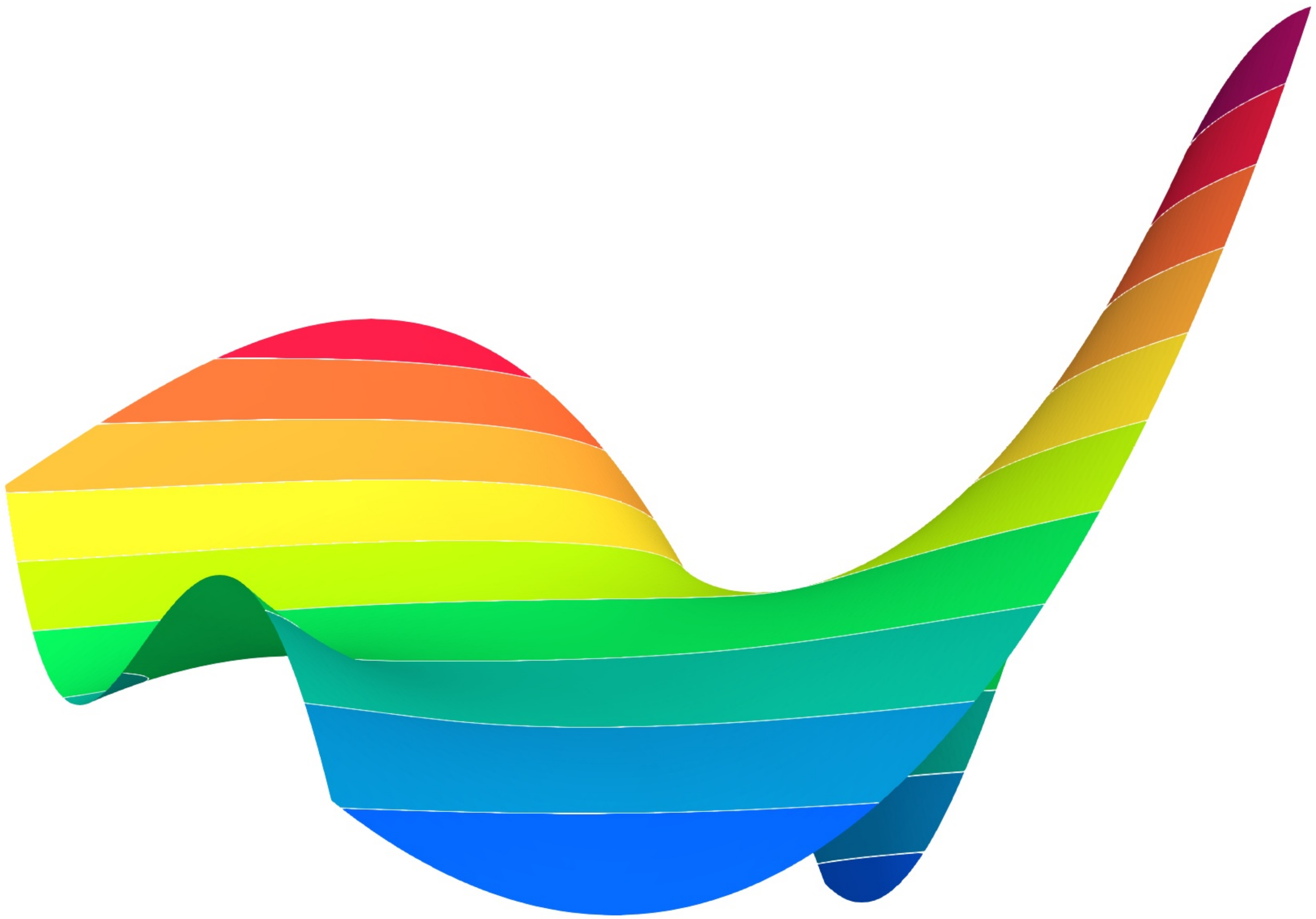
Determining Mesh Validity - 1D Example



Piecewise linear bounds, $N = 5, M = 6$

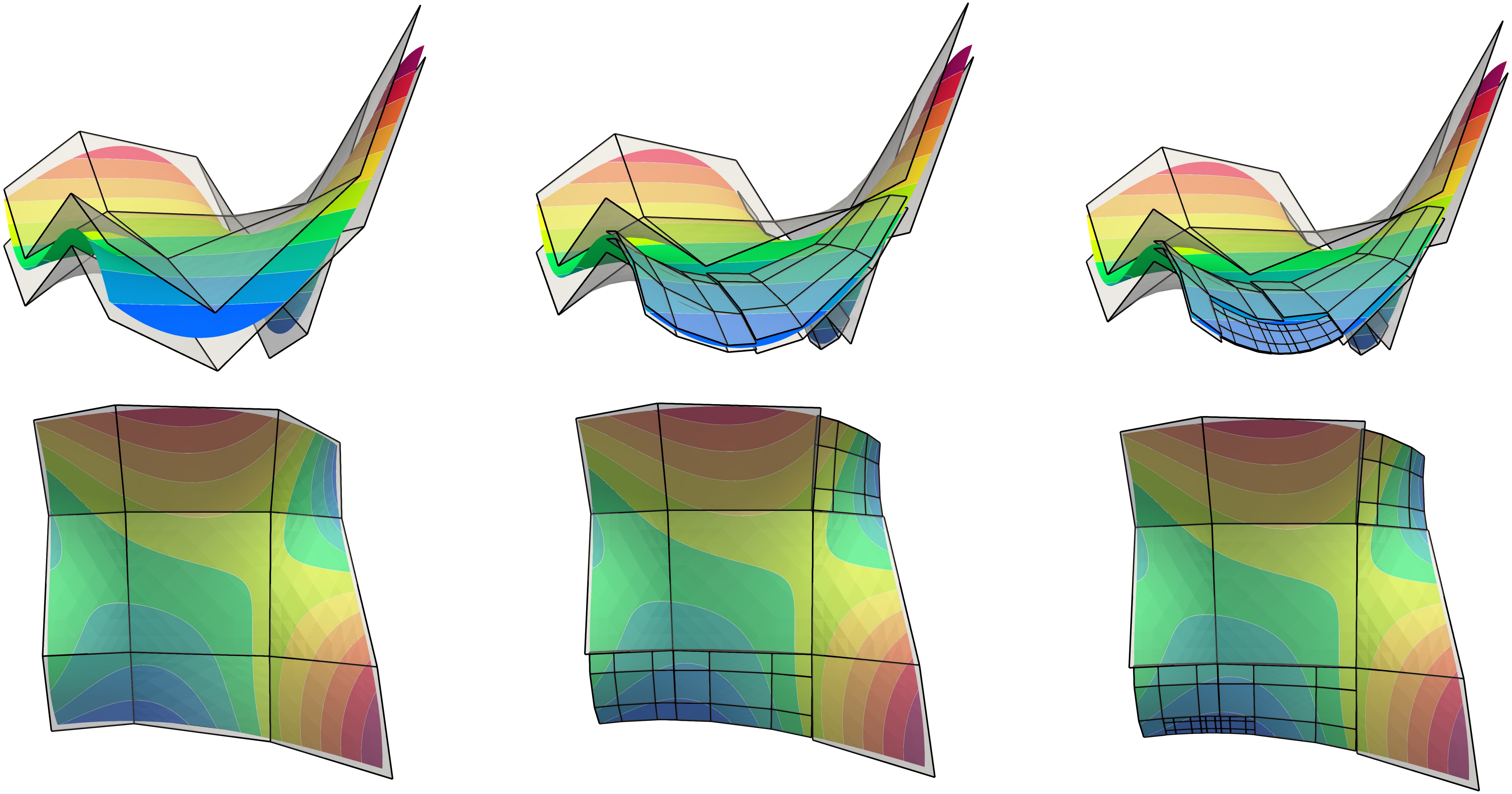
Determining Mesh Validity - 2D Example





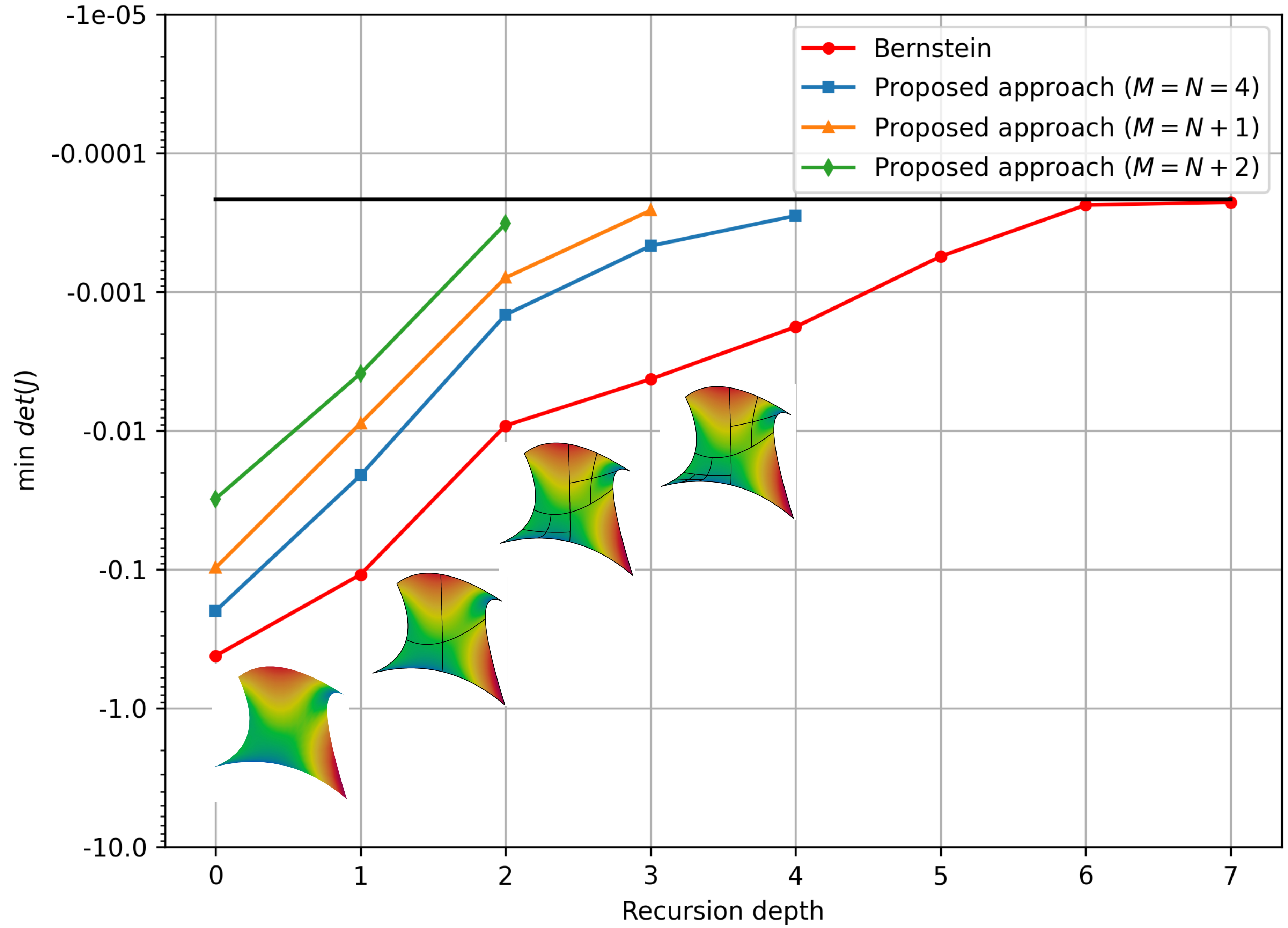
Piecewise linear bounds on the Jacobian determinant of a 2D quadrilateral element,
$p_{\mathtt{mesh}} = 2, p_{\mathtt{det}(J)} = 3.$

Determining Mesh Validity - 2D Example



Recursion based on the piecewise linear bounds on the Jacobian determinant to determine element validity $[N = 4, M = 4]$.

Determining Mesh Validity in 2D - Comparison with the Bernstein Bases

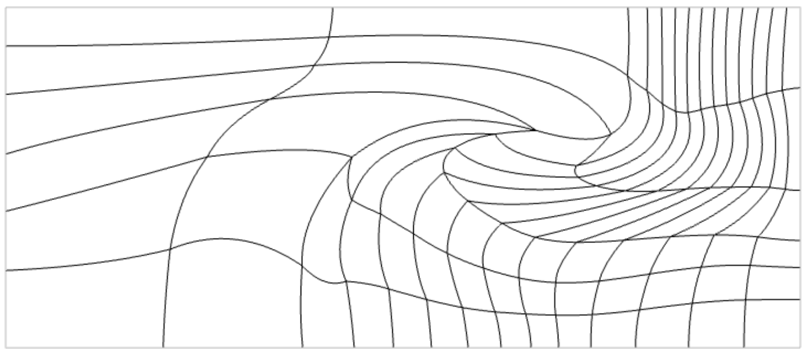


Interface in MFEM

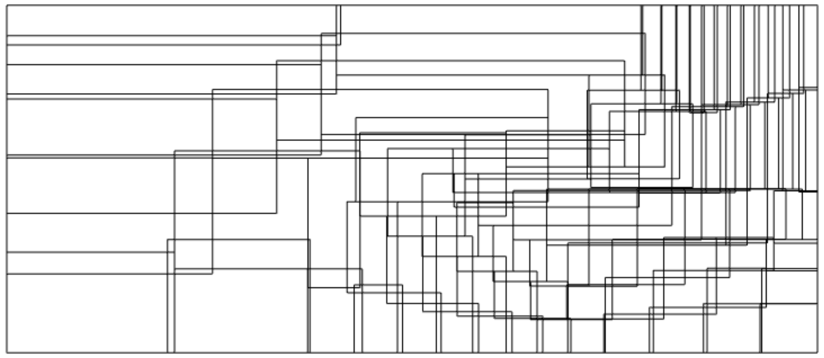
```
// Constructor
PLBound(const int nb_i, const int ncp_i, const int b_type_i,
        const int cp_type_i, const real_t tol_i)
int nb; // #mesh nodes in 1D
int ncp; // #control points in 1D
int b_type; // bases type: 0 - GL, 1 - GLL, 2 - Bernstein
int cp_type; // control points type: 0 - GL+Ends, 1 - Chebyshev
real_t tol = 0.0; // offset bounds to avoid round-off errors

/// Compute piecewise linear bounds for the lexicographically-ordered
/// coefficients in @a coeff in 1D/2D/3D.
PLBound::GetNDBounds(int rdim, Vector &coeff,
                     Vector &intmin, Vector &intmax) const
```

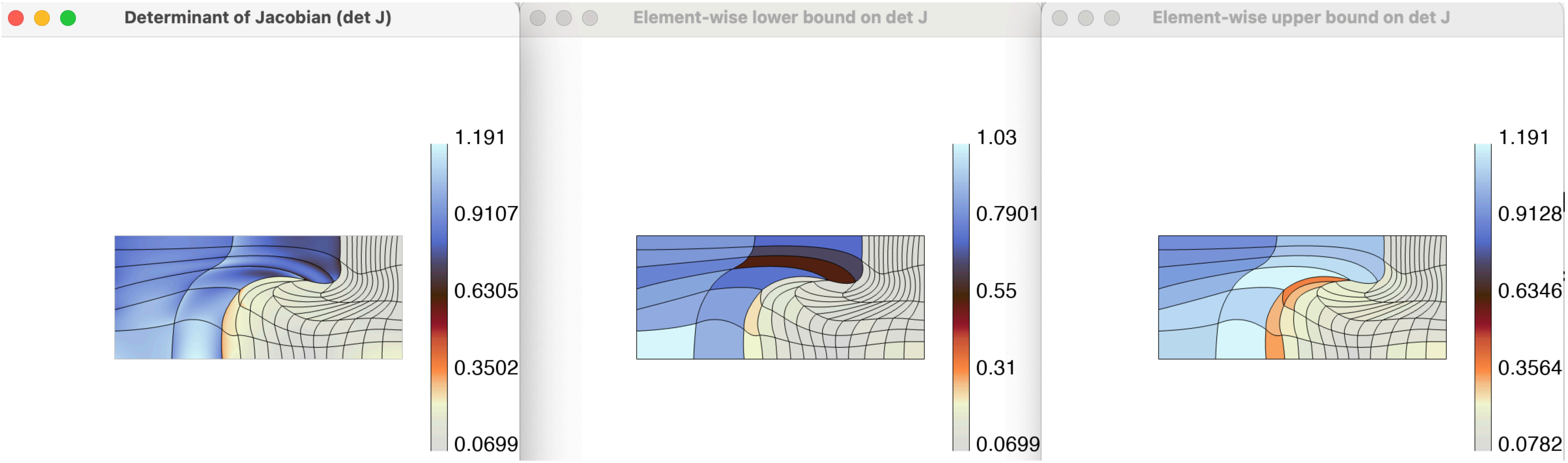
```
GridFunction::GetBounds(Vector &lower, Vector &upper,
                        const int ref_factor, const int vdim)

GridFunction::GetElementBounds(Vector &lower,
                               Vector &upper,
                               const int ref_factor,
                               const int vdim)
```

miniapps/meshing/mesh – bounding – boxes

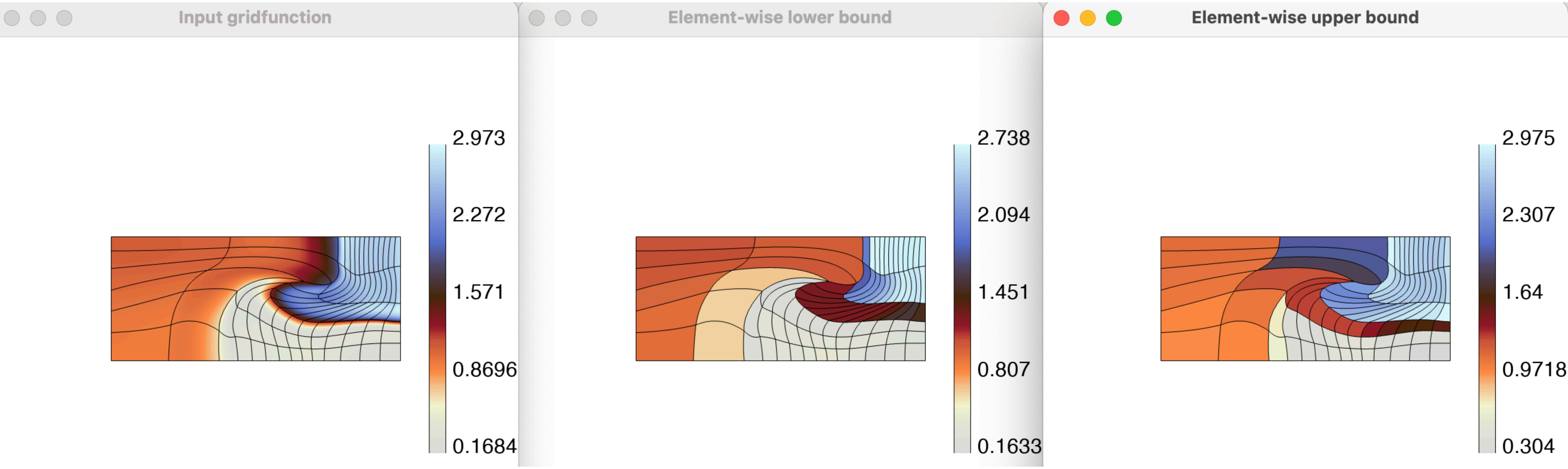


```
GridFunction *nodes = pmesh.GetNodes();
Vector lower, upper;
nodes->GetElementBounds(lower, upper);
```



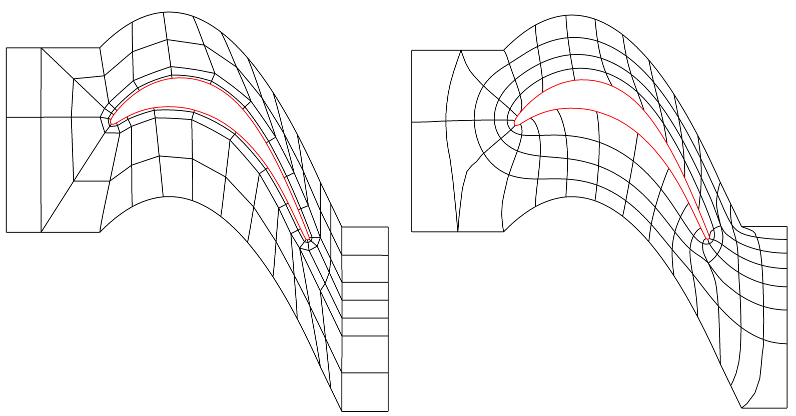


miniapps/tools/gridfunction – bounds

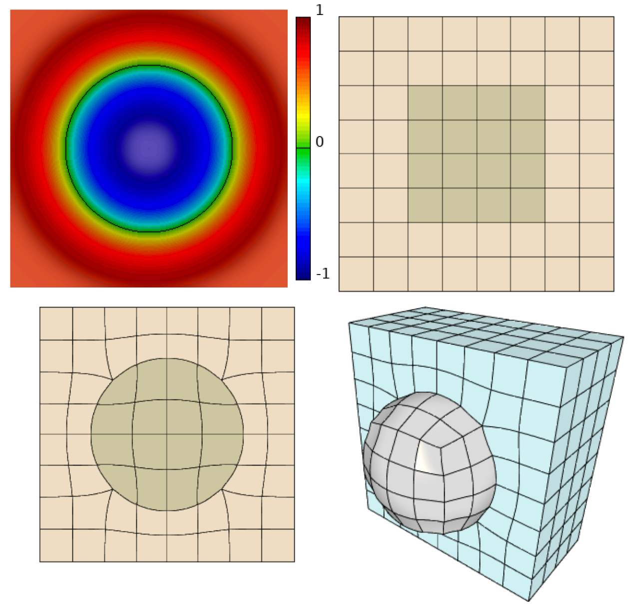


Recent Developments in High-Order Mesh Optimization

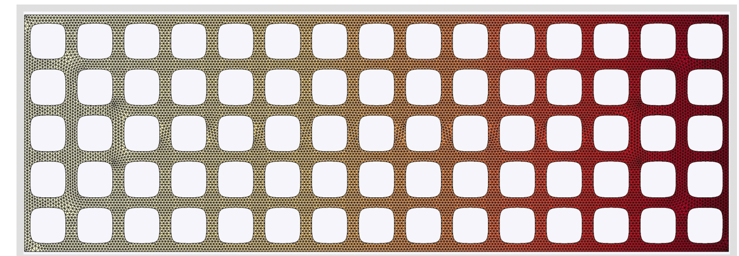
Mesh Quality Improvement with TMOP



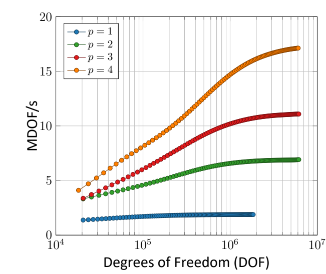
The target-matrix optimization paradigm for high-order meshes, SISC.



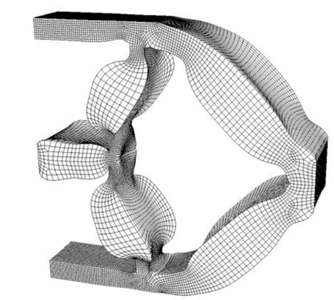
Adaptive surface fitting, IMR.



Mesh fitting for conformal shape optimization, CADJ.

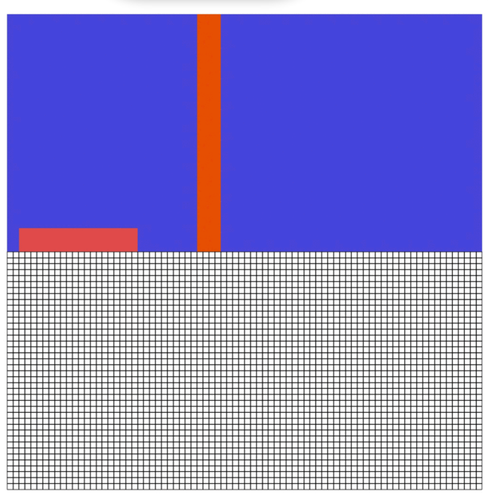


TMOP on GPUs, JCP.

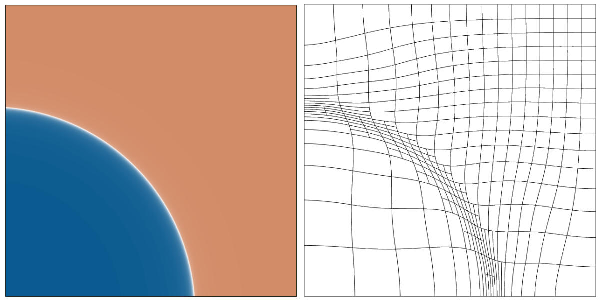


Mesh optimization with node limiting for shape optimization, SMO.

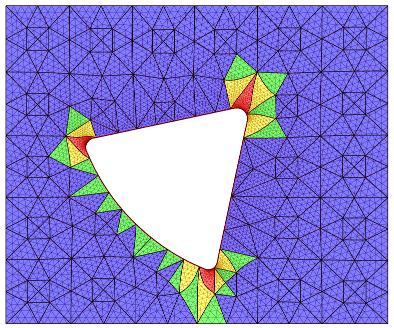
2019 → 2020 → 2021 → 2022 → 2023 → 2024



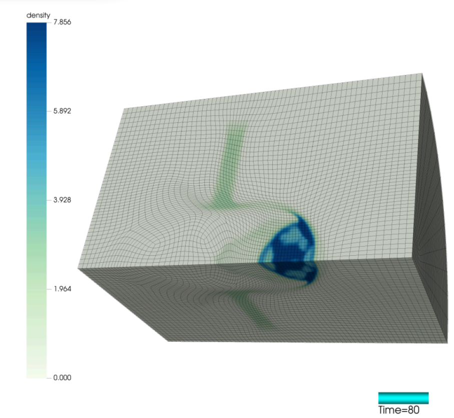
Simulation-driven optimization (mesh optimization + node limiting) SIAM IMR and Computers & Fluids.



Simulation-driven hr-adaptivity, Engineering with Computers.

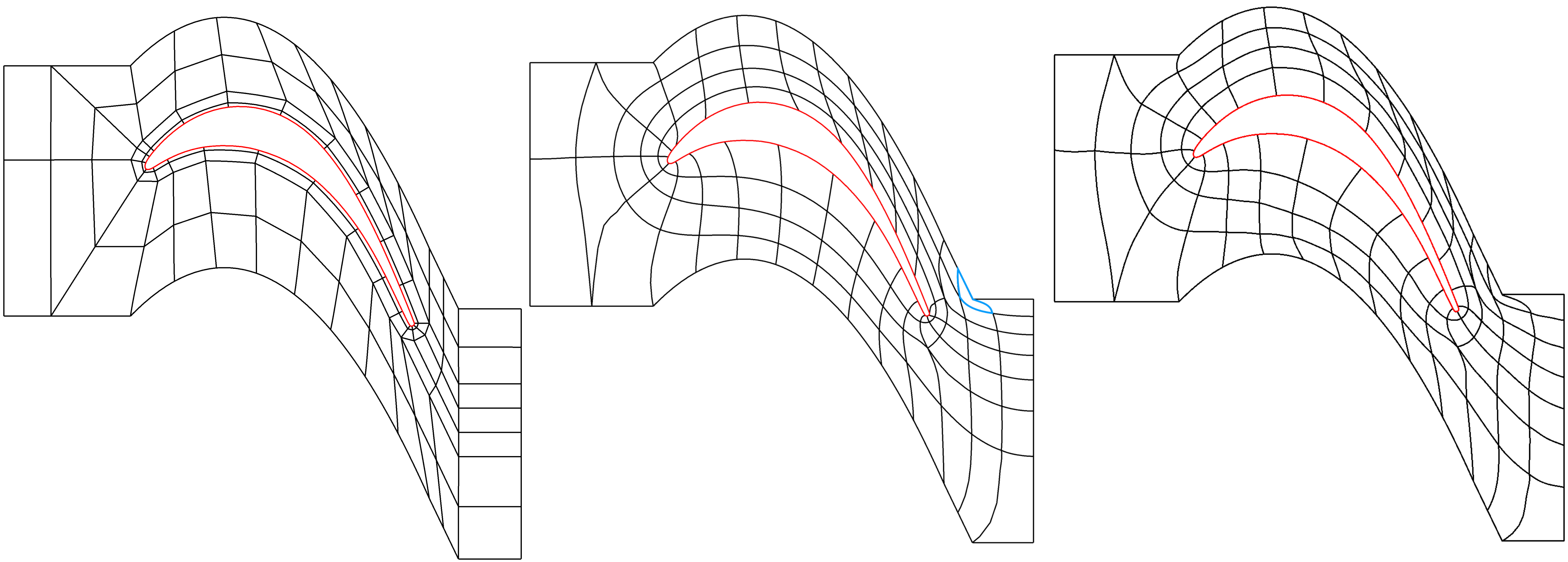


Mesh rp-adaptivity, IMR.



Asymptotic analysis of compound metrics, IMR.

Guaranteeing Mesh Validity



4th order mesh for a turbine blade.

$p_{\mathtt{mesh}} = 4, p_{\mathtt{det}(J)} = 7, N_{1D} = 8.$

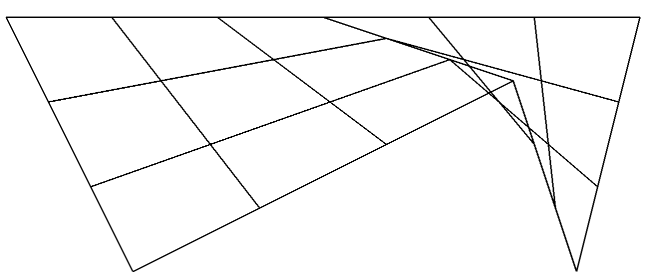
r-adaptivity ensures elements are valid at quadrature points but not necessarily continuously.

r-adaptivity with a guaranteed valid mesh via bounds on the determinant of the Jacobian.

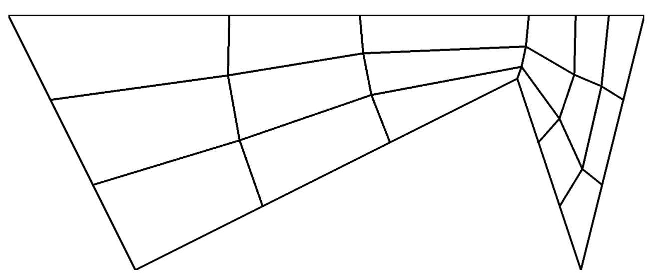
Mesh Untangling with a Shifted-Barrier Metric

$$\mu(T) = \frac{\tilde{\mu}(T)}{2\left(\tau - \tau_b\right)}$$

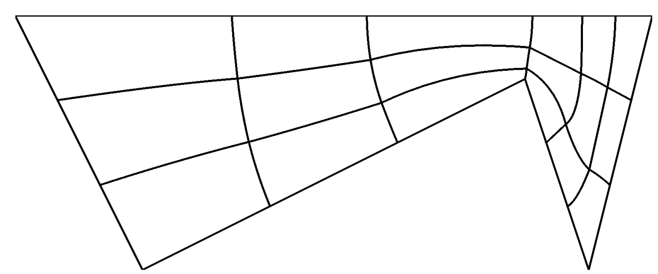
$$\tau_b = \begin{cases} \beta \cdot \tau_{\texttt{qp,min}} - \epsilon & \text{if } \tau_{\texttt{qp,min}} \le 0 \\ 0 & \text{otherwise} \end{cases}$$



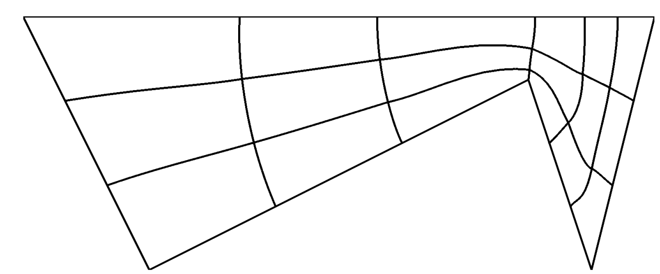
Tangled mesh



Optimized ($p = 1$)



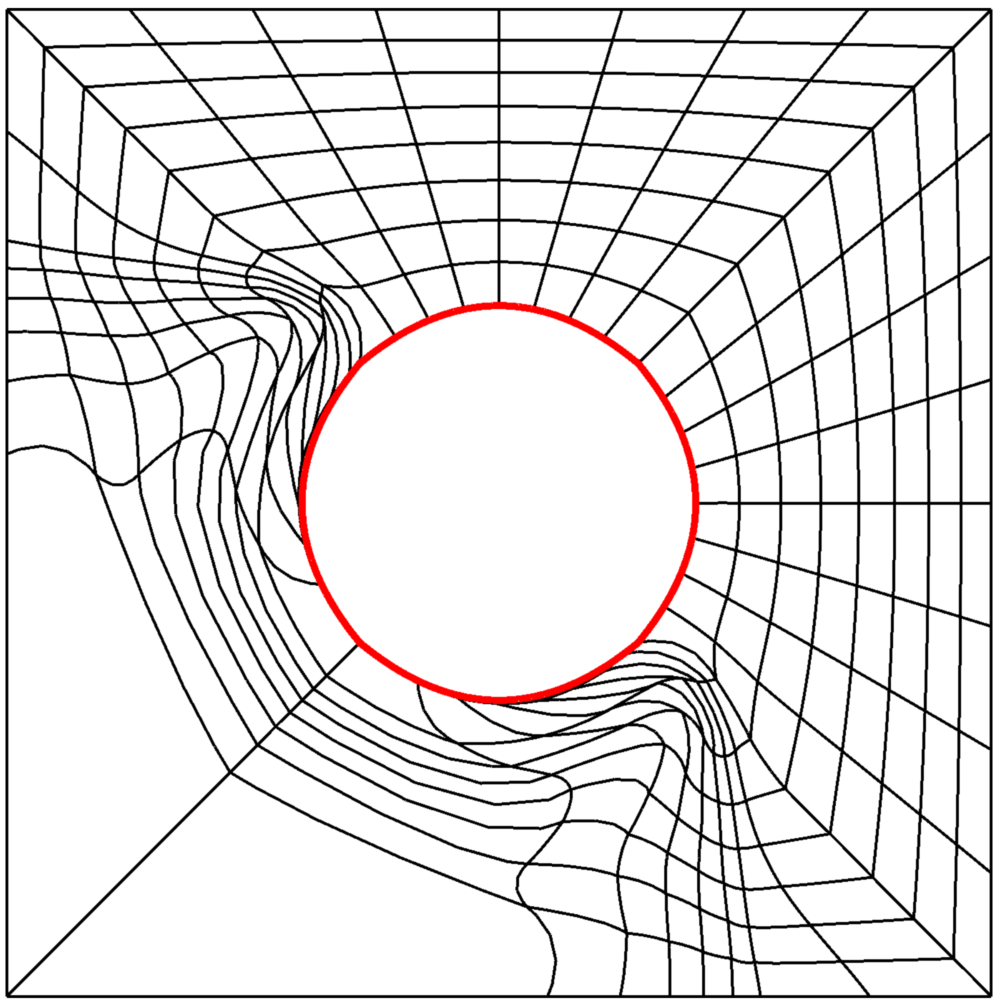
Optimized ($p = 2$)



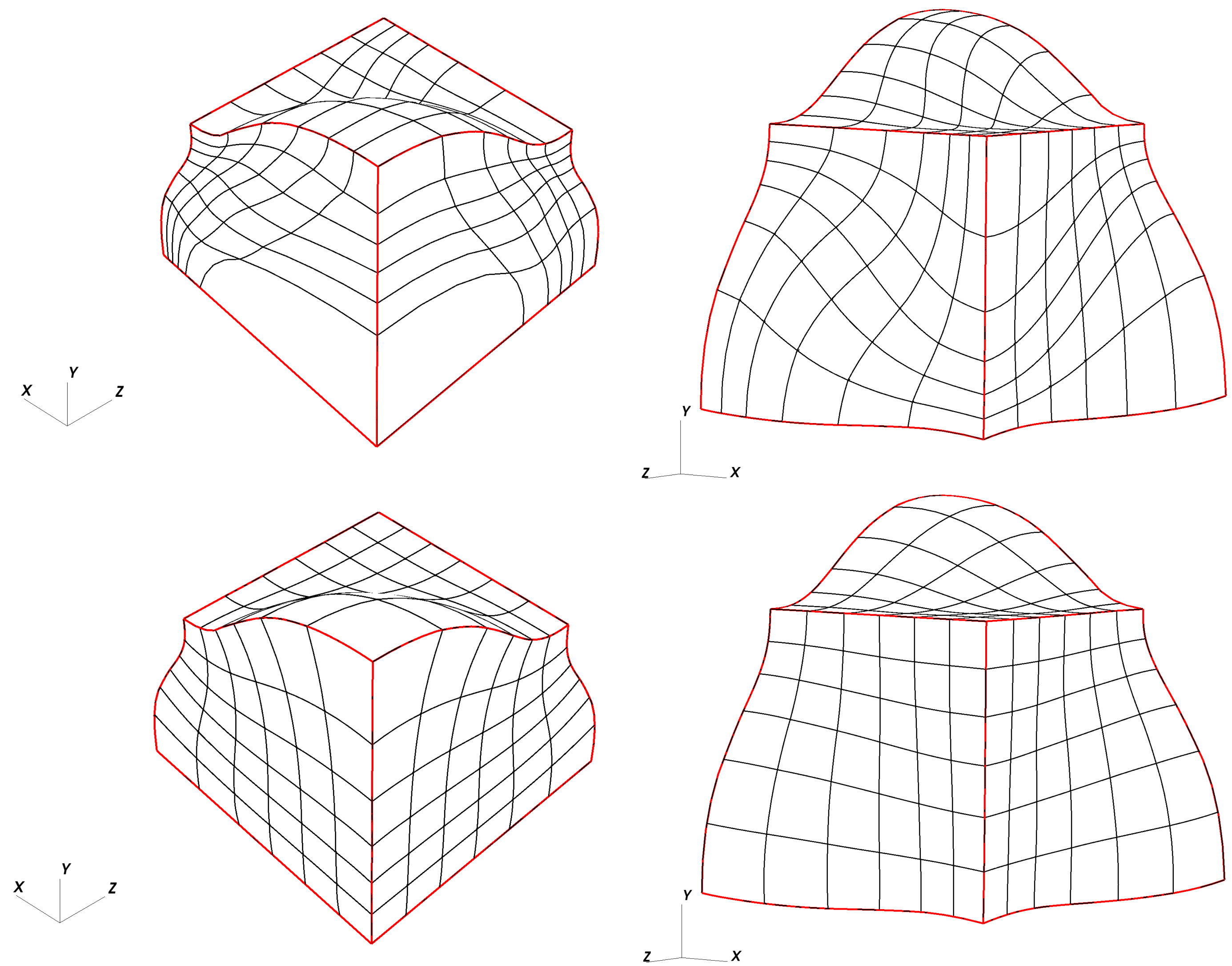
Optimized ($p = 3$)

Tangential Relaxation on Curved Boundaries

- Tangential relaxation enabled by closest point projection on surface meshes via a recent extension of FindPointsGSLIB.

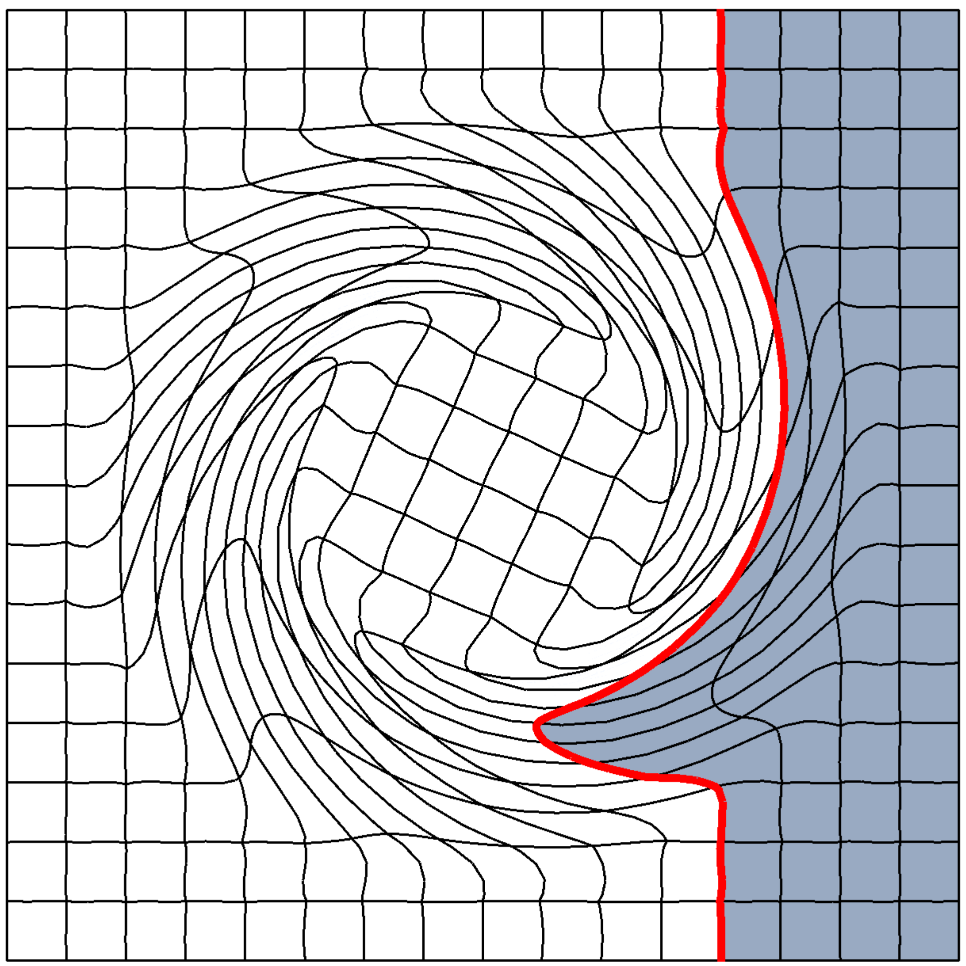


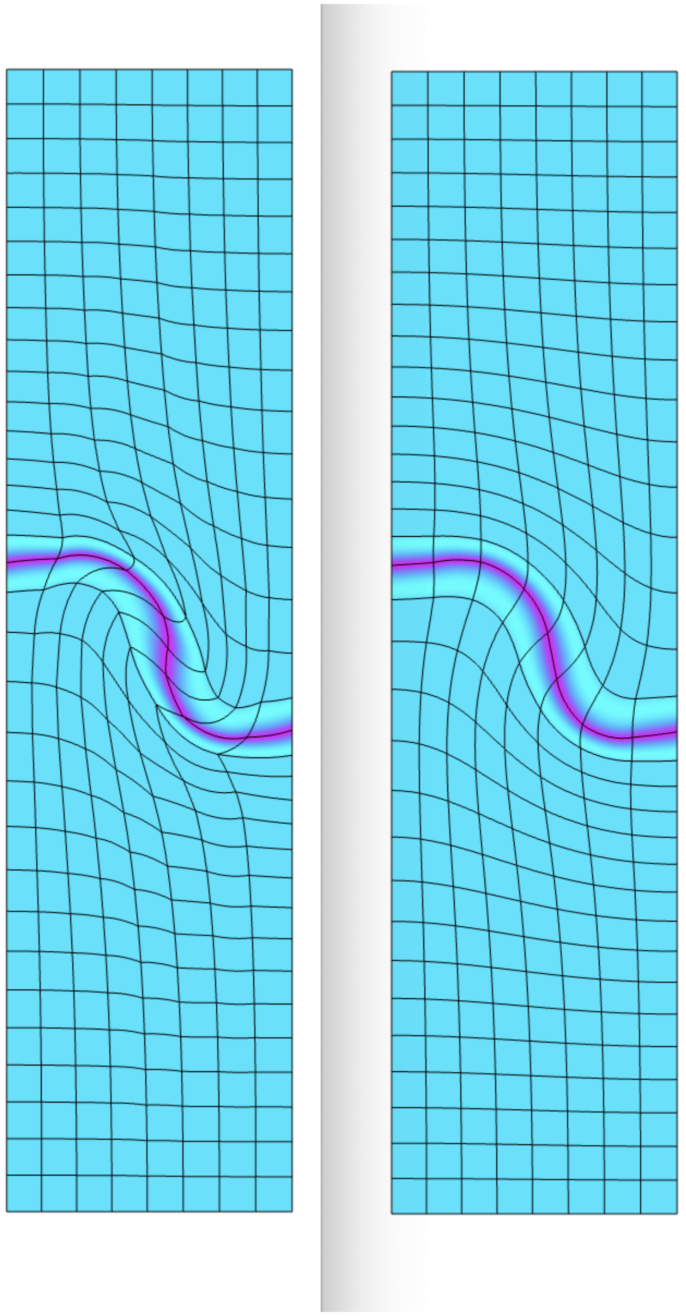
Quadratic mesh untangled and optimized with tangential relaxation



r-adaptivity with tangential relaxation for a 3D mesh.

Tangential Relaxation on Curved Interfaces





Tangential relaxation for volume fraction-based interface

PDE-Constrained Optimization

- Novel technique to improve mesh quality and PDE solution accuracy.

$$F(\boldsymbol{x}) = \underbrace{F_\mu}_{\text{mesh quality}} + \underbrace{\alpha}_{\text{weight}} \underbrace{F_p(u(\boldsymbol{x}), \boldsymbol{x})}_{\text{Error surrogate}} , \quad \text{s.t.} \quad \underbrace{\mathscr{R}_P(u)}_{\text{PDE residual}} = 0$$

  - F_μ based on TMOP for mesh quality
  - F_p is the error estimator, e.g., $F_p(u(\boldsymbol{x}), \boldsymbol{x}) = \sum_e \int_{\Omega^e} (u_e - \bar{u}_e)^2 \, d\Omega^e$

- Adjoint sensitivity analysis used to compute the implicit dependency of the objective:

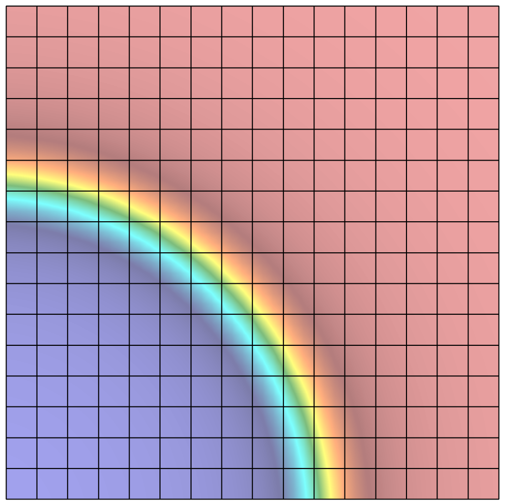
$$\frac{dF}{d\boldsymbol{x}} = \underbrace{\frac{\partial F}{\partial \boldsymbol{x}}}_{\text{explicit}} + \underbrace{\frac{\partial F}{\partial u} \frac{\partial u}{\partial \boldsymbol{x}}}_{\text{implicit}} = \frac{\partial F_\mu}{\partial \boldsymbol{x}} + \alpha \frac{\partial F_P}{\partial \boldsymbol{x}} + \alpha \frac{\partial F_P}{\partial u} \frac{\partial u}{\partial \boldsymbol{x}}$$

- Algebraic approach extends to any PDE with a well-defined Adjoint operator

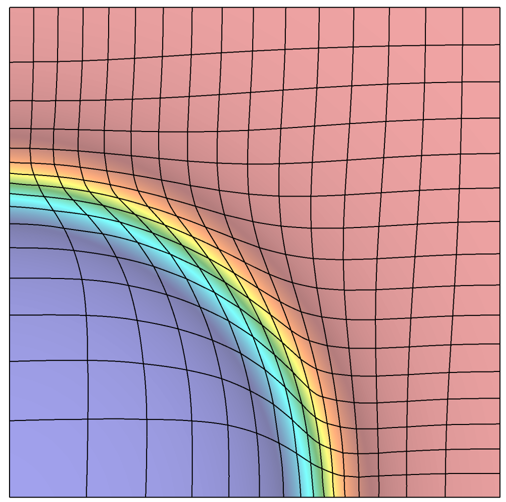
PDE-Constrained High-Order Mesh Optimization, arXiv: 2507.01917.

PDE-Constrained Optimization - Poisson

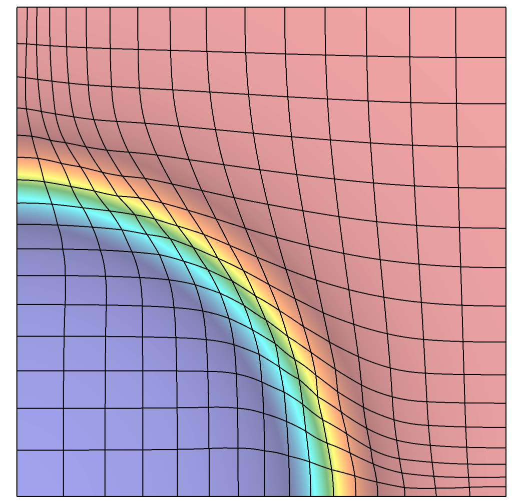
Original mesh



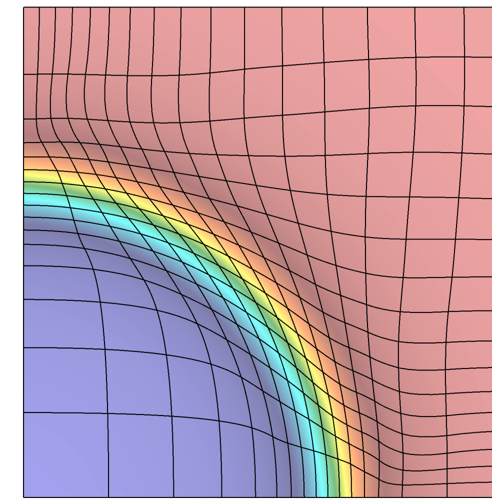
$$F_p(u(\boldsymbol{x}),\boldsymbol{x}) = \sum_e \int_{\Omega^e} (u_e - \bar{u}_e)^2 \; d\Omega^e$$

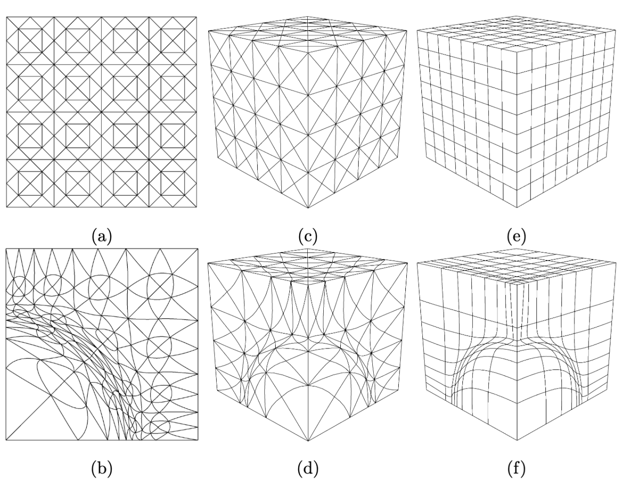


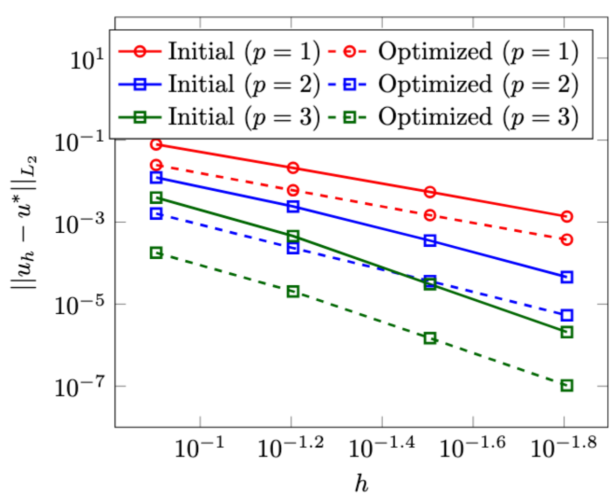
$$F_p(u(\boldsymbol{x}),\boldsymbol{x}) = \int_{\Omega} (\nabla u - \Pi \nabla u)^2 \; d\Omega$$

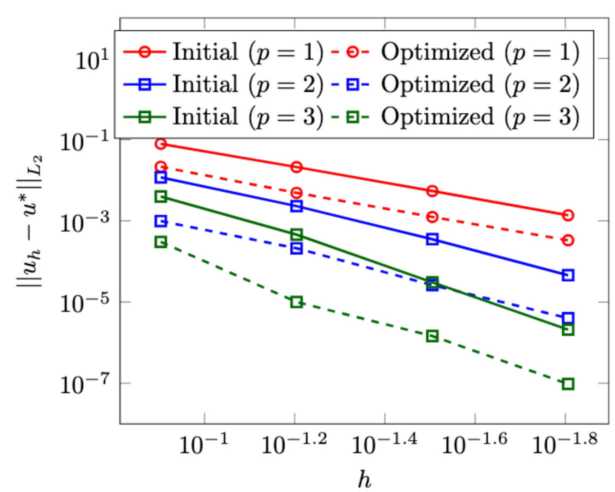


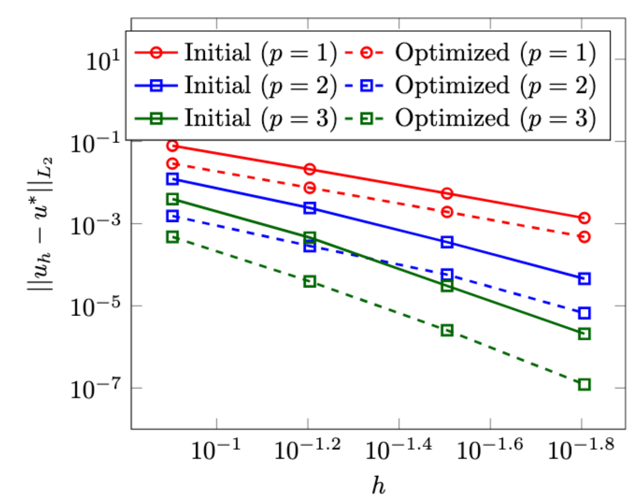
$$F_P(\boldsymbol{x}, u(\boldsymbol{x})) = -\int_{\Omega} u \cdot f \; d\Omega$$



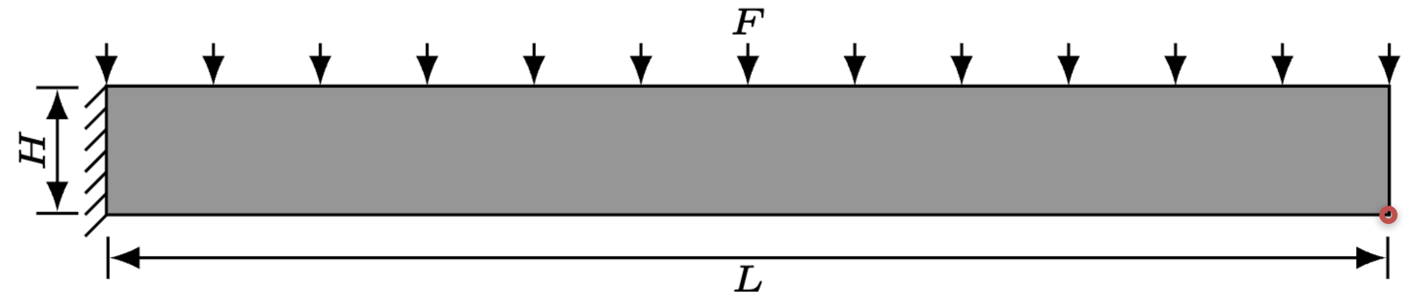


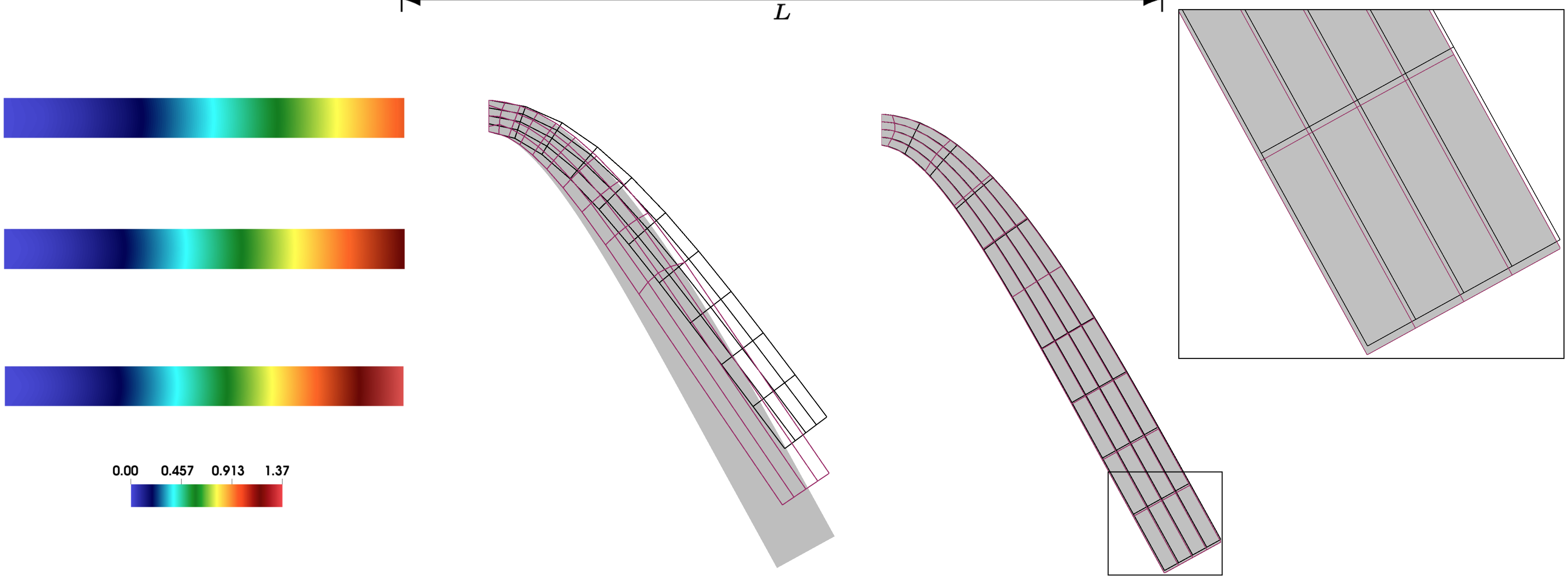






PDE-Constrained Optimization - Linear elasticity



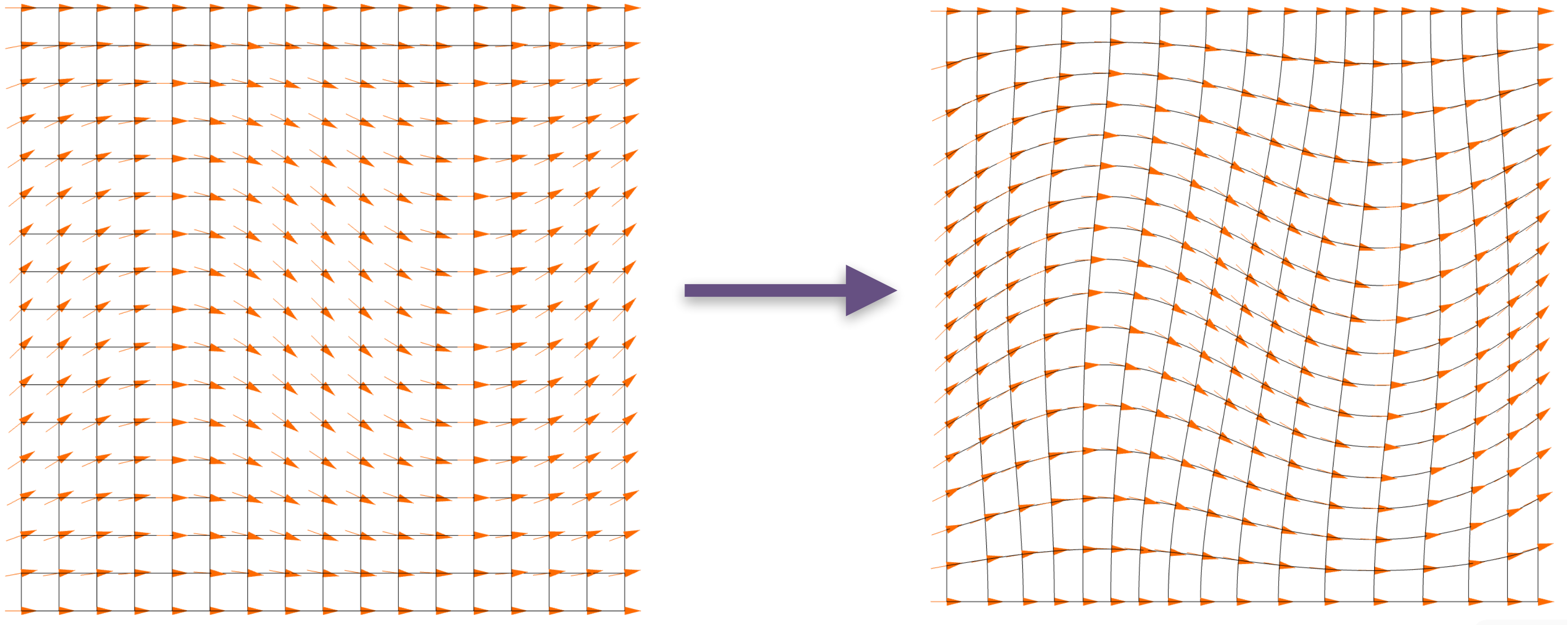


Mesh	$\vec{u}$	$\lvert\vec{u}\rvert$
Reference	(-0.0907,-1.3766)	1.3796
Initial (p=1)	(-0.0662,-0.9999)	1.0021
Optimized (p=1)	(-0.0736,-1.1776)	1.1799

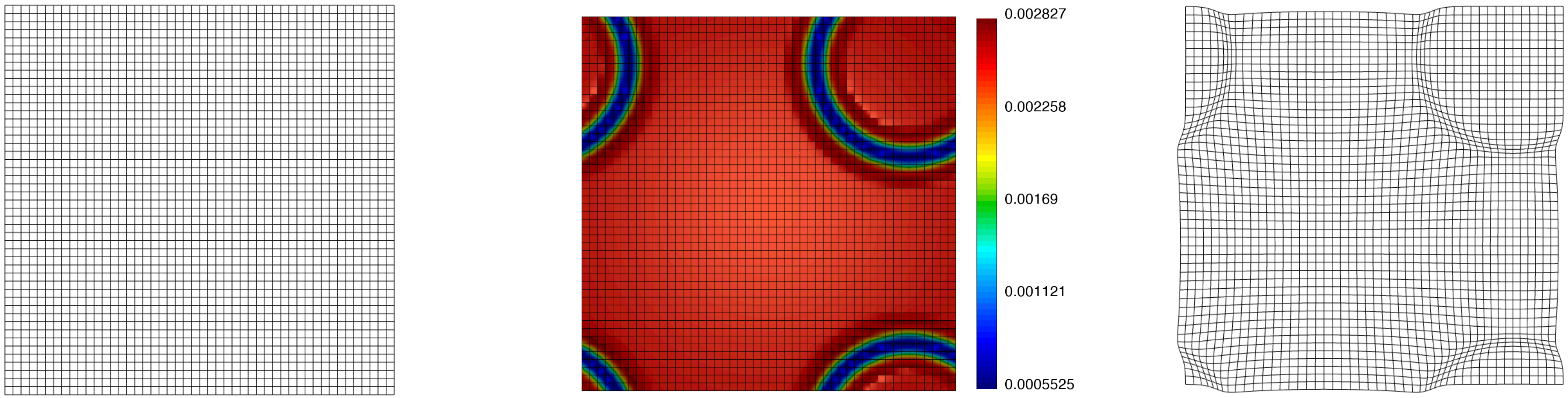
Mesh	$\vec{u}$	$\lvert\vec{u}\rvert$
Reference	(-0.0907,-1.3766)	1.3796
Initial (p=2)	(-0.0903,-1.3679)	1.3708
Optimized (p=2)	(-0.0907,-1.3759)	1.3789

Other Updates

- Automatic differentiation for mesh quality metrics.



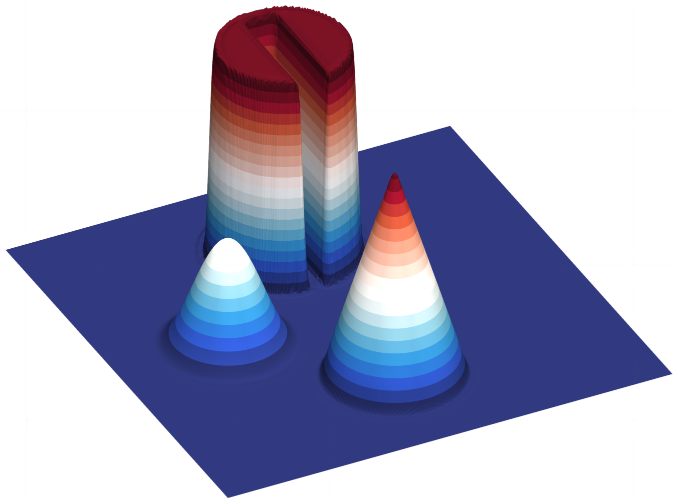
- r-adaptivity for periodic meshes.



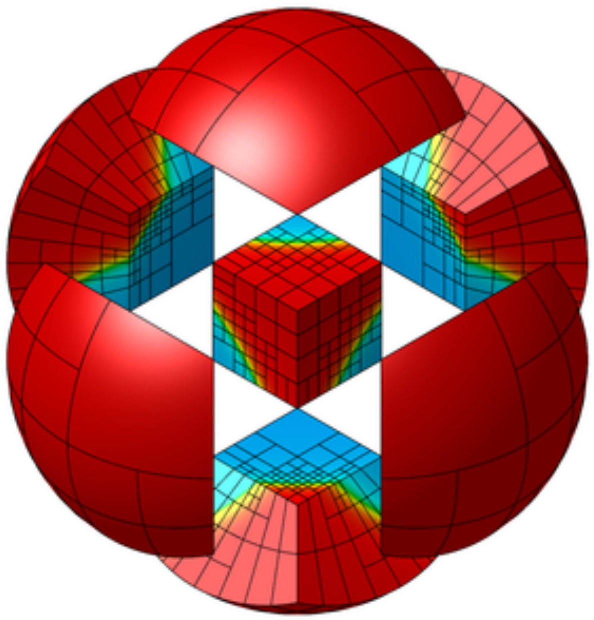
Uniform periodic mesh adapted to a sizing function.

Summary & Future Work

- Method for bounding high-order functions that supports different element types and bases.
  - Exploring ways to use it for remap
- High-order mesh r-adaptivity with guaranteed mesh validity, tangential relaxation for curved boundaries, and PDE-constrained optimization
  - Tangential relaxation for curved interfaces
  - Automatic differentiation for PDE-constrained optimization



Bounds preserving limiting for advection.



mfem.org

CASC

Center for Applied Scientific Computing

Lawrence Livermore National Laboratory