Mesh Quality Optimization via the Target-matrix Paradigm

Enabling automatic mesh generation, along with mesh updating for more accurate & efficient simulations.

**Motivation**
- Automatic Initial-Mesh Post-Processing:
  - Quality Improvement: Shape, Size, Orientation
  - Removal of Inverted Elements
  - Triangle, Tetrahedral, Quadrilateral, Hexahedral Meshes
- Application-embedded Mesh Updating:
  - Adaptive Mesh R-Refinement and Local Swapping
  - Updating Meshes on Deforming Domains
  - Mesh Rezone in Arbitrary Lagrange-Eulerian Simulations
- Target-matrix Paradigm: do all of the above as special cases of a unified theory of mesh optimization
  - Each canonical smoothing algorithm is encapsulated within a wrapper for non-expert users
  - Rapid deployment of custom-build smoothers
  - Allows use of maintainable software
  - Mesh quality optimization is a cross-cutting technology

**Target-matrix Paradigm Implemented in Mesquite Library**
- Low-level API implements key objects in Target-matrix Paradigm:
  - Finite Element Mappings
  - Active Jacobian Matrices
  - Target Calculators
  - Local TMP Quality Metrics, along with Analytic Gradients & Hessians
  - Power-mean and Hierarchical Objective Functions
  - TMP-based Local Relaxation Solvers
- Special Cases Implemented as Wrappers
  - Mesquite doubles as both an application service and a research platform (so need to balance flexibility vs. efficiency)
  - Mesquite is limited to node-movement, but compliments other interoperable ITAPS tools: (e.g., swap, refine, geometry, visualization)

**Target-matrix Paradigm Enables Rapid Delivery of Custom-Built Smoothers**
- The Application: Computational Mechanics Simulation Code with Explicit-Time Stepping
  - The Mesh Quality Issue: Generated mesh contains needlessly small edge-lengths that overly constrained the time-step size. Calculation therefore ran too slowly
  - The Customers' Question: Can Mesquite re-locate the mesh vertices so that the small-edge-lengths are eliminated?
- The Research Question: Can we devise a suitable Target-construction algorithm within TMP to solve this problem?
- Observations:
  1. All the mesh edge-lengths were the same, then none of them would be needlessly small.
  2. Element shapes should ideally be squares since that is the goal of the Paving Algorithm
- Conclusions:
  1. A Shape=Size metric is needed (equal-area squares have equal edge-lengths)
  2. The Target-matrix should be W = L^1, where L = average edge length in initial mesh, and W is the identity matrix (represents the shape of a square element)
- Results:
  1. A new, concrete Target-construction algorithm was implemented in Mesquite, based on the above.
  2. The Proposed Algorithm Met the Customers' Needs (see figure)

**Mathematical Description of the Target-matrix Paradigm**
- 1. Objective Function
  \[
  F(x) = \frac{1}{N_x} \sum_{n=1}^{N_x} \left[ \sum_{k=1}^{N_t} \mu(t_k) \right]^{1/p}
  \]
  Measure of Global Mesh Quality

- 2. Element Mappings & Sample Points
  Master Element
  \[ X = X(\xi, \eta) \]
  Physical Element
  \[ X = X(\xi, \eta) \]

- 3. Active Jacobian Matrix
  \[
  \begin{align*}
  \mathbf{E} &= \frac{\partial X}{\partial \mathbf{u}} \\
  \mathbf{J} &= \begin{bmatrix} \frac{\partial X}{\partial \xi} & \frac{\partial X}{\partial \eta} \end{bmatrix} \\
  \end{align*}
  \]
  Active Jacobian Matrix

- 4. Target Matrix
  For each Sample Point there exists a pair of active & target geometries \( (X, X') \). Target matrices represent the mapped locations in the optimal mesh.

- 5. Scaled Active Matrix
  \[
  T = A W^{-1}
  \]
  Control Shape Point/Orientation: Active Jacobian matrices order target-matrix size, when \( A \) is \( W \) of an arbitrary rotation matrix.
  Control Shape Size: Active Jacobian matrices order target-matrix size, when \( A \) is \( W \) of an arbitrary translation matrix.
  Active Shape: Active Jacobian matrices order target-mesh size, when \( A \) is \( W \) of an arbitrary translation matrix.

- 6. Local TMP Quality Metrics
  \[
  \mu = |X - X'|
  \]

- 7. Tensor Tradeoff Coefficients
  Tradeoff coefficients allow to weigh in qualitative and quantitative components in the mesh generation process.

- 8. Prior Direct Optimization Methods
  - Compliance-Target-matrix Formulation
  - \( T = A W^{-1} \)
  - Methods do not mix geometric and elastic metrics; therefore, they cannot satisfactorily avoid mesh distortions.

- 9. TRP Reports
  8. "Relaxation Solvers for TMP Quality metrics," in progress

**Digging Deeper: A result from Quality Metrics Paper (Ref. 3)**
- Propagation 1: Let \( T \) be any valid \( 2 \times 2 \) matrix and define \( \mu = \sum_{T} | T - 1 | \mu_1 \). Then \( T \) is a scaled \( T \) matrix.
- Propagation 2: \( \mu(T) = |T| \mu_1 \) is a rigid-body matrix.
- Propagation 3: \( \mu(T) \) is generally non-unique if \( T \) is an affine transformation.
- Propagation 4: \( \mu(T) \) is differential on the set of \( 2 \times 2 \) matrices that are not scaled \( T \) matrices.
- Propagation 5: The stationary points of \( \mu(T) \) are rotation matrices.
- Propagation 6: \( \mu(T) \) can be expressed as \( \mu(T) = \sum_{i=1}^{4} \mu_{ii} \).

**On-Going Research & Development of Target-Matrix Paradigm**

**Optimizing Quality of High-Order-Node Meshes**
- The Issue: To create finite element meshes with higher-order nodes, most mesh generators first create linear elements and then add the high-order nodes later. After the high-order nodes are created, they must be "strained" to the geometry to achieve high accuracy. Snapshots often create poor quality or even inverted elements.

**Sliver-removal for a viscous CFD Mesh**
- The Problem: Given a high-fidelity CFD mesh, use mesh optimization to remove the sliver elements without distorting the boundary layer of the flow.

**Local Relaxation Solvers for TMP Quality Metrics**
- Minimize \( || f(x) ||^2 \)
- Using block coordinate descent (i.e., Gauss-Seidel)

**Mesh Rezoning for Arbitrary Lagrange-Eulerian Methods**
- \( \text{K-rm, A-rm, L-rm} \) are all contain mesh rezoning algorithms to prevent mesh tearing and improve quality.

**Challenges**
- Derive the update rules for each TMP metric.
- Ensure the rules converge in convergent iteration.
- Rules depend on the selected metric.
- Operations count & timing tests.