Accelerating High-Order Mesh Optimization Using Finite Element Partial Assembly

ICOSAHOM

14-18 August 2023

CASC
Center for Applied Scientific Computing

Ketan Mittal

J. S. Camier, V. Dobrev, P. Knupp, T. Kolev, V. Tomov

LLNL-PRES-852734

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

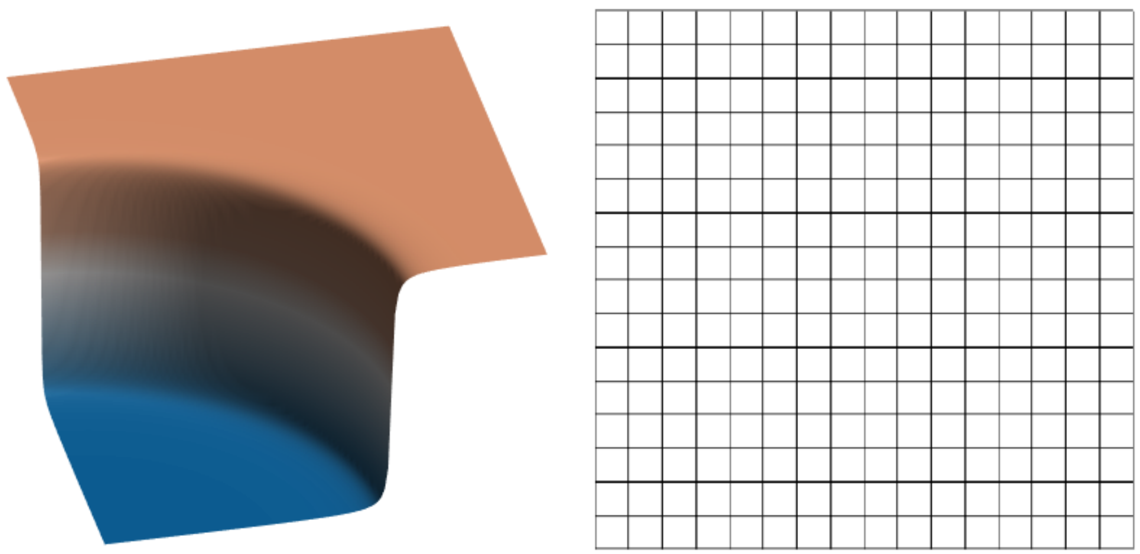
Lawrence Livermore National Laboratory

Overview

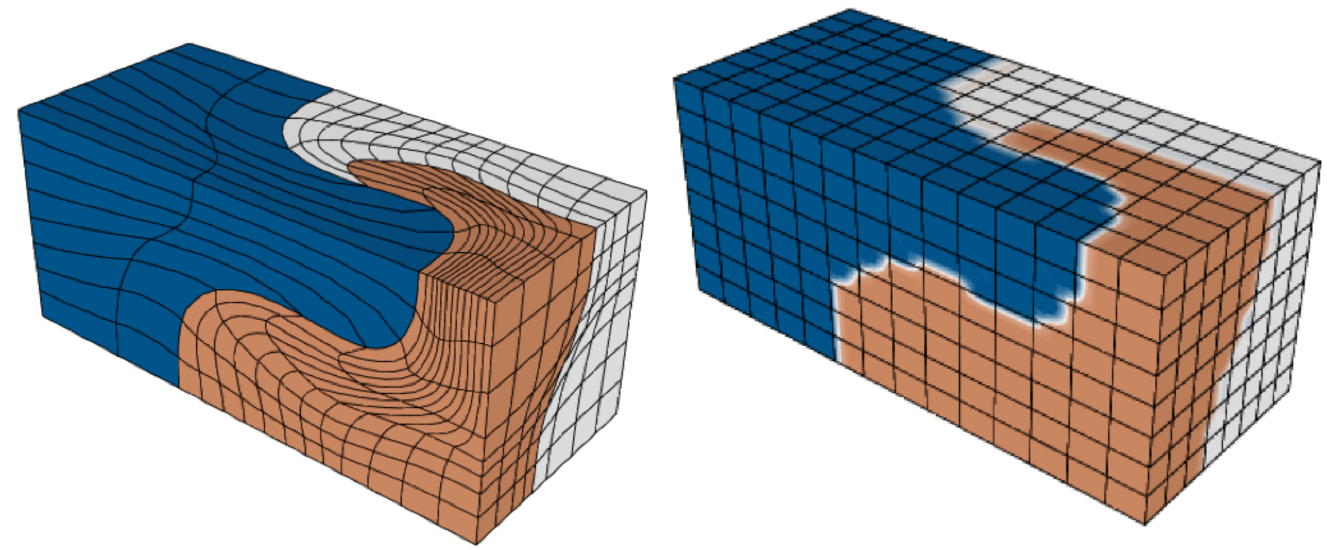
- High-order meshes
- Target Matrix Optimization Paradigm (TMOP) for r-adaptivity
- Acceleration through Partial Assembly
- Acceleration through Metric Linearization
- Future work

High-Order Mesh Optimization

- Why mesh optimization?



Outward propogating shock-wave



Multi-material Lagrangian Hydrodynamics

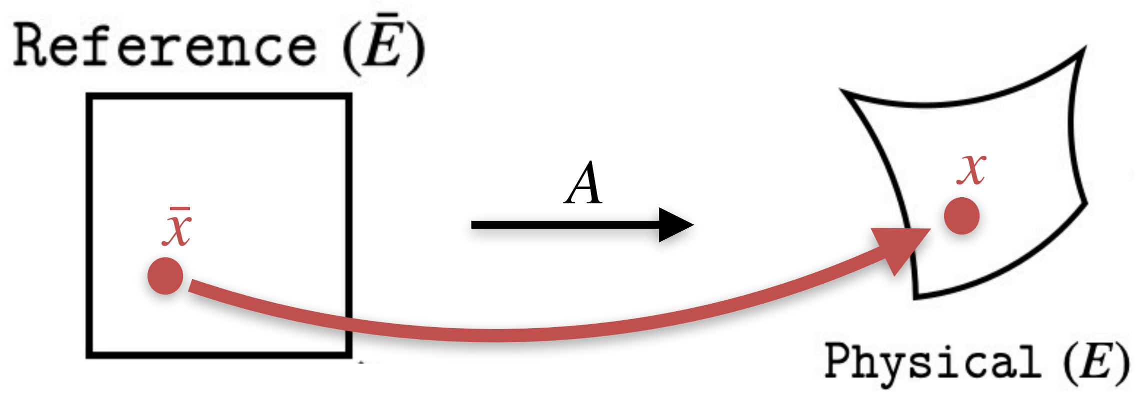
- Mesh optimization can help adapt the mesh to the solution and ultimately reduce error.

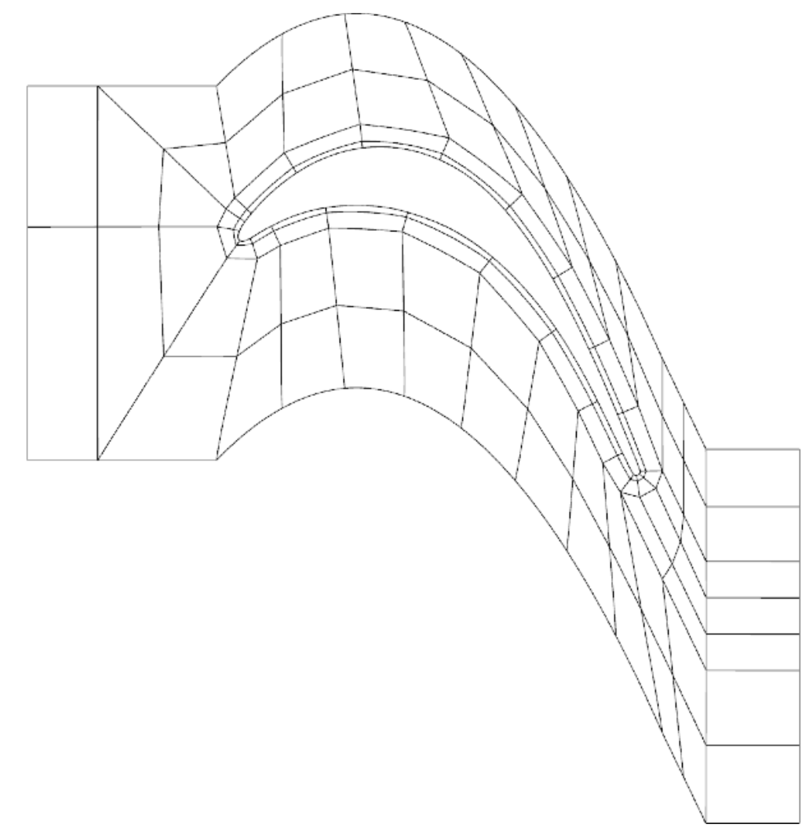
- Improve conditioning of the resulting system.

High-Order Mesh Representation

- Each element in the mesh is represented using scalar basis functions $\{\bar{w}_i\}_{i=1}^{N_p}$ on the reference element $\bar{E}$.

$$x(\bar{x}) = \Phi_E(\bar{x}) \equiv \sum_{i=1}^{N_p} \mathbf{x}_{E,i}\bar{w}_i(\bar{x}), \qquad \bar{x} \in \bar{E}, \quad x = x(\bar{x}) \in E$$



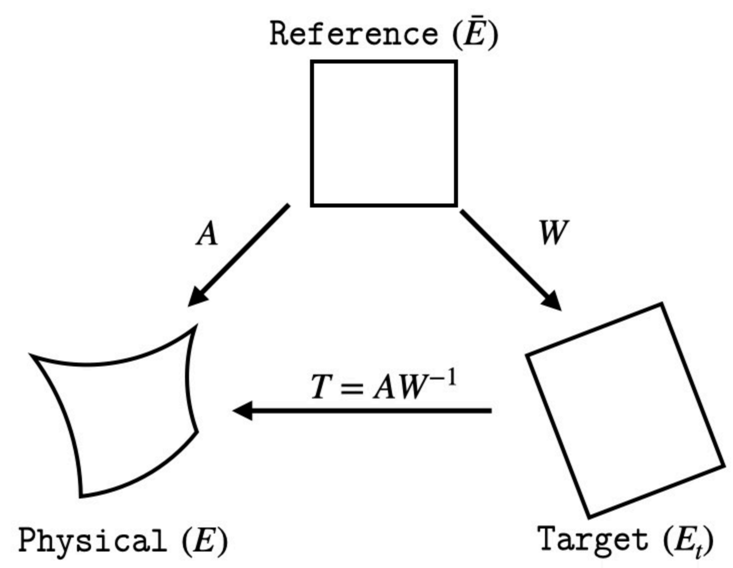


4th order mesh for a turbine blade

- The Jacobian of the transformation at each point represents the local deformation of the element with respect to the reference element:

$$A(\bar{x}) = \frac{\partial \Phi_E}{\partial \bar{x}} = \sum_{i=1}^{N_w} \mathbf{x}_{E,i}[\nabla \bar{w}_i(\bar{x})]^T$$

Target-Matrix Optimization Paradigm (TMOP)



- Any Jacobian transformation can be represented using four geometric parameters:

$$W = \underbrace{\zeta}_{\text{[volume]}} \quad \underbrace{R}_{\text{[rotation]}} \quad \underbrace{Q}_{\text{[skewness]}} \quad \underbrace{D}_{\text{[aspect-ratio]}}$$

- The transformation T from the active to target element can be defined using the Jacobian transformation A and W.

TMOP Mesh Quality Metrics

- Quality metric $\mu(T)$ is a measure of the deviation between the active and target Jacobian transformation.

- Different metrics depend on different geometric parameters.

  - $\texttt{Shape}$ metric - depends on Skew (*Q*) and Aspect-ratio (*D*). $\mu_2(T) = 0.5\frac{|T|^2}{\det(T)} - 1$

  - $\texttt{Size}$ metric - depends on ζ. $\mu_{77}(T) = 0.5\left(\det(T) - \frac{1}{\det(T)}\right)^2$

  - Other kinds include $\texttt{Alignment}$, $\texttt{Shape} + \texttt{Size}$, $\texttt{Shape} + \texttt{Alignment}$, etc.

  - We typically deploy $\texttt{Shape} + \texttt{Size}$ metrics but seldom also use $\texttt{Alignment}$ metrics.

- $\mu(T)$ typically non-linear and defined such that its minima is $T = I$ with $\mu(I) = 0$.

Node Movement with TMOP

- Using the quality metric and the Jacobian transformation T, the TMOP objective function is defined as:

$$F(\mathbf{x}) = \sum_{E \in \mathcal{M}} F_E(\mathbf{x}_E) = \sum_{E(\mathbf{x}_E)} \int_{E_t} \mu(T(\mathbf{x})) d\mathbf{x}_t$$

where $\mathbf{x}$ represents mesh coordinates. The element-by-element integral is computed as:

$$\sum_{E \in \mathcal{M}} \int_{E_t} \mu(T(\mathbf{x}_t)) d\mathbf{x}_t = \frac{1}{N_E} \sum_{E \in \mathcal{M}} \sum_{\mathbf{x}_q \in E_t} w_q \det(W(\bar{\mathbf{x}}_q))\, \mu(T(\mathbf{x}_q))$$

- In practice, we can use multiple metrics with different spatial weights.

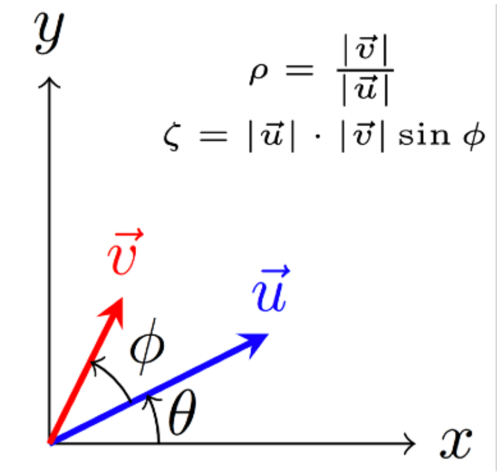
Node Movement with TMOP

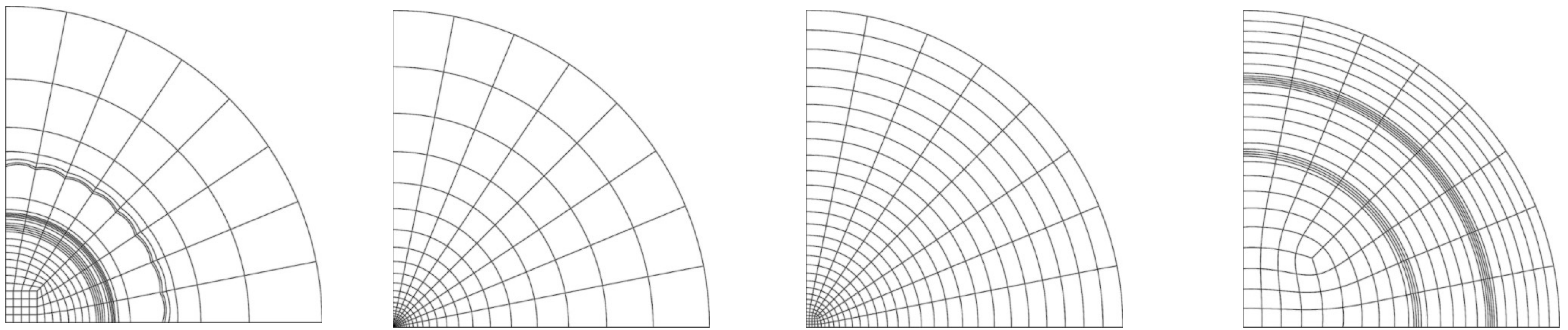
- r-adaptivity - $F(\mathbf{x})$ is minimized using a technique such as the Newton's method to optimize the mesh.
- Node movement direction: $\Delta\mathbf{x}_k = -\underbrace{[\partial^2 F(\mathbf{x}_k)]}_{\mathscr{H}(\mathbf{x}_k)}^{-1}\underbrace{\partial F(\mathbf{x}_k)}_{\mathscr{J}(\mathbf{x_k})}$
- Solution using MINRES with preconditioning: $\mathscr{H}(\mathbf{x}_k)\mathbf{\Delta}\mathbf{x}_k = \mathscr{J}(\mathbf{x}_k) \leftrightarrow Ay = b.$
- Newton update with line-search: $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha\Delta\mathbf{x}_k$. α is backtracked starting from 1.0 until:
  - $F(\mathbf{x}_{k+1}) \leq 1.2F(\mathbf{x}_k)$
  - $|\mathscr{J}(\mathbf{x}_{k+1})| \leq 1.2|\mathscr{J}(\mathbf{x}_k)|$
  - $\texttt{min}\Big(\det(A(\mathbf{x}_{k+1}))\Big) > \epsilon_{\texttt{det}}.$

Geometric r-adaptivity

- TMOP for *r-adaptivity*:

$$W = \sqrt{\frac{\zeta}{\sin\phi}} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & \cos\phi \\ 0 & \sin\phi \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{\rho}} & 0 \\ 0 & \sqrt{\rho} \end{bmatrix}$$





Original 2nd order mesh

Ideal shape + shape-metric.

Ideal shape, equal size + shape-metric.

Ideal shape, spatially varying size + shape+size -metric.

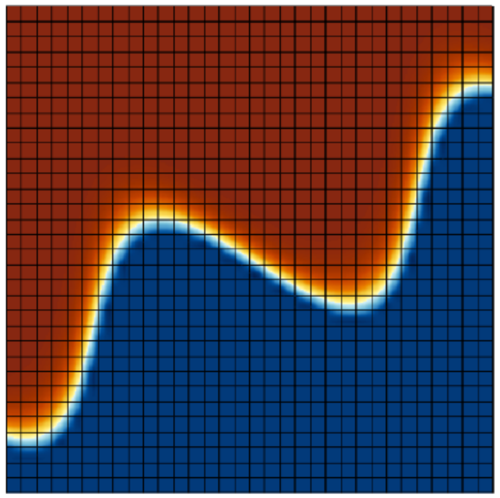
$\phi = \frac{\pi}{2}, \rho = 1, \mu_{Sh}(T)$

$\zeta = \frac{\mathscr{V}}{N_E}, \phi = \frac{\pi}{2}, \rho = 1, \mu_{ShSz}(T)$

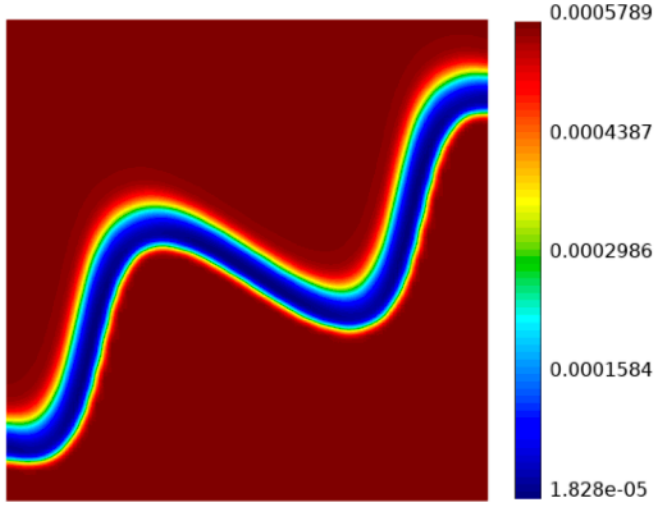
$\zeta(\mathbf{x}), \phi = \frac{\pi}{2}, \rho = 1, \mu_{ShSz}(T)$

"The target-matrix optimization paradigm for high-order meshes." SIAM Journal on Scientific Computing (2019).

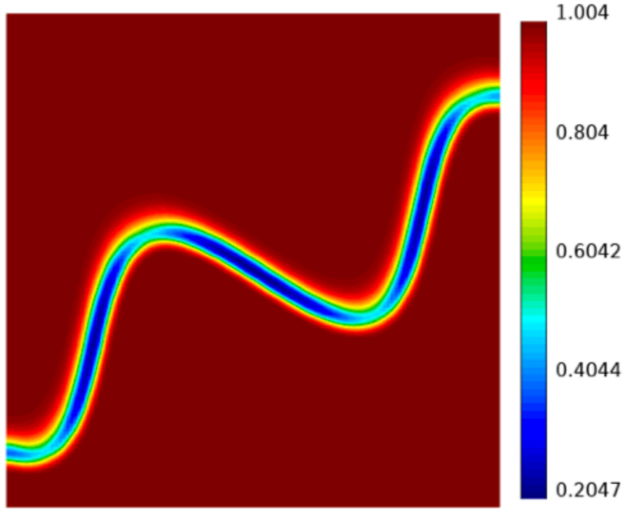
Simulation-driven Adaptivity



Simulation data material indicator (η)



Size - $\zeta \propto 1/|\nabla \eta|$



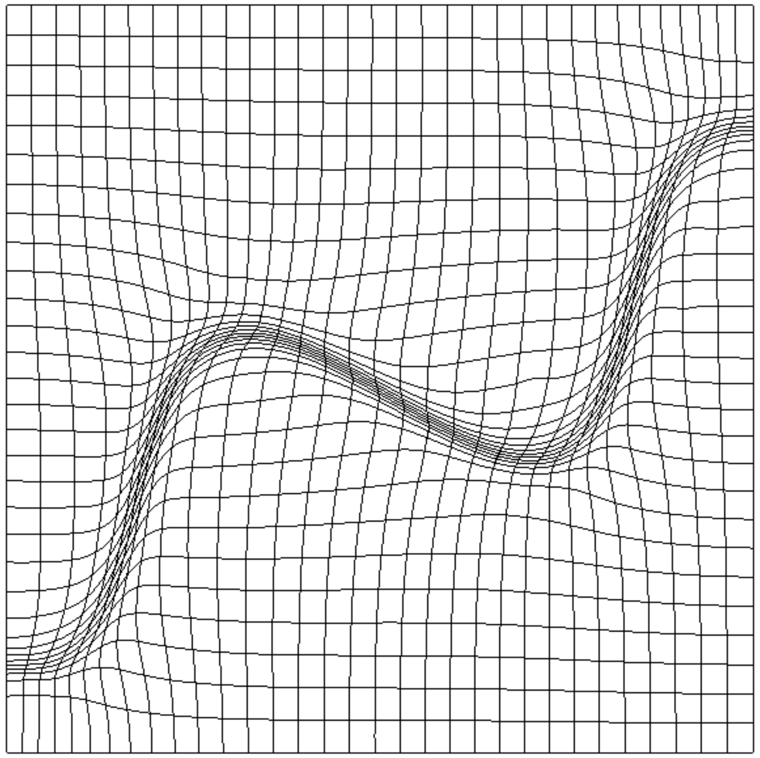
Aspect-Ratio - $\rho \propto |\eta_x/\eta_y|$

$$W = \sqrt{\frac{\zeta}{\sin\phi}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \cos\phi \\ 0 & \sin\phi \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{\rho}} & 0 \\ 0 & \sqrt{\rho} \end{bmatrix},$$

- $\phi = \frac{\pi}{2}$ for an ideal square.
- Use a Shape + Size polyconvex metric, $\mu_{80} = (1-\gamma)\mu_2 + \gamma\mu_{77}$.

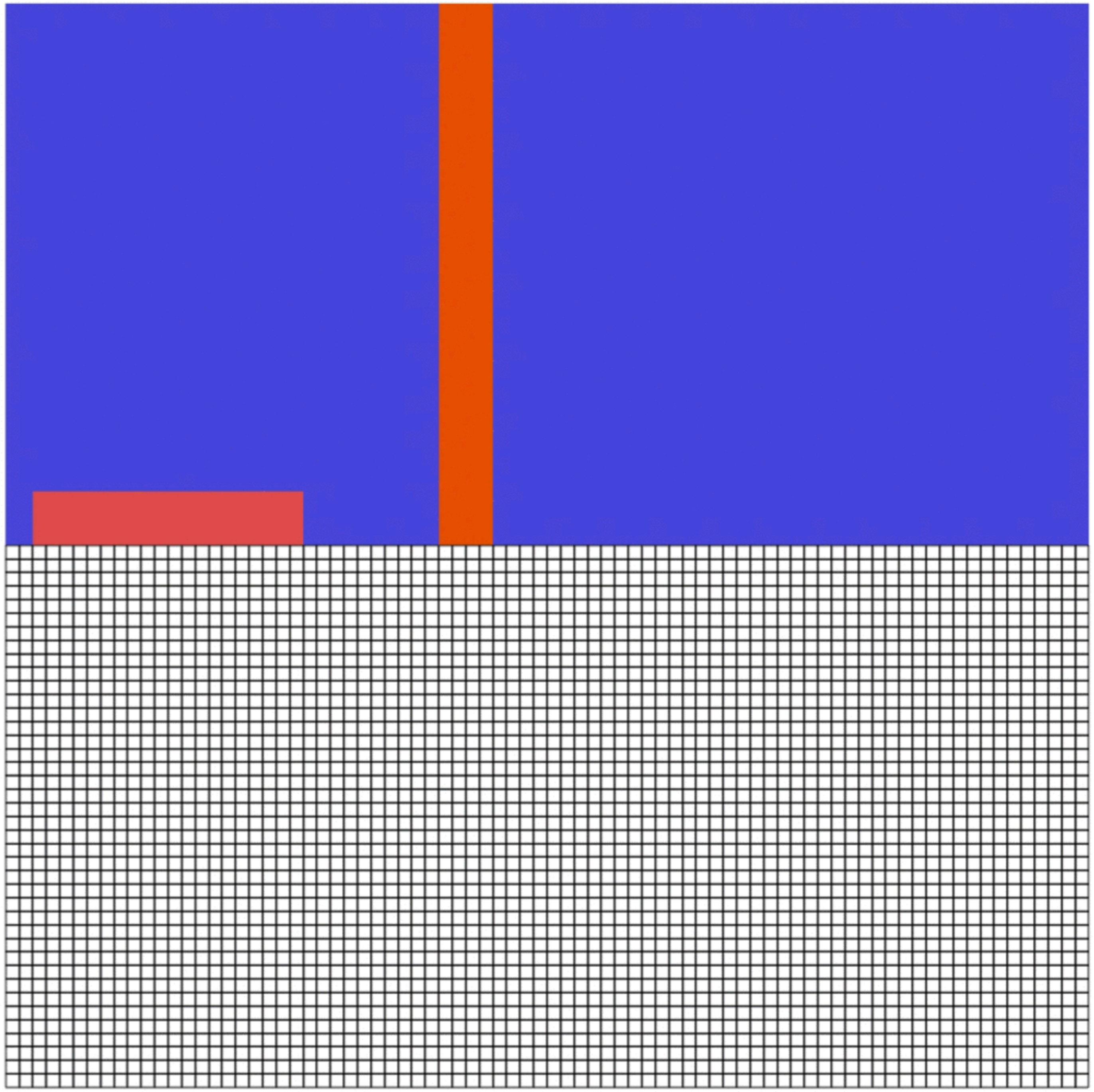
$$\mu_2(T) = 0.5\frac{|T|^2}{\det(T)} - 1 \qquad \mu_{77}(T) = \frac{1}{2}(\tau - \frac{1}{\tau})^2$$

- Note: η must be remapped between and after Newton iterations.



"Simulation-driven optimization of high-order meshes in ALE hydrodynamics." Computers & Fluids (2020).

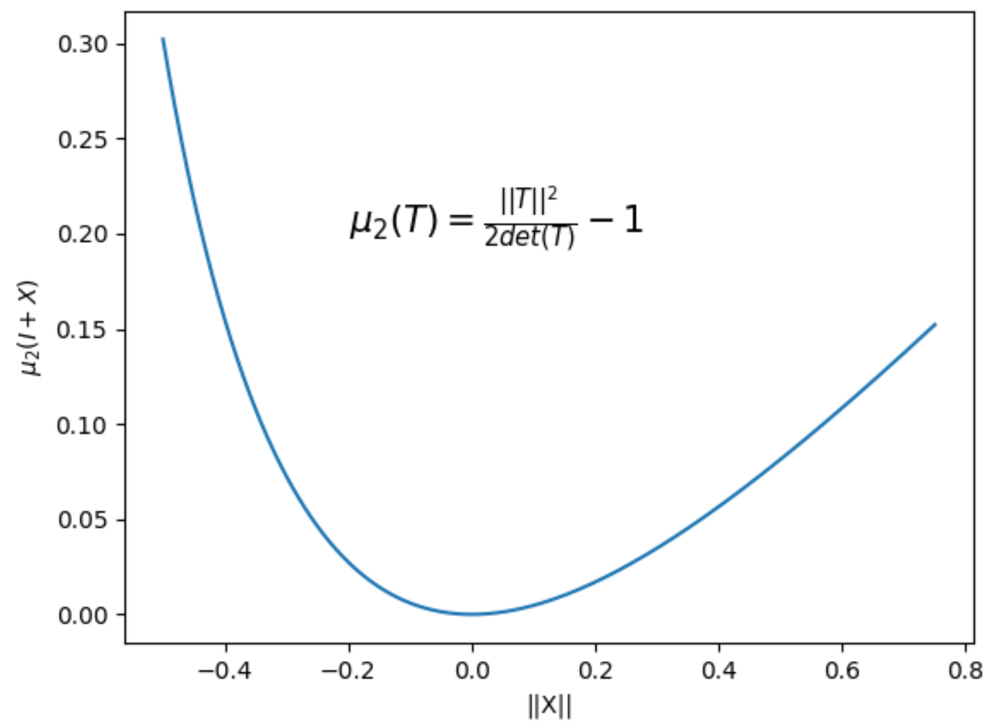
Simulation-driven Adaptivity



Simulation-driven adaptivity with TMOP for multi-material ALE.

Computational Cost of High-Order Mesh Optimization

- Nonlinear mesh quality metric → problem is not quadratic so need more than 1 Newton iteration.
  - Each Newton iteration requires assembly of matrix $\mathscr{H}$.
  - Each Newton iteration has O(10-100) MINRES iterations where we do matrix-vector products using $\mathscr{H}$.



Computational Cost of High-Order Mesh Optimization

- Minimizing the TMOP objective function entails solving $\mathscr{H}(\mathbf{x}_k)\mathbf{\Delta x}_k = \mathscr{J}(\mathbf{x}_k)$.

$$F(\mathbf{x}) = \sum_{E\in\mathscr{M}} F_E(\mathbf{x}_E) = \sum_{E(\mathbf{x}_E)} \int_{\bar{E}} \mu(T(\mathbf{x}))d\bar{\mathbf{x}}$$

- Assume $W = I$:

$$\frac{\partial F(\mathbf{x})}{\partial x_{a,i}} = \int_{\bar{E}} \frac{\partial \mu}{\partial T(\mathbf{x})} : \frac{\partial T(\mathbf{x})}{\partial x_{a,i}} d\bar{x} = \int_{\bar{E}} \frac{\partial \mu}{\partial T} : \left(\frac{\partial A}{\partial x_{a,i}}\right) d\bar{x}$$

$$a = 1,\ldots,d, \quad i = 1,\ldots,N_x$$

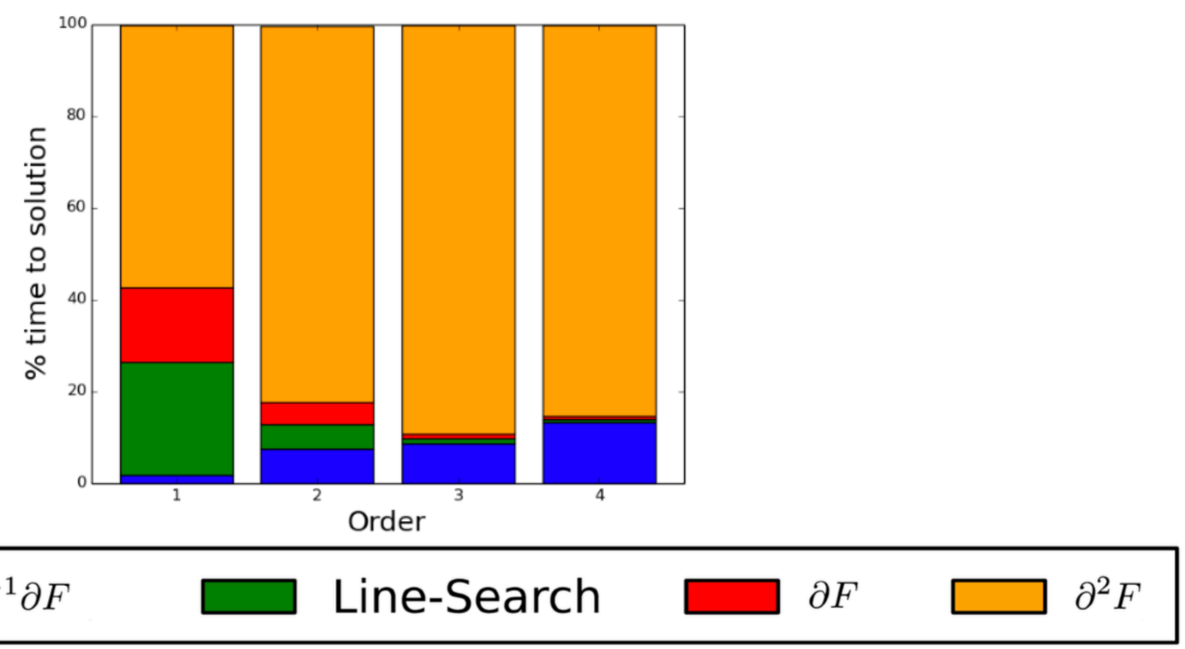
$$\frac{\partial^2 F(\mathbf{x})}{\partial x_{b,j}\partial x_{a,i}} = \int_{\bar{E}} \frac{\partial}{\partial x_{b,j}}\left[\frac{\partial \mu}{\partial T} : \left(\frac{\partial A}{\partial x_{a,i}}\right)\right] d\bar{x}$$

$$a, b = 1,\ldots,d, \quad i, j = 1,\ldots,N_x$$

Computational Cost of High-Order Mesh Optimization

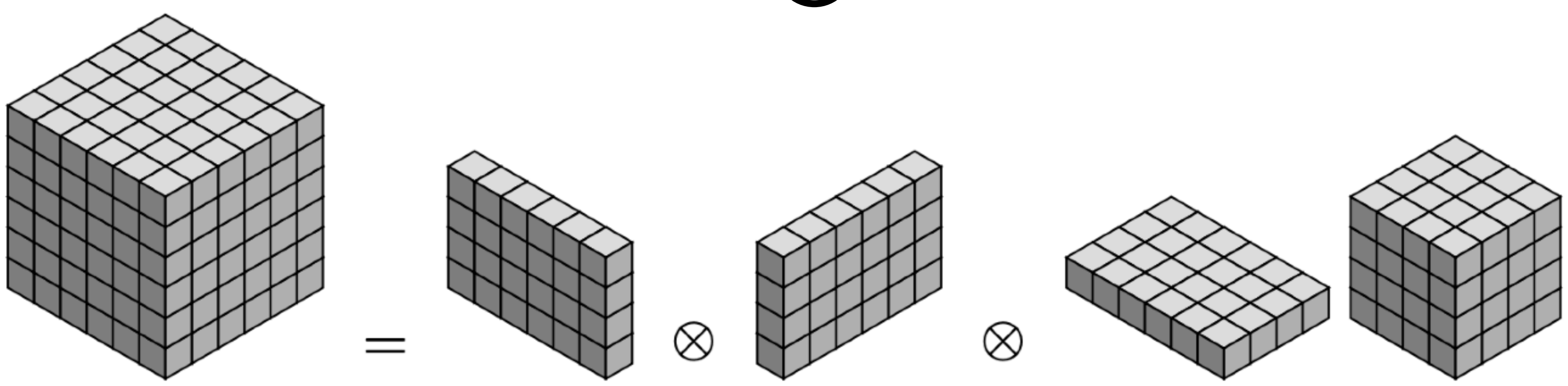
- Simple implementation (Full assembly):
  - Construct and store the global matrix $\mathscr{H}$ for the entire mesh at each Newton iteration.
  - Easy to setup but computationally expensive (prohibitive as p increases):

Method	Storage	Assembly	Evaluation
Traditional full assembly + matvec	$\mathcal{O}(p^{2d})$	$\mathcal{O}(p^{3d})$	$\mathcal{O}(p^{2d})$



Partial Assembly

- Acceleration for tensor-product elements (quads/hexes) using partial-assembly and matrix-free evaluation.
  - Construct nD Operators as a Kronecker product ($\bigotimes$) of 1D operators.



$$B_{3D} = \left(B_{1D} \otimes B_{1D} \otimes B_{1D}\right) u$$

  - Store only locally assembled 1D matrices.
  - At each iteration, store only quadrature point data and use locally stored 1D operatores to perform element-by-element matrix-vector products.

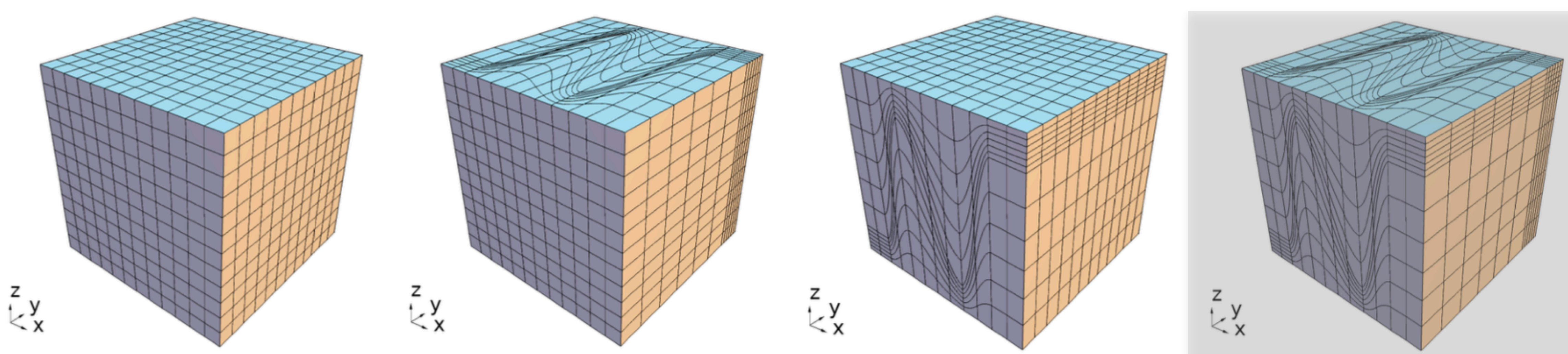
Method	Storage	Assembly	Evaluation
Traditional full assembly + matvec	$\mathcal{O}(p^{2d})$	$\mathcal{O}(p^{3d})$	$\mathcal{O}(p^{2d})$
Partial assembly + matrix-free action	$\mathcal{O}(p^{d})$	$\mathcal{O}(p^{d})$	$\mathcal{O}(p^{d+1})$

- Well suited for GPUs due to low storage complexity and matrix-vector products.

Kershaw Benchmark

- Easy-to-setup benchmark for timing high-order mesh optimization.
- Two parameters $(\epsilon_y, \epsilon_z) \in (0,1]^2$ control the element deformation.
- In our tests, we use a $24 \times 24 \times 24$ mesh, 9 quadrature points in each direction in an element, and a shape metric $\mu_{303} = \frac{|T|^2}{3\tau^{2/3}} - 1$.

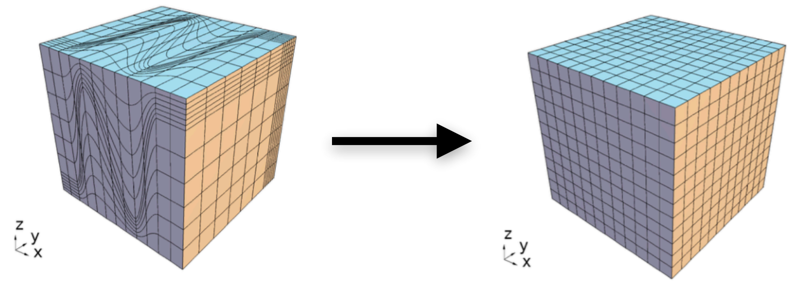
```
double right(const double eps, const double x) // 1D transformation at right boundary
{
   return (x <= 0.5) ? (2-eps)*x : 1+eps*(x-1);
}
double left(const double eps, const double x)  // 1D transformation at left boundary
{
   return 1-right(eps,1-x);
}
double step(const double a, const double b, double x)
{
   if (x <= 0) { return a; }
   if (x >= 1) { return b; }
   return a + (b-a)*(x*x*x*(x*(6*x-15)+10));   // Smooth transition from a to b
}
void kershaw(const double epsy, const double epsz,
             const double x, const double y, const double z,
             double &X, double &Y, double &Z)  // (x,y,z) -> (X,Y,Z) Kershaw transform
{
   X = x;
   int layer = x*6.0;
   double lambda = (x-layer/6.0)*6;
   switch (layer)
   {
      case 0:
         Y = left(epsy, y);
         Z = left(epsz, z);
         break;
      case 1:
      case 4:
         Y = step(left(epsy, y), right(epsy, y), lambda);
         Z = step(left(epsz, z), right(epsz, z), lambda);
         break;
      case 2:
         Y = step(right(epsy, y), left(epsy, y), lambda/2);
         Z = step(right(epsz, z), left(epsz, z), lambda/2);
         break;
      case 3:
         Y = step(right(epsy, y), left(epsy, y), (1+lambda)/2);
         Z = step(right(epsz, z), left(epsz, z), (1+lambda)/2);
         break;
      default:
         Y = right(epsy, y);
         Z = right(epsz, z);
         break;
   }
}
```



(a) $\varepsilon_y = \varepsilon_z = 1$ (b) $\varepsilon_y = 0.3,\ \varepsilon_z = 1$ (c) $\varepsilon_y = 1,\ \varepsilon_z = 0.3$ (d) $\varepsilon_y = \varepsilon_z = 0.3$

Camier et al. *Accelerating high-order mesh optimization using finite element partial assembly on GPUs.* Journal of Computational Physics (2023).

Kershaw Benchmark: Timing Results



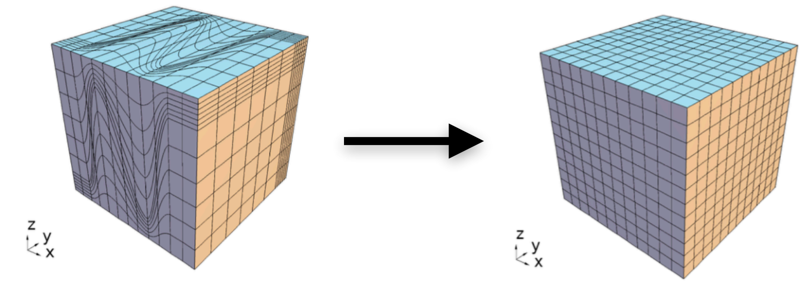
- Timing comparison on Lassen, a Livermore Computing supercomputer, for full- and partial-assembly on CPU vs partial-assembly on GPU.
  - CPU - 36 cores with 44 CPUs per core.
  - GPU - 1 core with 1 GPU and 4 CPUs per core.

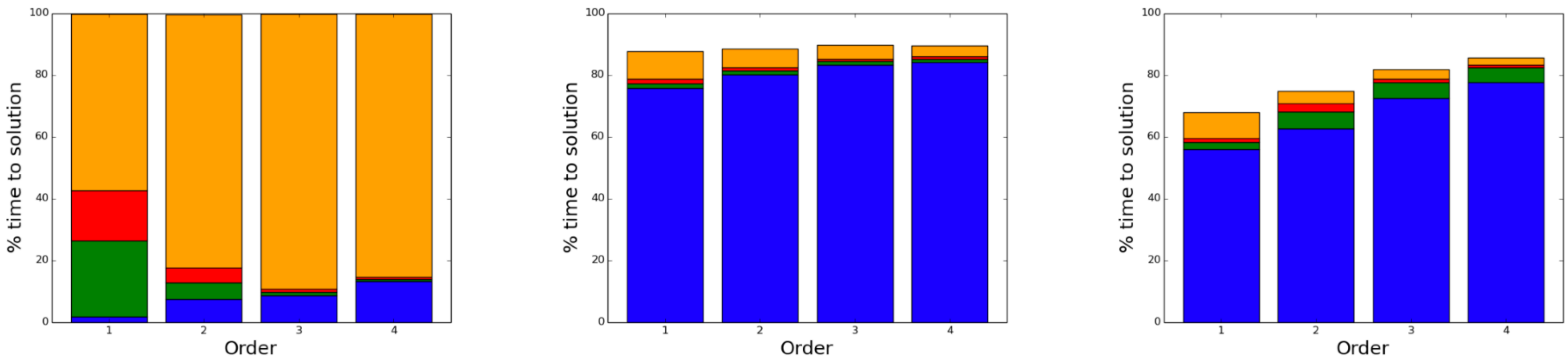
	Time to solution (sec)			
	$p = 1$	$p = 2$	$p = 3$	$p = 4$
$\mathrm{CPU}^{\mathrm{FA}}$	2.9	31.1	489.6	2868.8
$\mathrm{CPU}^{\mathrm{PA}}$	18.0	41.0	128.5	298.0
$\mathrm{GPU}^{\mathrm{PA}}$	0.4	0.9	3.9	8.5
	Speedup ($\mathrm{GPU}^{\mathrm{PA}}$ vs $\mathrm{CPU}^{\mathrm{PA}}$)			
	42$\times$	**43**$\times$	**32**$\times$	**35**$\times$

$\mathcal{O}(30\times)$ speed-up on GPUs versus CPUs

PA benefecial on CPUs for higher p.

Kershaw Benchmark: Timing Results





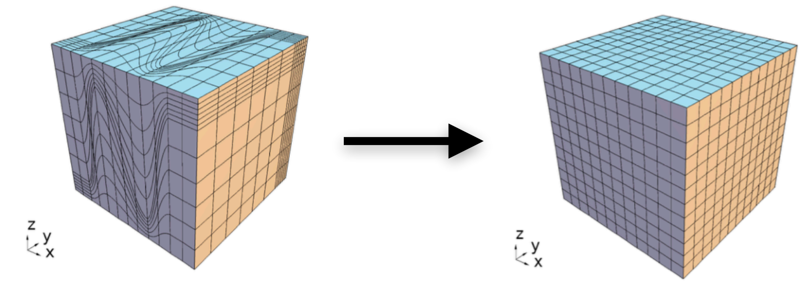
(a) $\mathrm{CPU}^{\mathrm{FA}}$ (b) $\mathrm{CPU}^{\mathrm{PA}}$ (c) $\mathrm{GPU}^{\mathrm{PA}}$

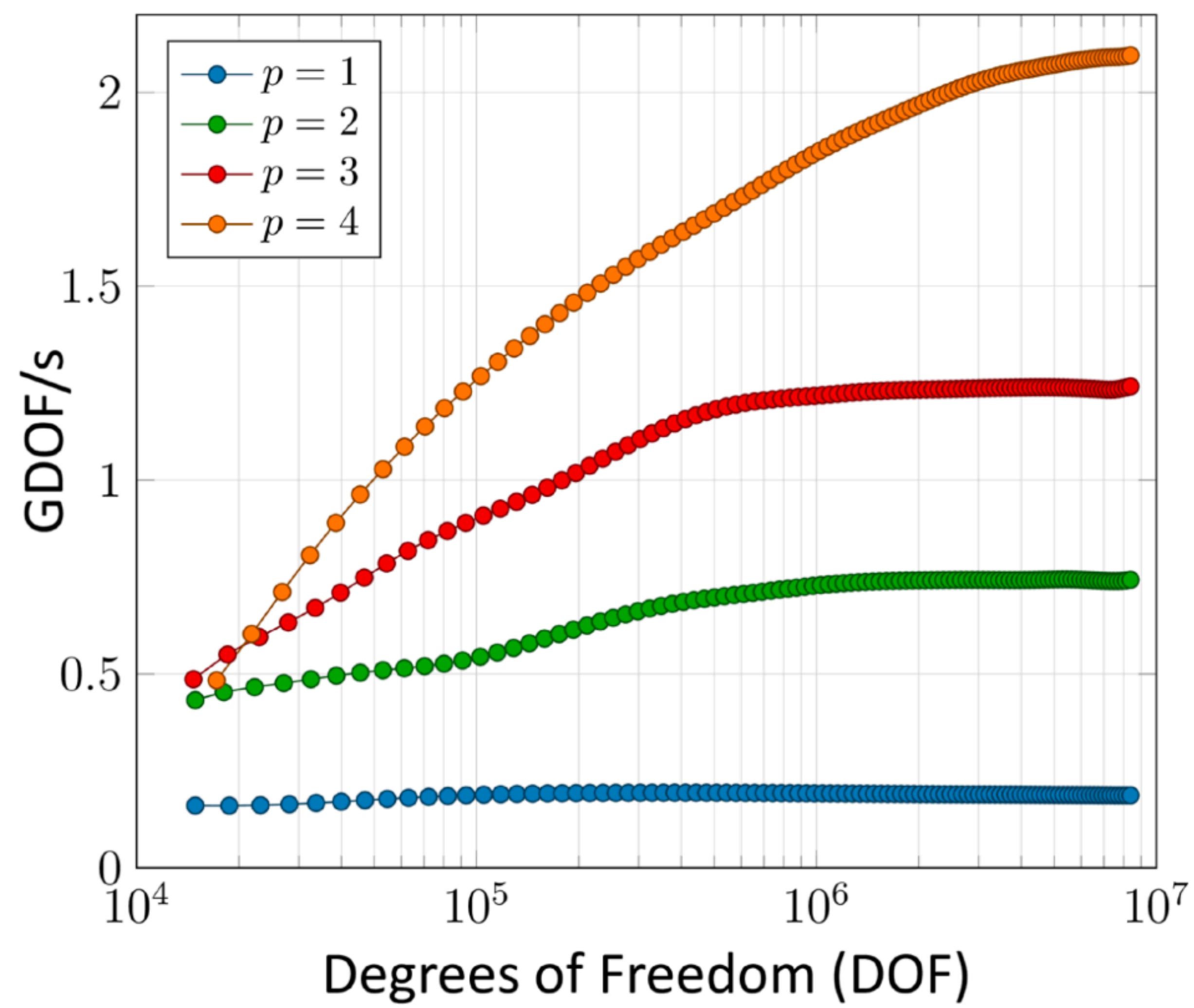
$\Delta x = (\partial^2 F)^{-1} \partial F$ Line-Search ∂F $\partial^2 F$

Partial-assembly virtually eliminates the assembly cost associated with $\mathscr{H}$.

GPUs provide further acceleration.

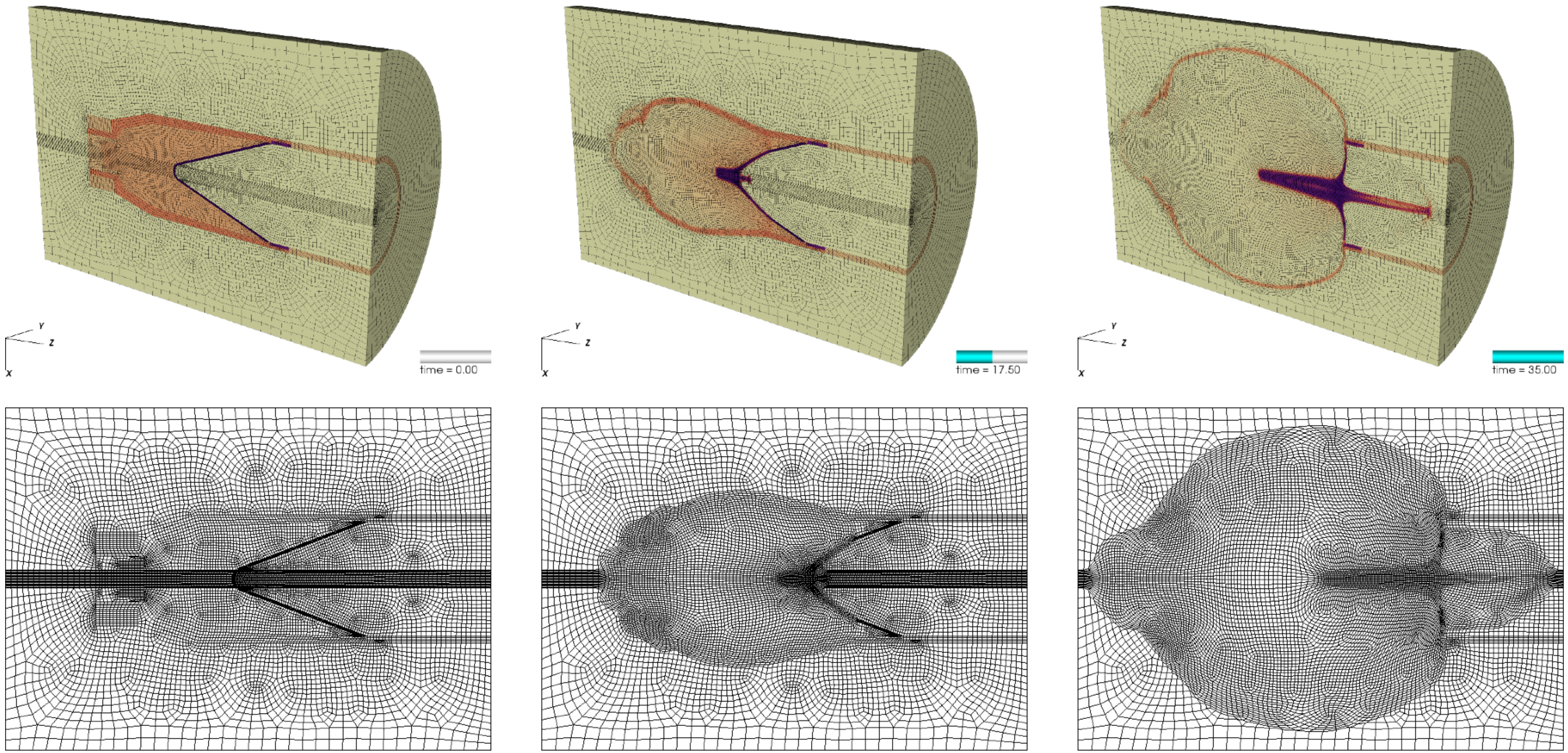
Kershaw Benchmark: Throuput





Throughput over 1 Newton iteration on 4 NVIDIA V100s

GPU Acceleration for MultiMaterial ALE with Solution-Driven Adaptivity

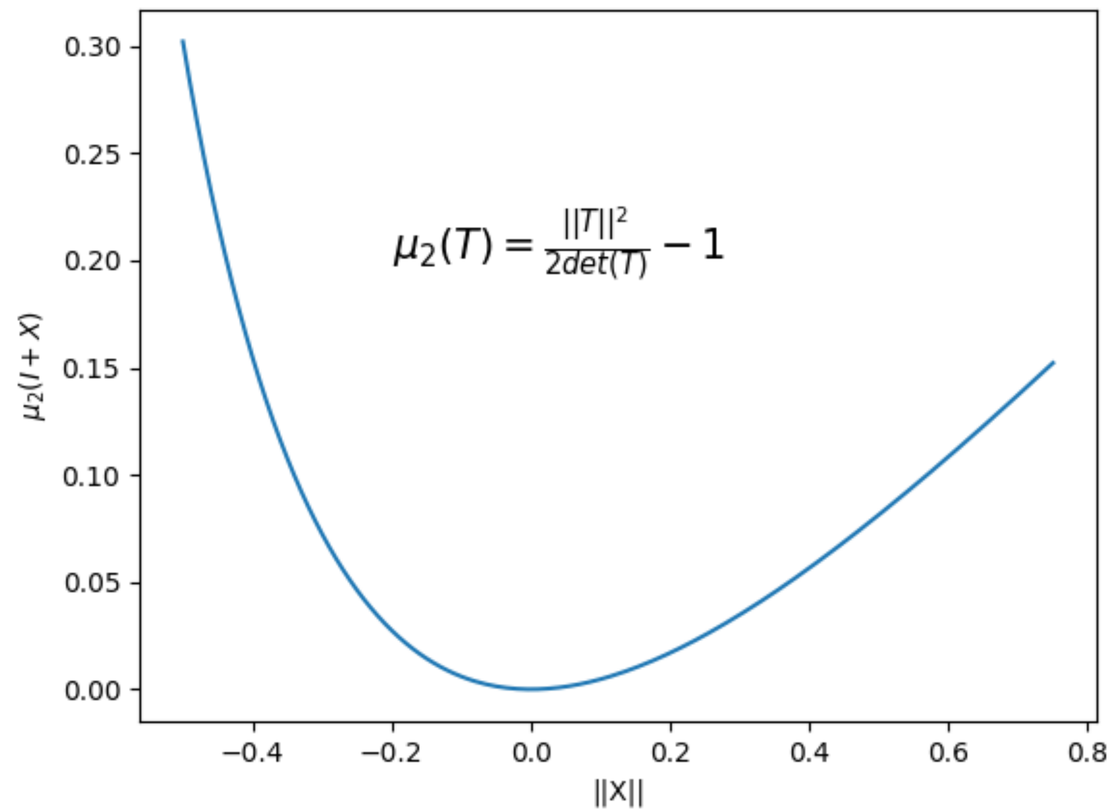


Density (top) and 2nd order mesh (bottom) for the ALE shaped charge GPU simulation.

20x speed-up for TMOP step in the solver.

Metric Linearization

- TMOP problem requires multiple Newton iterations due to the non-linearity from the mesh quality metric.



- We can linearize the problem using Taylor expansion when:
  - The target matrix W is constant throughout the domain. [Dependence of Hessian only on A]
  - Deviation of current jacobian A is *small* with respect to the target Jacobian W.

Metric Linearization

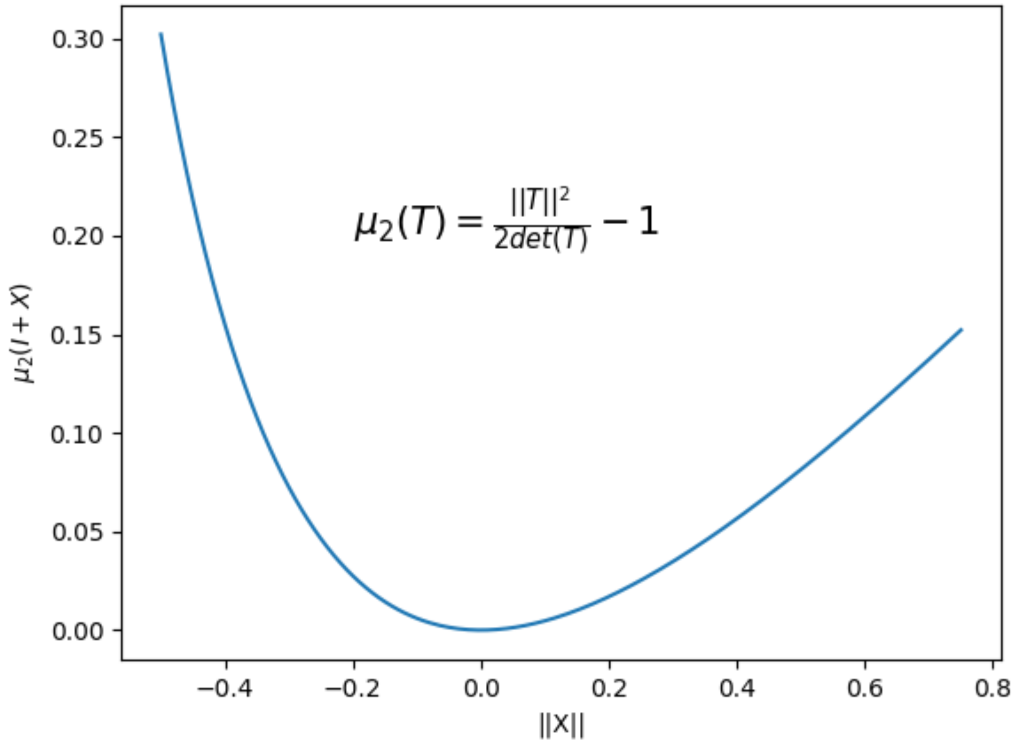
- Linearize the metric around the minima I using $T = I + X$:

$$\mu(T) = \mu(I+X) = \mu(I) + X : \frac{\partial \mu}{\partial T}\Big|_{T=I} + \frac{1}{2} X : \frac{\partial^2 \mu}{\partial T^2}\Big|_{T=I} : X + \ldots$$

$$\mu(T) = \frac{1}{2} X : \frac{\partial^2 \mu}{\partial T^2}\Big|_{T=I} : X$$

$$\frac{\partial \mu}{\partial T} = X : \frac{\partial^2 \mu}{\partial T^2}\Big|_{T=I}$$

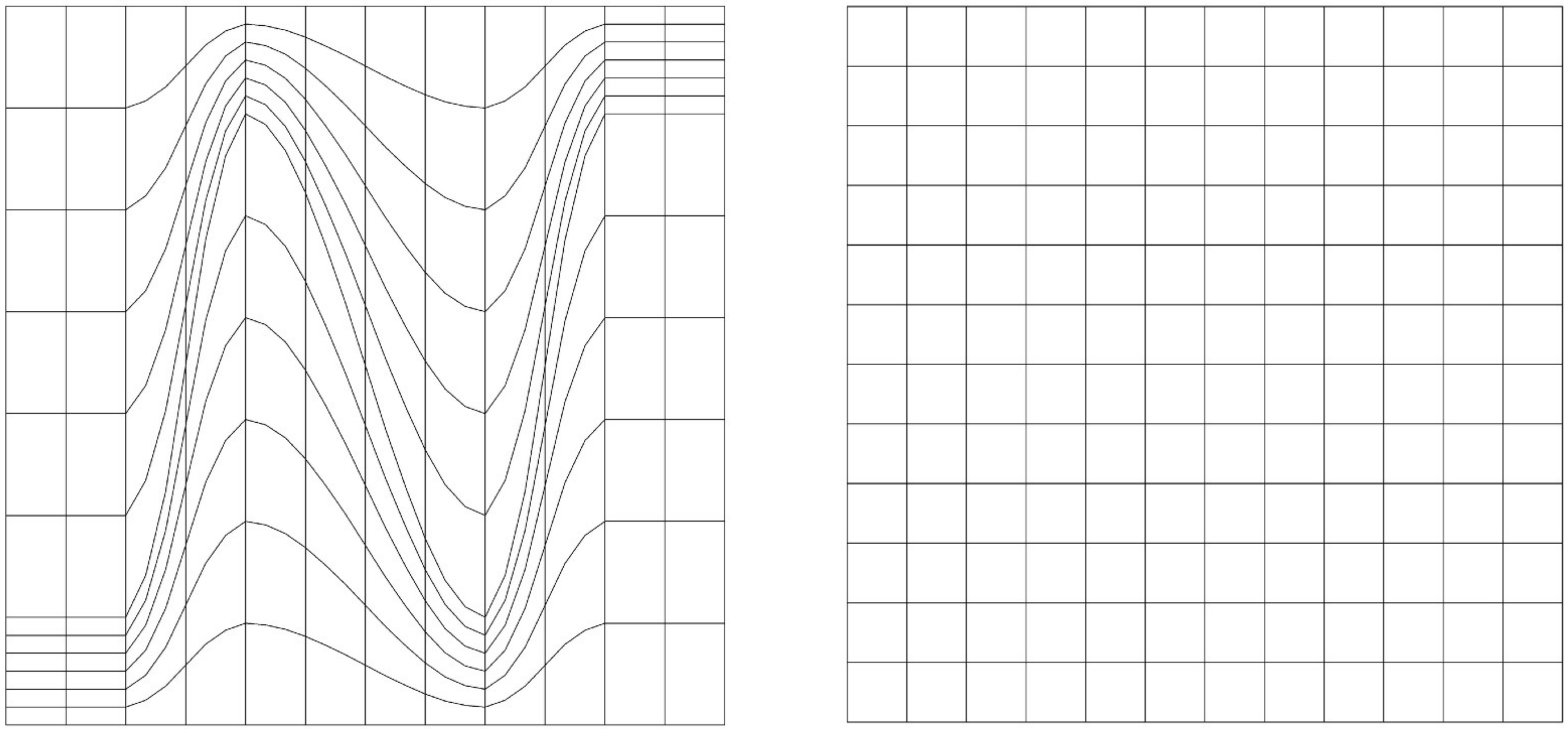
$$\frac{\partial^2 \mu}{\partial T^2} = \frac{\partial^2 \mu}{\partial T^2}\Big|_{T=I}$$



Compute $\frac{\partial^2 \mu}{\partial T^2}\Big|_{T=I}$ once and re-use at all quadrature points.

Problem is quadratic so 1 Newton iteration is sufficient.

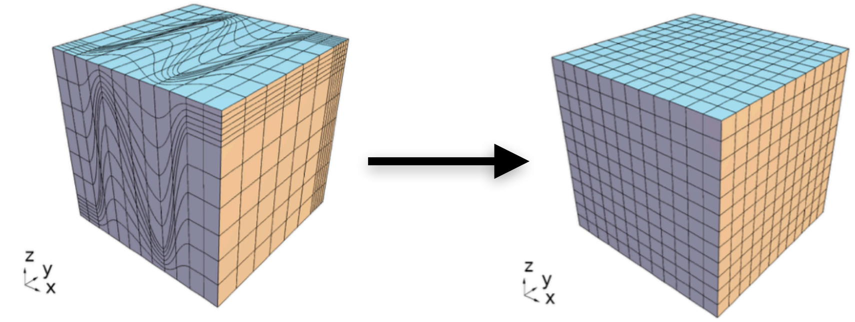
Metric Linearization - Results



Kershaw transformed 2nd order mesh optimized using ideal shape + shape metric.

- Linearized problem requires 1 Newton iteration with 82 MINRES iterations.
- Non-linear form requires 8 Newton iterations with a total of 1457 MINRES iterations.

Metric Linearization + GPU acceleration - Kershaw

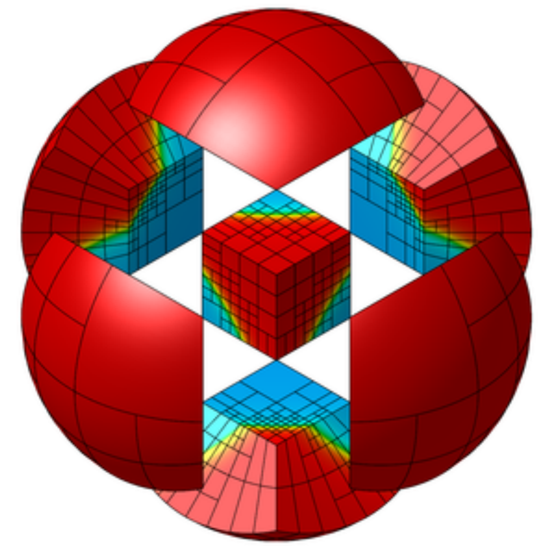


- Further $\mathcal{O}(10\times)$ speed-up on GPUs + Partial assembly + Linearization in comparison to GPUs + Partial assembly.
  - $\mathcal{O}(300\times)$ speed-up in comparison to CPUs + Partial assembly.

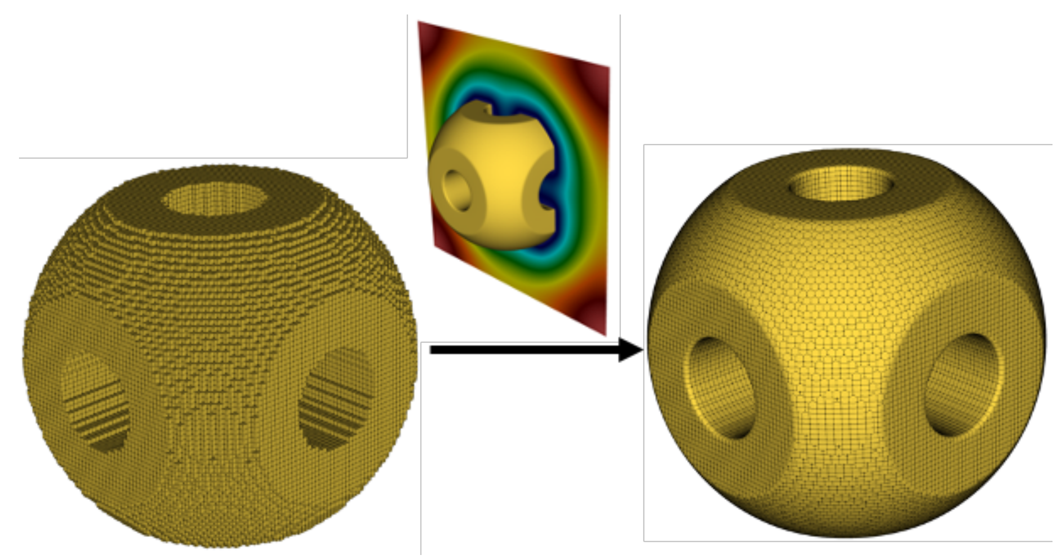
	Time to solution (sec)			
	$p=1$	$p=2$	$p=3$	$p=4$
$\mathrm{CPU}^{\mathrm{PA}}$	18.0	41.0	128.5	298.0
$\mathrm{GPU}^{\mathrm{PA}}$	0.4	0.9	3.9	8.5
$\mathrm{GPU}^{\mathrm{PA+Linearized}}$	0.05	0.16	0.32	0.8
	Speedup ($\mathrm{GPU}^{\mathrm{PA+Linearized}}$ vs $\mathrm{GPU}^{\mathrm{PA}}$)			
	8$\times$	**5**$\times$	**10**$\times$	**10**$\times$
	Speedup ($\mathrm{GPU}^{\mathrm{PA+Linearized}}$ vs $\mathrm{CPU}^{\mathrm{PA}}$)			
	360$\times$	**256**$\times$	**400**$\times$	**372**$\times$

Summary & Future Work

- High-order mesh optimization using TMOP.
- $\mathcal{O}$(30x) speed up for mesh optimization using partial-assembly.
- Another $\mathcal{O}$(10x) gain from metric linearization.
- Functionality based on open-source high-order FEM library, MFEM.
    - Learn more about it at the virtual MFEM Community workshop in October: ***www.mfem.org/workshop***.
- Partial assembly and matrix-free action for other TMOP-based functionalities in future.



mfem.org



Barrera et al. *High-Order Mesh Morphing for Boundary and Interface Fitting to Implicit Geometries.* Computer Aided Design (2023).

CASC

Center for Applied Scientific Computing

Lawrence Livermore National Laboratory