# Mesquite: Mesh Quality Improvement Toolkit

August 22, 2008

**Lori Diachin**

**Science & Technology PAD/Computation/CAR**

**This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344**

**Lawrence Livermore National Laboratory**

# Overview

- Introduction to mesh quality improvement
  - Why worry?
  - Quality metrics and solution methods
- Mesquite softare
  - Capabilities and design principles
  - Basic classes
  - User interfaces
- Parallel mesh smoothing
  - Basic algorithm
  - Parallel Mesquite
- Use in application and downloading Mesquite

# Mesh Quality Improvement

# Why worry about mesh quality?

- In general, mesh size and quality affects:
  - **Solution efficiency** (e.g. Axelsson, 1976; Fried 1972; Axelsson and Barker, 1984)
    - Iterations grow as a function of N
    - Iterations grow as a function of minimum angle
  - **Solution accuracy**
    - Solution from iterative solver is less accurate
    - Discretization error adversely affected by distorted elements (e.g. Babuska and Aziz, 1976)
- In the final analysis, mesh quality really depends on the application
  - stretched elements are more accurate than equilateral elements for boundary layer flow

# Solver Efficiency

- Convergence of linear solvers depends on the eigenvalues of the matrix
  - For uniform spectral distribution number of iterations is bound by (Axelsson and Barker, 1984)

$$I_\varepsilon < 1/2 \text{ sqrt}(\kappa_N / \kappa_1) \ln (2/\varepsilon) + 1$$

  - In general, the number of iterations required for convergence of the solvers increases as *N* increases
- Isolated extremal eigenvalues
  - Condition number overestimates iteration bound
  - New bounds based on exact and finite precision arithmetic (Axelsson and Lindskog, 1986; Jennings, 1977; Notay, 1993; Strakos, 1991)

# Relationship with Mesh Quality

- For triangular meshes the eigenvalues depend on the shape of the mesh and the number of elements incident on a vertex

$$\kappa_1 > c1 * \Delta_{min} * \min A_T$$

$$\kappa_N < c2 * \Delta_{max} * (\max A_T + \max (1/\sin \eta_{min}))$$

$A_T$ : Area of triangle T

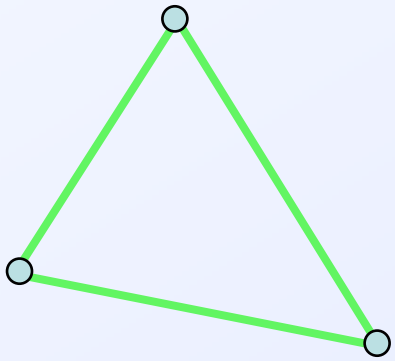
$\eta_{min}$: Minimum angle in the mesh

$\Delta$ : number of elements incident on a vertex

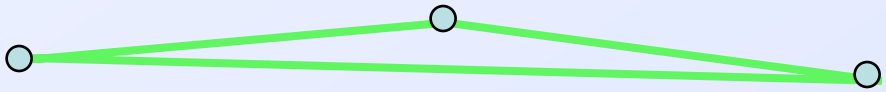
- As *N* grows, $\kappa_1$ decreases
- As $\eta_{min}$ decreases, $\kappa_1$ and $\kappa_N$ are adversely affected

# Expectations

Elements tending toward equilateral



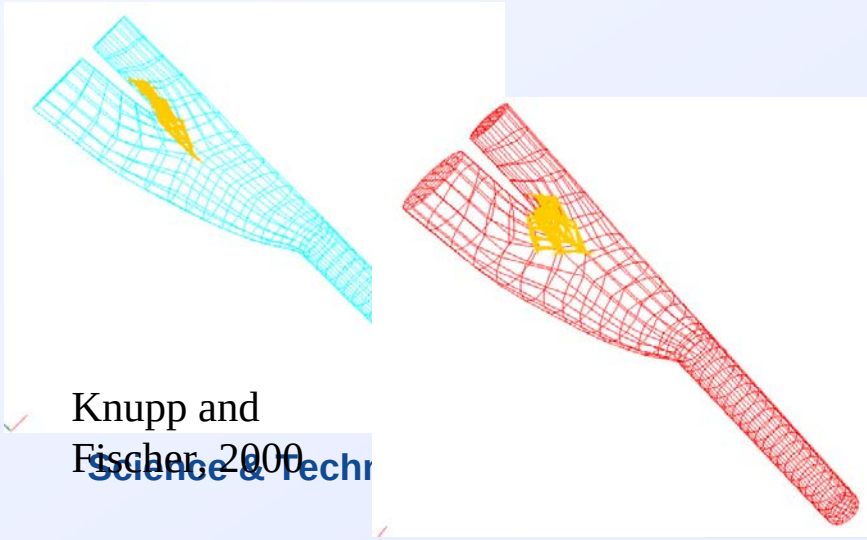
will give better results, faster, than distorted or stretched elements for isotropic problems



# Improving mesh quality can dramatically affect time to solution

- Arteriovenous Graft Turbulent Flow Simulation
- Compute maximum shear stress with high order spectral methods
  - Poorly-shaped Elements Increase CG Solver Iterations
- Mesh Optimized by Condition Number
  - reduced maximum number of solver iterations from 169 to 150
  - reduced the average condition number from 18.06 to 15.46 (about a 17% savings).

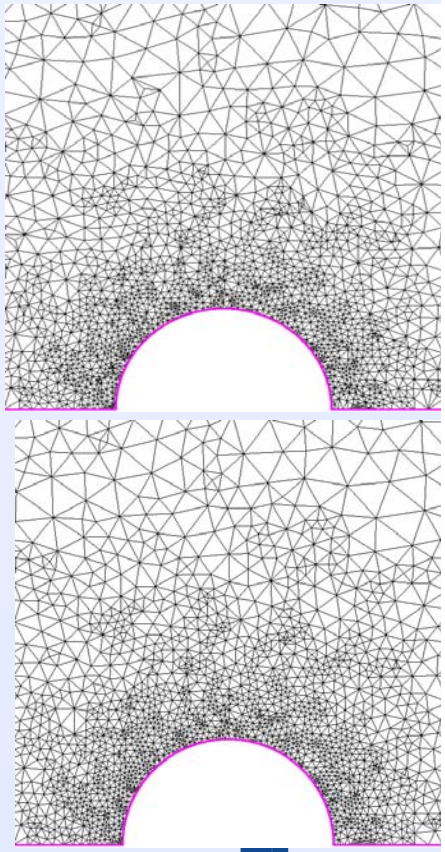
*Four hours of applications solver time was traded for 19 minutes of mesh smoothing time.*



Knupp and Fischer, 2000

- **Compressible Flow**
- **Mesh Optimized w/ Active set solver**
  - **Improved the convergence rate by 25%**

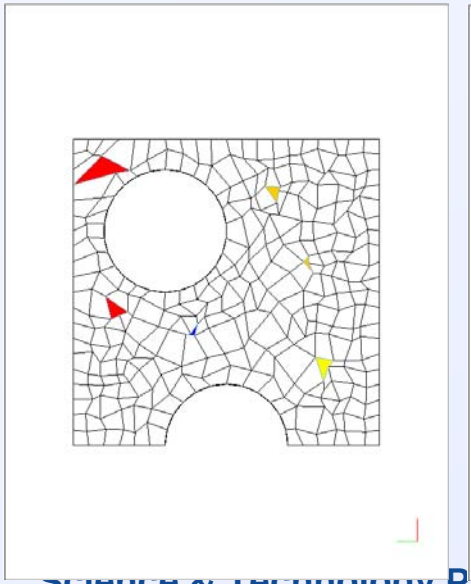
***Mesh improvement cost less than one multigrid iteration***

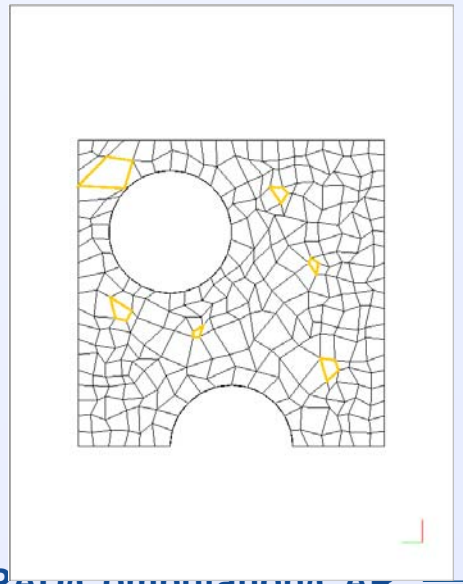


Freitag and Ollivier-Gooch, 1998

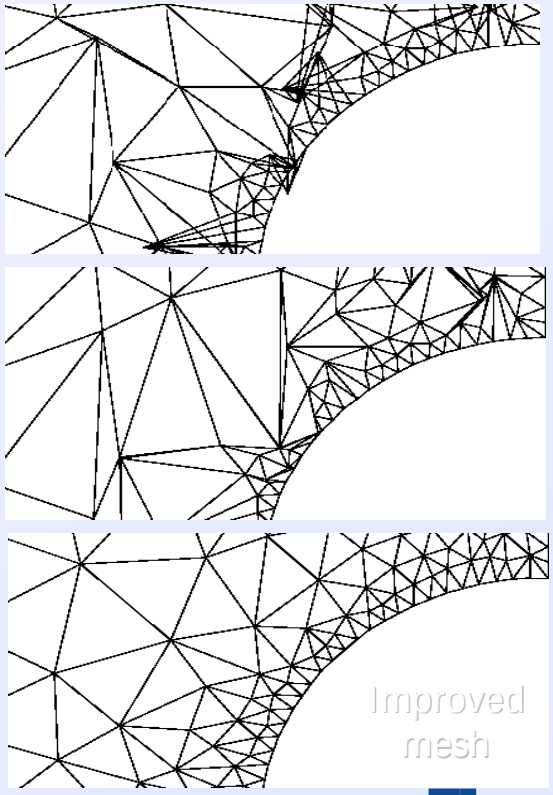
# Mesh untangling algorithms reduce unnecessary effort

- Few hex-meshing algorithms guarantee the quality of the mesh
  - Inverted elements are produced
- Mesh untangling algorithms can remove inverted elements quickly
  - Eliminates the need to remesh
  - Eliminates the need to decompose the geometry





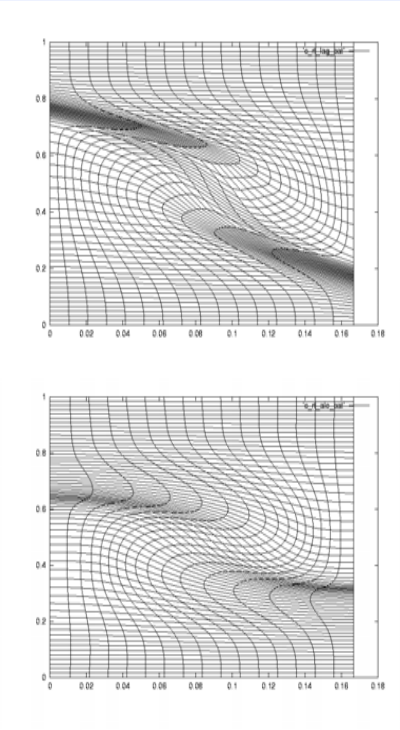
Knupp



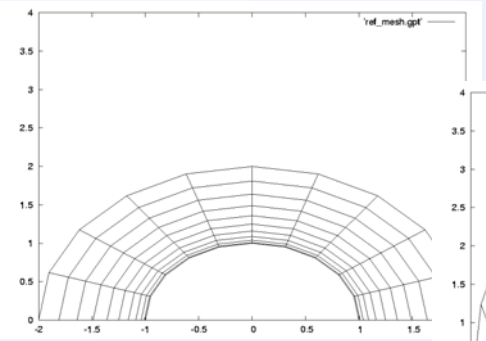

Freitag

# Mesh alignment can improve simulation results

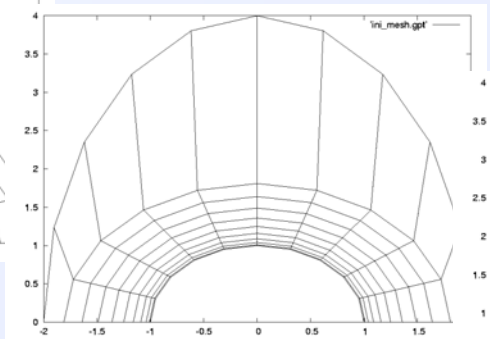
- Moving vertex positions to match a vector or scalar field
- Improving ALE mesh quality while preserving flow characteristics
- Deforming a mesh to match a perturbed geometrical domain



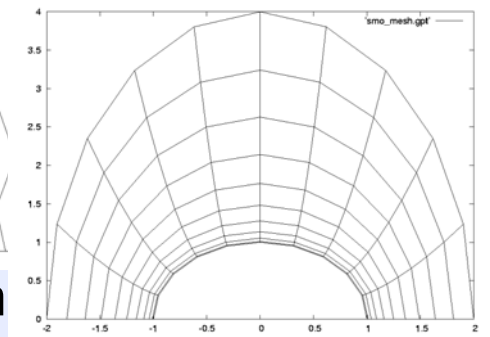
Knupp, Shashkov, Garimella 2000



Original Mesh



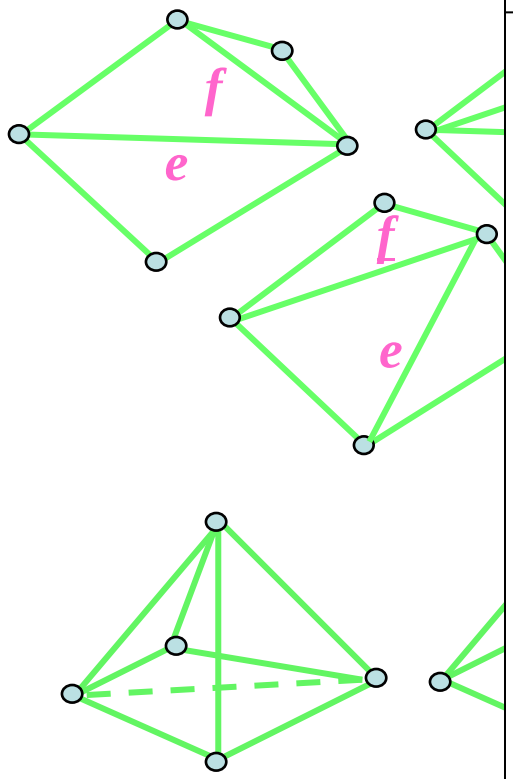
Deformed Mesh



Smoothed Mesh

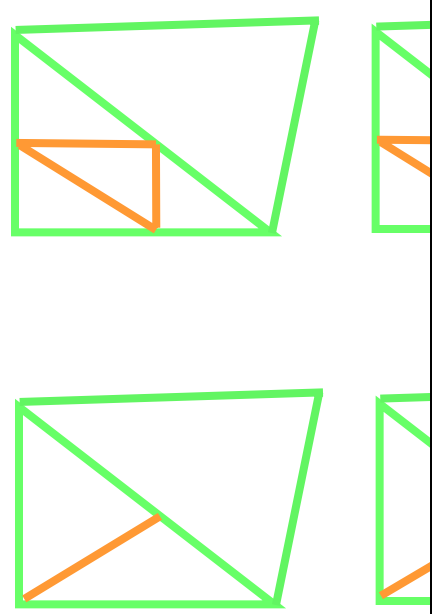
# There are three primary techniques for improving the quality of existing meshes

## Edge or Face Flipping



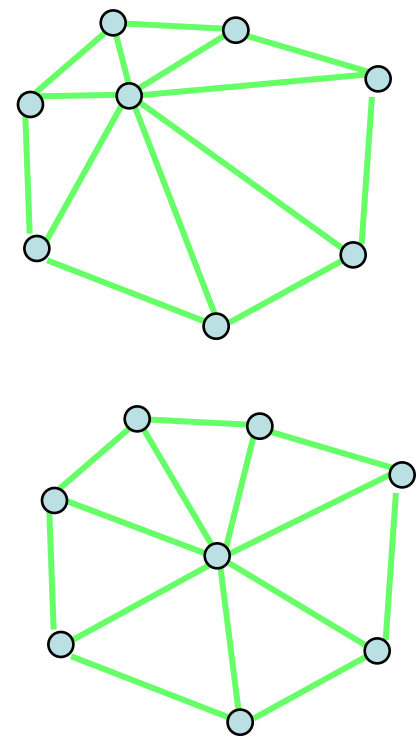

Modify topology witho
grid point location

## Refinement/Coars



Adding or deletin
improve local reso

## Node Movement



Moving grid points without changing mesh topology

# Node movement: Smoothing

***Definition*:** "Smoothing is a procedure for improving mesh quality via a node-movement strategy in which a non-linear system of equations is solved."

Discrete Equation: $f(\ldots, x_{ij}, y_{ij}, \ldots) = 0$

Iterative Form (Most often Picard):

$$x_{ij}^{n+1} = g(\ldots, x_{ij}^{n}, y_{ij}^{n}, \ldots)$$

$$y_{ij}^{n+1} = h(\ldots, x_{ij}^{n}, y_{ij}^{n}, \ldots)$$

Can do single-vertex or all-vertex.

# Laplace smoothing is on of the most commonly used techniques for node movement

## *Structured Grids*

PDE: $x_{\xi\xi} + x_{\eta\eta} = 0$

$$y_{\xi\xi} + y_{\eta\eta} = 0$$

Discrete Form

$$(x_{i+1,j} - 2x_{ij} + x_{i-1,j}) \big/ (\Delta\xi)^2 +$$

$$(x_{i,j+1} - 2x_{ij} + x_{i,j-1}) \big/ (\Delta\eta)^2 = 0$$

Iterative Form

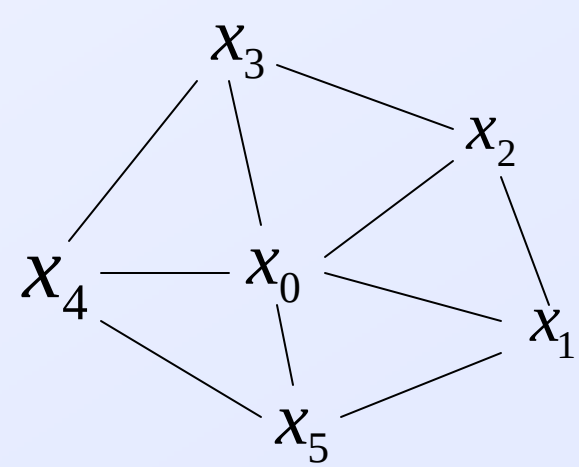
$$x_{i,j}^{n+1} = \frac{\dfrac{x_{i+1,j}^n + x_{i-1,j}^n}{(\Delta\xi)^2} + \dfrac{x_{i,j+1}^n + x_{i,j-1}^n}{(\Delta\eta)^2}}{2\left(\dfrac{1}{(\Delta\xi)^2} + \dfrac{1}{(\Delta\eta)^2}\right)}$$

## *Unstructured Grids*

No PDE invoked

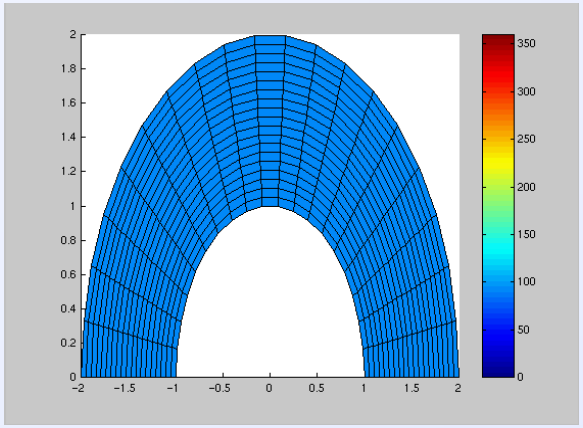
Discrete Form: $Mx_0 - \sum_{i=1}^{M} x_i = 0$

Iterative Form: $x_0^{n+1} = \frac{1}{M} \sum_{i=1}^{M} x_i^n$

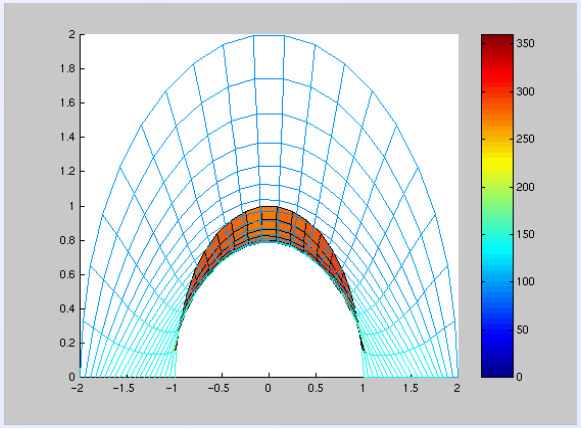


# Laplacian smoothing has both pros and cons

- Move the free vertex to the geometric center of the adjacent vertices
- Quality improvement is *not* guaranteed
  - Can result in invalid, or tangled, meshes
- Computationally inexpensive
- Easy to implement
- Constrained Laplacian smoothing only moves vertex if quality is improved [Freitag, 1998]




Original Mesh




Smoothed Mesh

# Node movement: optimization

***Definition:*** "Mesh quality optimization is the process of changing nodal positions to find the extremae of some scalar objective function that measures one or more aspects of mesh quality."

Two parts:

1) Problem Formulation (metrics and objective functions)
2) Problem Solution

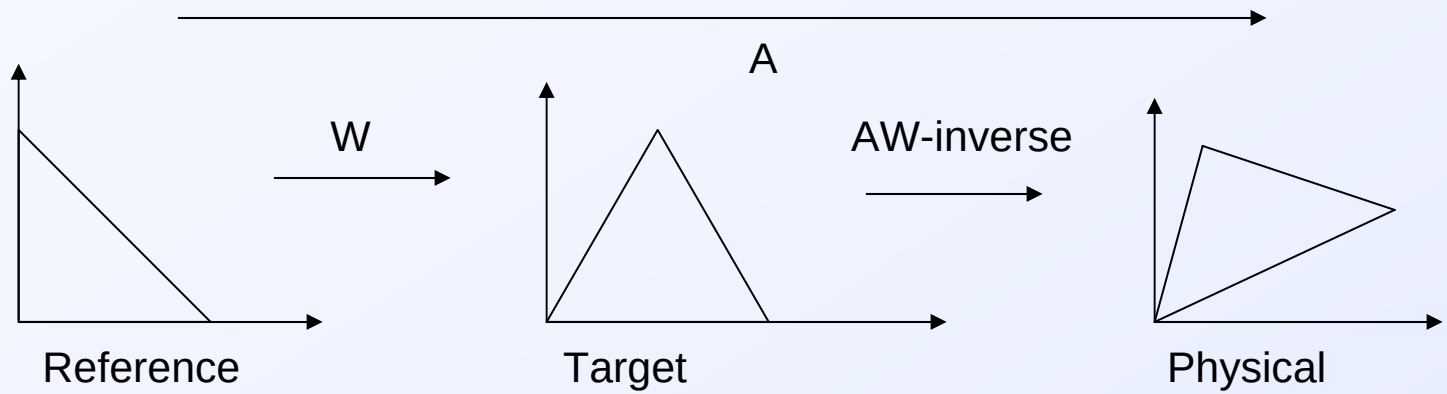
*Connection between Optimization & Smoothing*: For unconstrained optimization, extremae of the objective function occur where the gradient is zero. Setting the gradient of the objective function to zero yields a non-linear set of equations that can result in a smoothing scheme.

# Problem Formulation: Quality Metrics

- Classic *a priori* geometric criteria
  - Ratio of volume to face areas (e.g., Shephard and Georges, 1991, Bank 1994)
  - Angle-based (e.g., Freitag, et al. 1995)
  - Distortion metrics (e.g, Canaan, 1998)
  - Element condition number & matrix norms (Knupp, 1999)

- *A posteriori* local error analysis (e.g., Bank and Smith, 1997, Berzins, et. al., 2000 )

# The Target Matrix Paradigm: $T = A\,W^{-1}$




- Provides a solid math foundation for direct mesh quality optimization based on Jacobian matrices: *A*.
  - *A* describes local shape, size, and orientation.
- Target matrices, *W*, are the weight functions which are used to control the mesh.
  - Effective mesh control by linking the weight functions to their geometric impact on the mesh
- *T* used as the building block for creating quality metrics

# Metrics can be designed to guarantee invertability

Example: Weighted Condition Number:

$$f(T) = |T||T^{-1}| = \frac{|T||adj(T)|}{|\tau|}$$

Singular at $\tau = 0$

- Singularity coincides with boundary of the set of valid elements, that is, *f=0* if and only if element is degenerate.
- This barrier approach can be used with lots of other metrics.
- Requires inverted initial mesh; use untanglers if necessary

# Problem Formulation: General form of the objective function

## Algebraic Approach

Let $T_k$ be the referenced weighted Jacobian matrix for the $k^{th}$ element of the mesh

Let $f_k = f(T_k)$ be an algebraic quality metric for element $k$. ($f$ could be shape, size, orientation, etc.). Let $\omega_k = 1 - f_k$

Form the vector of metric values: $\varpi = (\ldots, \omega_k, \ldots) \in R^N$

$\ell_2$ objective function: $|\varpi|_2 = (\sum_k |\omega_k|)^{1/2}$

$\ell_\infty$ objective function: $|\varpi|_\infty = \max_k \{|\omega_k|\}$

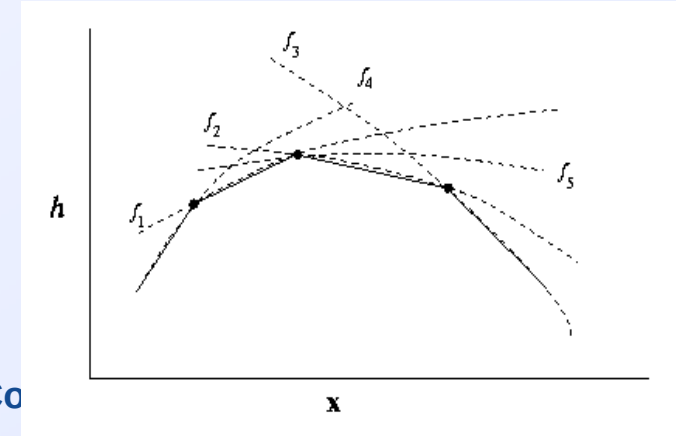
# Problem Solution: $\ell_2$ Objective Functions

- Conjugate Gradient
  - $C^1$ objective functions and quality metrics
  - Linear convergence rate
  - Requires function and gradient information
  - Polak-Ribiere search direction scheme

- Feasible Newton (Munson, 2004)
  - $C^2$ objective functions and quality metrics
  - Minimizes a quadratric approximation of a nonlinear objective function
  - Super-linear convergence
  - Requires function, gradient and Hessian information
    - Works best when the Hessian is sparse

# Solvers for $\ell_\infty$ Objective Functions

- Objective function: $min\ \varpi(x) = min\ max_k \{\omega_k(x)\}$
  - $\omega_i(x)$ are typically smooth, continuous functions
  - $\varpi(x)$ is continuous with discontinuous derivatives
- Solve using steepest descent active set methods (Freitag, 1995, Amenta, et. al. 1996, Canaan, 1998)
  - Step direction is determined from the linear convex combination of gradients in the active set
  - Step length determined by finding the location where the active set changes



# Combining Mesh Improvement Techniques

- Goals
  - Improve effectiveness
  - Maintain effectiveness without sacrificing performance
- Techniques
  - Constrained Laplacian smoothing and optimization-based techniques (Freitag, 1996)
  - Smoothing plus edge and face swapping (Freitag and Ollivier-Gooch, 1998)
  - $L_2$ and $L_\infty$ techniques (Freitag and Knupp, 1999)
  - Local and global techniques (Freitag, Knupp, Munson, Shontz, 2002)

# Mesquite Software

# Mesquite software incorporates these ideas into a stand-alone library

- Mesquite capabilities
  - Shape Quality Improvement
  - Mesh Untangling
  - Alignment with Scalar or Vector Fields
  - R-type adaptivity to solution features or error estimates
    - Maintain quality of deforming meshes
    - Anisotropic smoothing
    - Control skew on mesh boundaries

# Mesquite is designed for use in a wide variety of applications

- Mesh Types
  - Structured, Unstructured, Hybrid
  - 2D, 3D
- Element Types
  - Triangular, Tetrahedral, Quadrilateral, Hexahedral, Pyramidal currently
  - Prismatic planned, Polyhedral easily added
  - Linear currently, higher-order planned
- Customizable
  - User-defined metrics, objective functions, and algorithms
- Callable as a library
  - Mesh and geometry information obtained through simple accessor functions

# Mesquite is based on the mathematical principles of optimization-based techniques

- Element Quality: $q_i(x)\ i=1,\ldots,<n,k>$
  - A function of the vertex locations
  - Can be vertex or element based
- Mesh quality objective function

$$F = f(q_i(\mathbf{x}))\ i=1,\ldots,<n_s,k_s>$$

  - A function of element quality metrics defined on some subset of the free vertices
- Optimization problem

$$\min F \text{ subject to } c_j(\mathbf{x})$$

Element shape, $s_i$
0 = degenerate
1 = ideal

$F = 1/n\ \Sigma\ 1/s_i$

min F

# The Mesquite infrastructure provides advanced solution techniques

- State-of-the-art algorithms and metrics
  - Untangle, smooth, size, shape metrics
  - Simple (Laplace) to complex (optimization) smoothers
    - Conjugate gradient
    - Feasible Newton techniques
    - Active set solvers
  - Single-vertex and all-vertex solvers
  - Some metrics permit anisotropic smoothing
  - Combined approaches
- Efficient to run
  - Mesh culling to eliminate un-needed operations
  - Kernels written with inline functions and array-based access
  - Light-weight mesh data structure

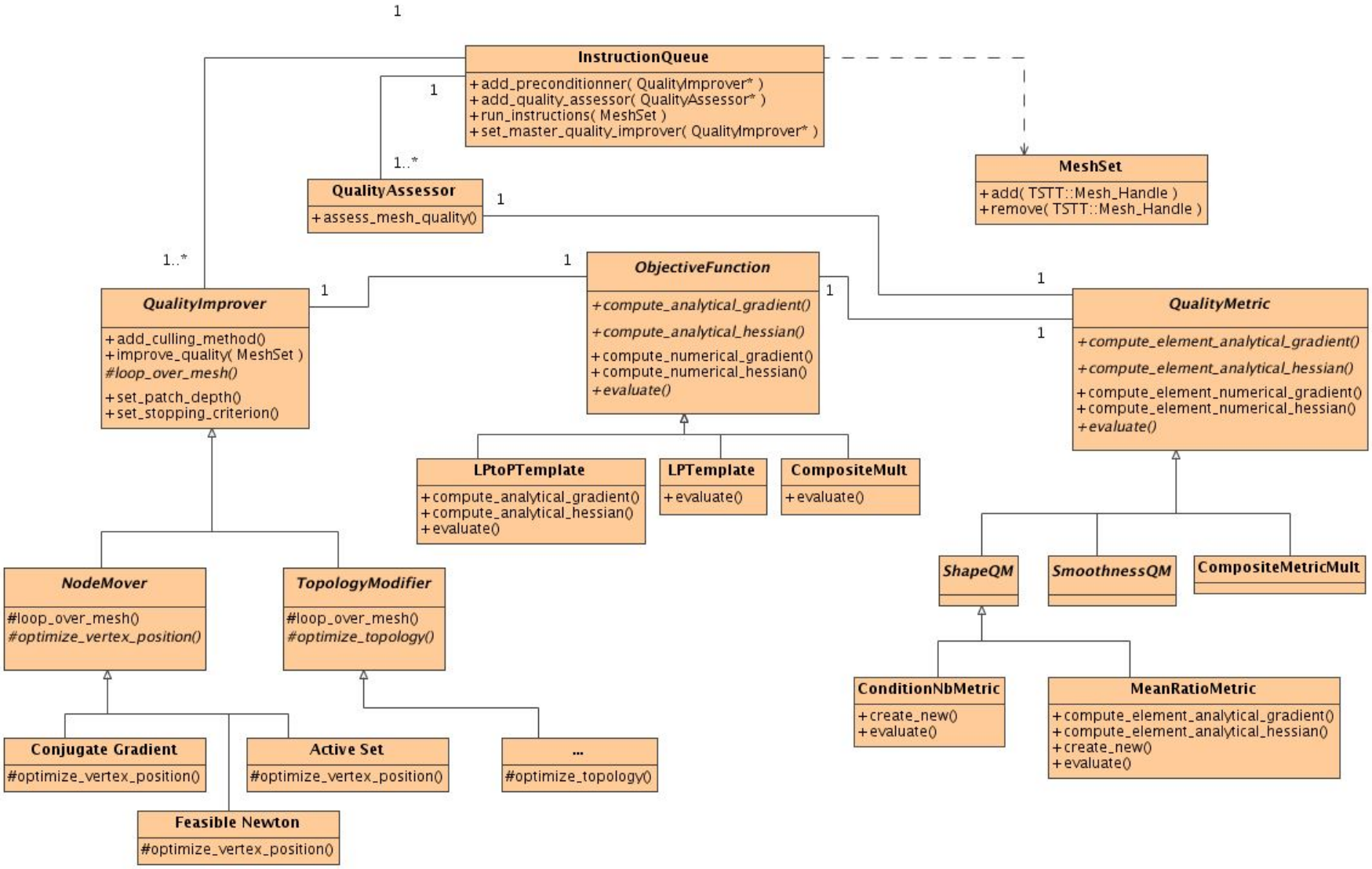
# Mesquite Software Design Principles

- Object oriented software
  - Objects correspond to mathematical abstractions
  - Use well-defined interfaces for interactions with mesh and geometry
- Provide automatic mesh improvement strategies and simple interfaces for ease of use
- Allow customization
  - Mix and match flexibility
  - User-set stopping criterion parameters

# Mesquite Architecture

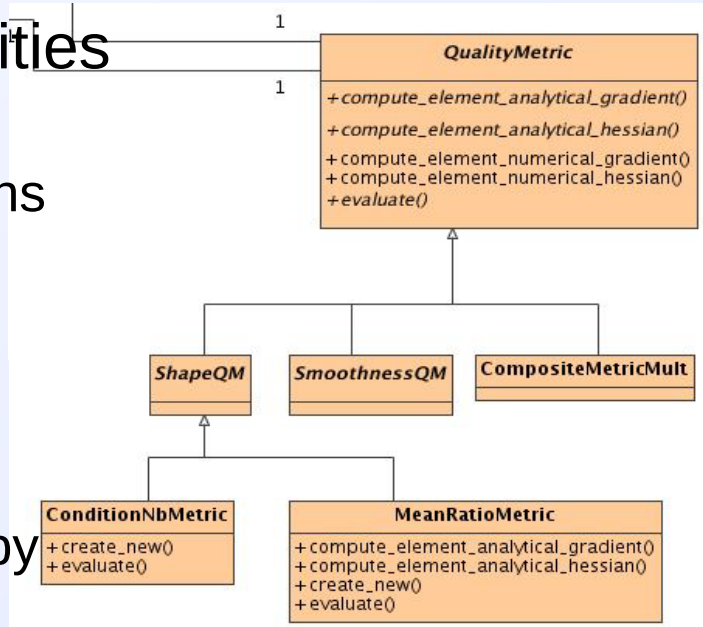
- Closely follows mathematical abstractions used to define optimization problem
- Core classes
  - `Quality Metric`
  - `Objective Function` (takes a quality metric as input)
  - `Quality Improver` (takes an objective function as input)
- Additional classes
  - `Quality Assessor`
  - `Termination Criterion`
  - `Instruction Queue`
  - `MeshSet`

# MESQUITE ARCHITECTURE



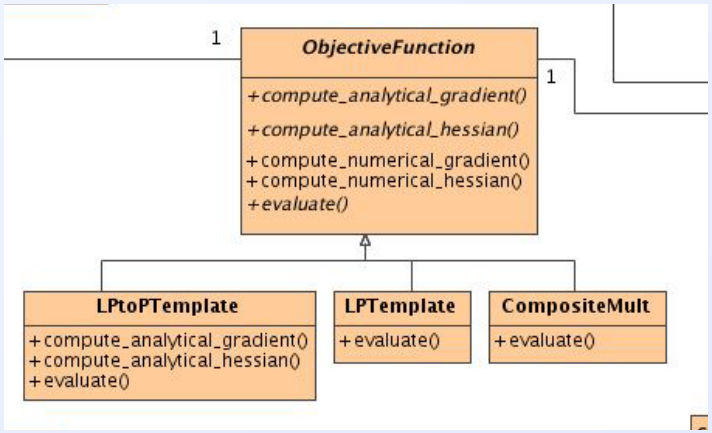
# Quality Metrics

- Provides a measure of individual entities
  - Element, e.g., mean ratio shape
  - Vertex, e.g., sum of adjacent edge lengths
- Primary functionality
  - Evaluate function
  - Gradient and Hessian for optimization algorithms
    - Numerical approximations provided by the base class
    - Analytic evaluations can be provided if desired
- Customizations
  - Non simplicial element evaluations can choose sample points and combination methods
  - Non-isotropic reference elements
  - Scale metrics or combine multiple values together



# Objective Function

- Combines values of quality metrics into a single number for the domain of the optimization problem
  - Entire mesh or a sub-mesh
- Developed as function 'templates' which take quality metrics as input
- Computes *F,* and as appropriate, $\nabla F$, and $\nabla^2 F$
  - Numerical evaluations provided for all templates
  - Analytical gradients for all smooth templates, analytic Hessians for $L_p^p$ templates

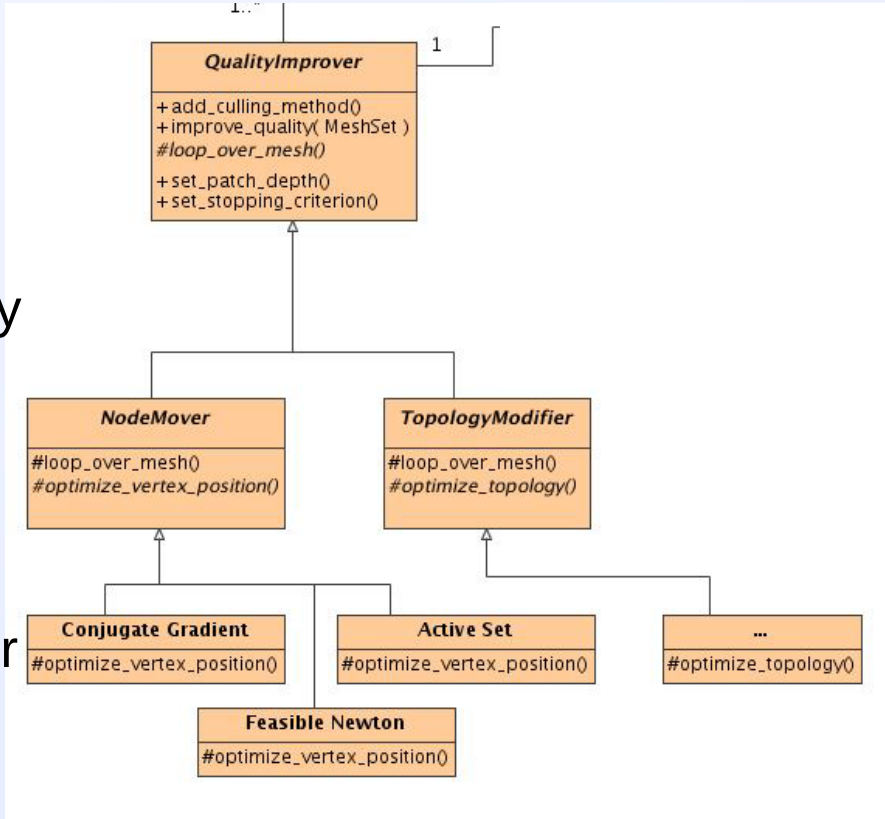


# Available Objective Function Templates

- Templates
  - $\ell_p$ (p>0) and $\ell_p^p$
  - $\ell_\infty$
  - Min/Max
- Composites
  - Add or multiply by a scalar
    - e.g. –1 to create an equivalent minimization problem
  - Add or multiply by each other
    - Allows several measures of mesh quality to be optimized simultaneously

# Quality Improvers

- Acts on objective functions (and quality metrics) to optimize mesh quality
- Domain
  - Entire set of free vertices
  - Subsets, or patches, as defined by the user
- Termination criterion
  - Inner criterion stops the optimization algorithm
  - Outer criterion stops the 'loop over mesh' iteration
- Available solvers
  - Steepest descent, Conjugate Gradient, Feasible Newton, Quasi-Newton, Active set



# Termination Criterion

- Two uses
  - Inner Criterion: terminate optimization on a submesh
  - Outer Criterion: terminate iteration over the submeshes
    - Global patches set the outer criterion to 1 pass
- 14 Available Criterion:
  - Cost, quality and progress centric types
    - Iteration count, optimality, CPU time, lack of progress, etc.
  - Disaster aversion (e.g., unbounded node movement)
  - Relative and absolute forms
  - Compound criteria (using OR)
    - May typically choose one of each type (cost, quality, progress and disaster aversion)

# Accessing Information I

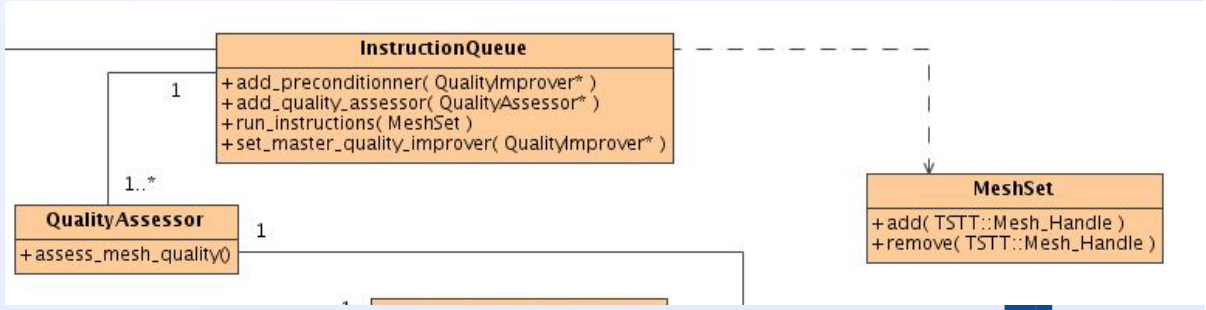
- Mesh and geometry information obtained through accessor functions
  - Community interfaces
  - Mesquite-specific interfaces
- The ITAPS mesh query interface
  - Developing common interfaces for access to mesh geometry & topology
    - Create a small set of interfaces that existing packages can support
    - Balance performance and flexibility
    - Uses a C-interface and SIDL/Babel (LLNL) for language interoperability
    - Enable both interchangeability and interoperability
  - Six Implementations underway: RPI (FMDB), ANL (MOAB), PNL (NWGrid), SUNY SB/BNL (Frontier), UBC (Grummp)
  - Information at *www.itaps-scidac.org*

# Accessing Information II

- Mesquite mesh interface
  - Small, specific set of interfaces needed for mesh quality improvement
  - Four abstract C++ base classes: Mesh, VertexIterator, ElementIterator, and MeshDomain
  - Must be implemented by application
  - Data structure neutral through the use of handles
  - Fully compatible with ITAPS interface
- Geometric model
  - MESQUITE will not have its own geometry engine
  - Some simple call-back functions such as “move to owner” and “surface normal” by the application or ITAPS
  - ITAPS has developed a common interface for this functionality

# Instruction Queue

- Allows a sequence of operations on a mesh set object
  - One or more quality improvers
    - One must be designated as the Master quality improver
    - Several can be used as preconditioners
  - Quality assessment
    - Measured with respect to the master quality improver
  - Can be used on more than one mesh set
- Some pre-defined instruction queues available through high-level wrappers



# MESQUITE User Interface

- Multi-level API
  - Simple to use wrapper interface
    - Access Mesquite functionality in a minimal number of calls
    - Uses default algorithms, settings, stopping criterial
  - Low level interface for customization
    - User chooses the combination of metric, objective function, solver
    - User determines the instruction queue
- Assessment Tools
  - Diagnostics
  - Statistics
  - A priori and a posteriori quality assessment
- Users' Manual and Documentation

# Mesquite User Interface: Wrappers

```
#include Mesquite.h

void some_application_function{

  TSTT_Mesh tri_mesh, quad_mesh;

  Mesquite::initialize();

 // create a Mesh Set to hold the TSTT meshes
 Mesquite::MeshSet ms;
 ms->add_mesh(tri_mesh);
 ms->add_mesh(quad_mesh);

 // state the improvement objective
 ShapeQualityObjective shape_quality_objective;

 // improve the quality
 shape_quality_objective.improve_quality(ms);

 Mesquite::finalize();
}
```

# Mesquite User Interface: Customized

```
void some_application_function{

  TSTT_Mesh tri_mesh, quad_mesh;

  Mesquite::initialize();

 // create a Mesh Set to hold the TSTT meshes
 MeshSet ms;
 ms->add_mesh(tri_mesh);
 ms->add_mesh(quad_mesh);

 ShapeQualityMetric *condition_number_metric = ConditionNumberMetric::create_new();

 ObjectiveFunction *shape_objective_function = new L2_

 QualityAssessor *shape_quality_assessor = new QualityAsses
 shape_quality_assessor->compute_this(QualityAssessor::Minimum);

 NodeMover *opt_L2 = new NodeMover(shape_objective_function);
 opt_L2 = set_optimization_method(NodeMover::FeasibleNew
 opt_L2 = set_stopping_criterion(MAX_NODE_MOVEMENT,0.001);
```

Initialize

Declare a shape quality metric

Declare an objective function

Create a quality assessor

Create a L2 Node Mover

# Mesquite User Interface: Customized

```
TopologyModifier *tet_swapper = new TopologyModifier(
tet_swapper->set_optimization_method(TopologyModifier:
tet_swapper->set_stopping_criterion(MESH_PASSES,1);

UntangleQualityMetric *untangle_metric = FirstUntangleMetric::create_new();
ObjectiveFunction *untangle_objective_function = new LINF_TEMPLATE(untangle_metric);
NodeMover *opt_LINF = new NodeMover(untangle_
opt_LINF->set_optimization_method(NoveMover::Si
opt_LINF->set_stopping_criterion(OBJ_FCN_VAL,0

InstructionQueue q;
q.add_quality_assessor(shape_quality_assessor);
q.add_preconditioner(opt_LINF);
q.add_preconditioner(tet_swapper);
q.add_master_quality_improver(opt_L2);
q.add_quality_assessor(shape_quality_assessor);
q->execute_instruction_queue(ms);

Mesquite::shutdown();
}
```

Create a Topology Modifier

Create an untangler (metric, Objective function, NodeMover)

Create an instruction queue

Execute the instruction queue

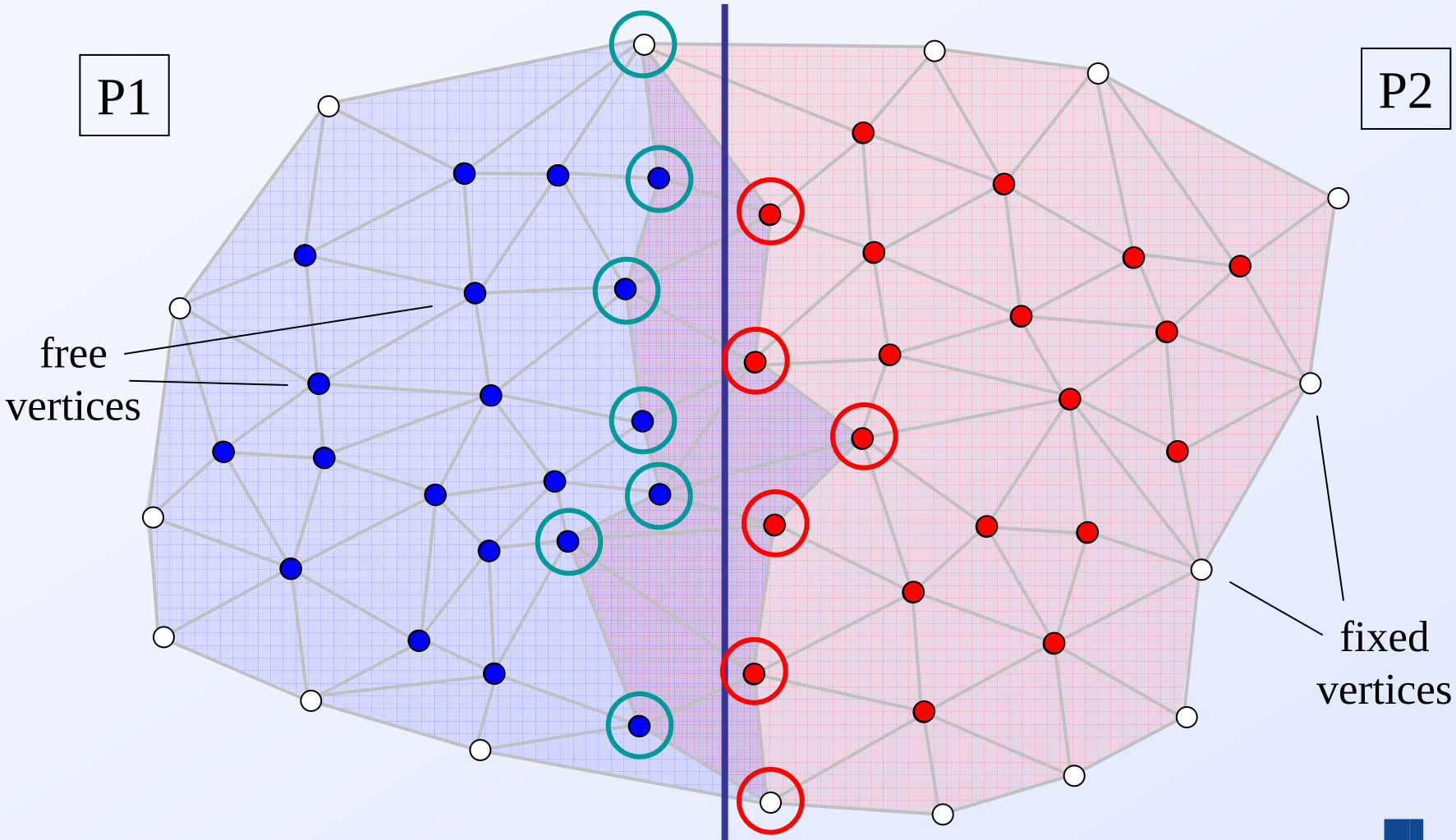
Finalize and shutdown

# User Customization

- Users can insert their own algorithms, objective functions, quality metrics without recompiling Mesquite
  - Inherit from VertexMover, ObjectiveFunction or QualityMetric
- User-defined metrics/objective functions can take advantage of existing MESQUITE algorithms
- Provides a platform for new research in mesh improvement algorithms
- Provides a platform for comparative studies

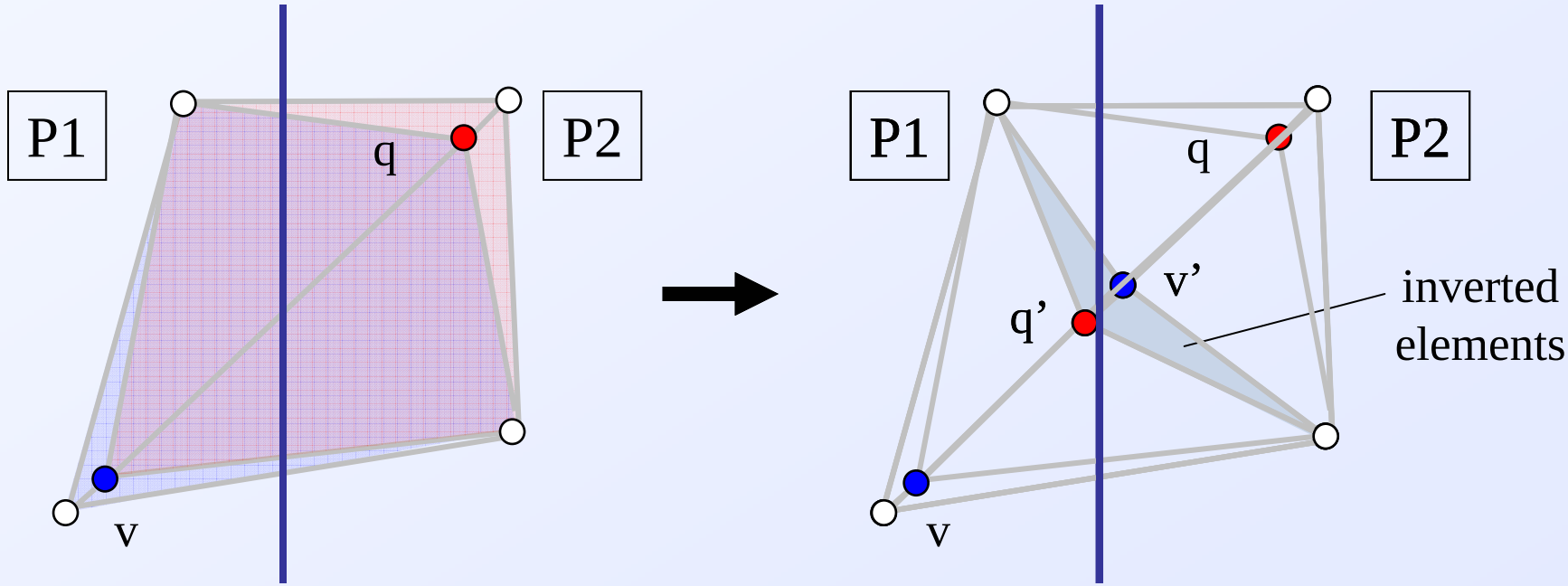
# Parallelizing Mesquite

# Parallel example with two processors (1)



# Problem for Partition Boundary Vertices

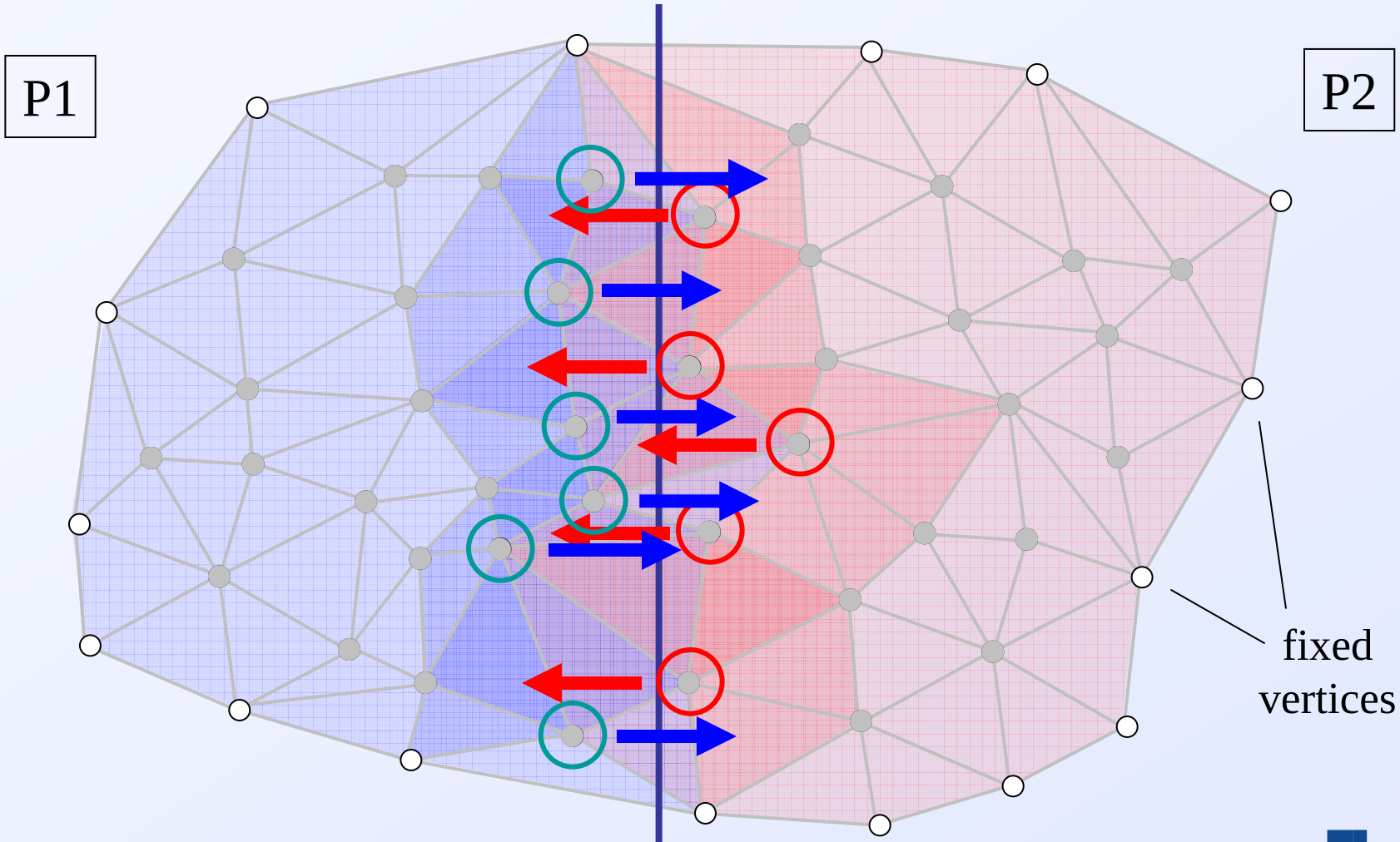
- unsynchronized, simultaneous smoothing of connected partition boundary (PB) vertices can destroy mesh validity



# Synchronize Processors for PB Vertices

- idea from [Freitag, Jones, and Plassmann, 1999]
  - treat partition boundary vertices separately
  - don't smooth at once ⇨ use *Independent Sets*
    - compute independent set
    - smooth
    - communicate
    - repeat until all partition boundary vertices smoothed
  - number of independent sets for bounded degree graph with $n=|V|$ is $O(log(n)/loglog(n))$ using a PRAM model with $O(n)$ processors

# Parallel example with two processors (2)



# Complete Parallel Smoothing Algorithm

**Determine** interior vertices $V_i$ and PB vertices $V_{PB}$
**Assign** unique random numbers to all $v \in V_{PB}$
**Compute** first independent set $I_0 \in V_{PB}$
**Smooth** all $v \in V_i \cup I_0$ (with local or global patch)
**Communicate** smoothed PB vertices for all $v \in I_0$
*k=1*
**while** (unsmoothed PB vertices remain) {
    **Compute** next independent set $I_k \in V_{PB}$
    **Smooth** all $v \in I_k$ (with local patch)
    **Communicate** smoothed PB vertices for all $v \in I_k$
    **Check** if unsmoothed PB vertices remain
}

# Using Mesquite in Parallel

- Requires use of ParallelMesh and ParallelHelper classes
- Application must provide
  - partitioned mesh with one layer of ghost nodes
  - communicator
  - global id's or (proc id, local id) pair
- ParallelHelper
  - Computes partition boundary
  - Computes independent sets on partition boundary
  - Communicates updated vertex information

# Mesquite ParallelHelper

```
Mesquite::ParallelMeshImp *mesh;
Mesquite::ParallelHelper *helper = new Mesquite::ParallelHelperImpl();

int *gid = init_mesquite_mesh(itaps_mesh,mesh);

helper->set_parallelmesh (mesh);
helper->set_communicator(communicator);
helper->set_communication_model(comm_model);
mesh->set_parallel_helper(helper);
```

Initialize ParallelHelper

Set the helper mesh and communicator

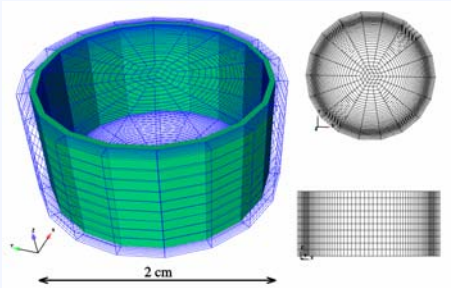
Set a communication model

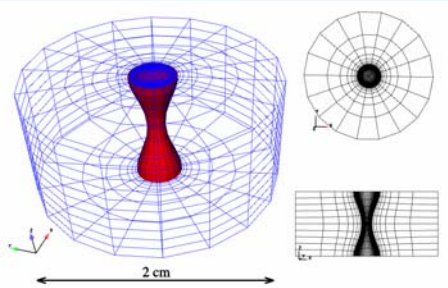
Set the helper on the mesh

Set QM, Solver, Term Criteria, and Instruction Queue as before

Options for blocking, non-blocking and avoiding all-reduce communications in the “comm_model”

# Our Mesh Quality Improvement Work has Impacted Many DOE Applications




*Application*: Plasma implosion using ALE methods

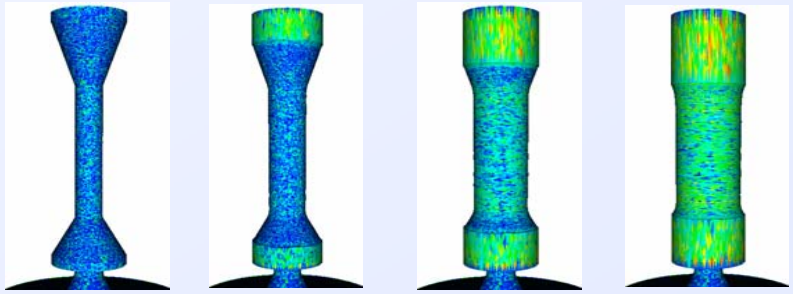
*Challenge*: Maintain good mesh quality and biasing during deformation of plasma.

*Impact*: Prior to use of Mesquite, this calculation could not be performed by Alegra due to ineffective mesh rezoning algorithm.

*Application*: Burn of rocket propellants in a time-deforming domain

*Challenge*: Maintain good tetrahedral element shape quality as domain deforms

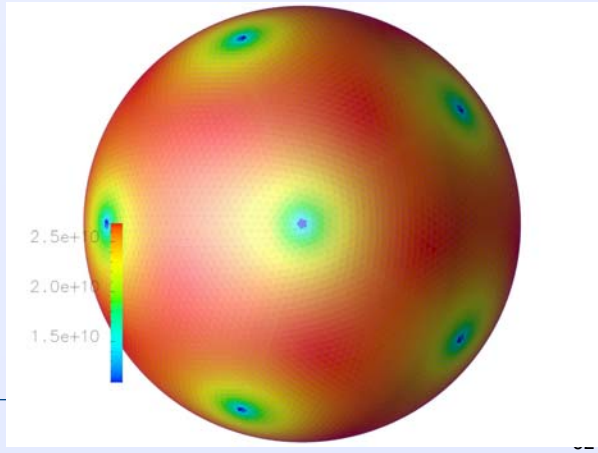
*Impact*: Condition number smoother (through ShapeImprovementWrapper) enabled many burn simulations at CSAR/UIUC.



*Application*: Climate: studies of finite volume discretization methods

*Challenge*: Create many different high quality geodesic meshes on a sphere

*Impact*: Used manydifferent mesh optimization methods to enable researcher Todd Ringler (CSU) to compare accuracy of discretization method on different meshes for climate calculations. CVT

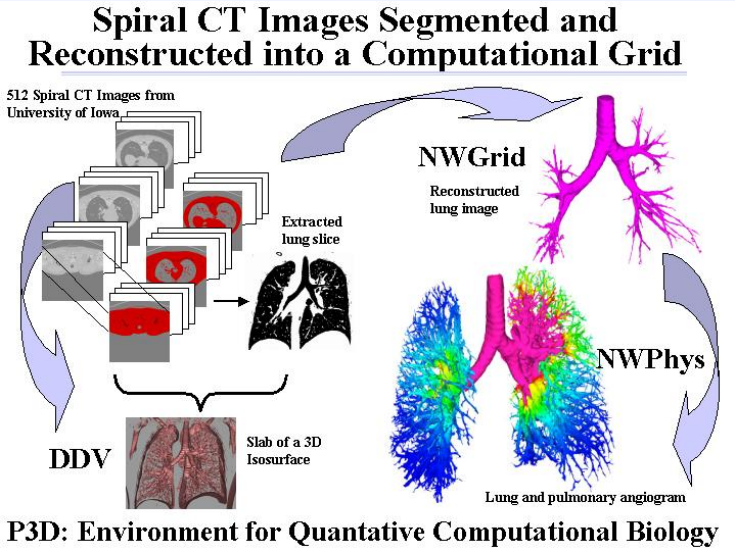


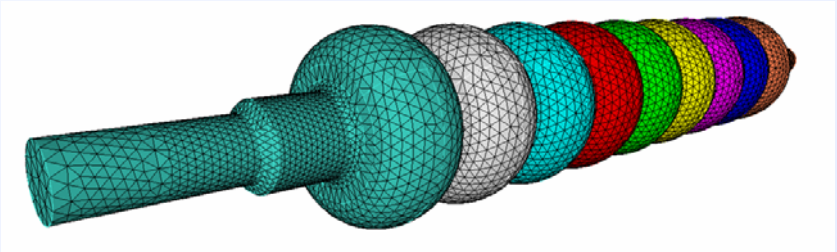
# Our Mesh Quality Improvement Work has Impacted Many DOE Applications

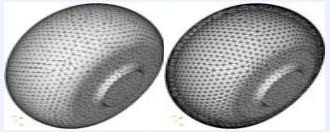
*Application*: Computational Biology: CT images converted into a computational grid

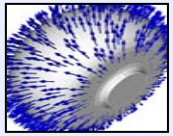
*Challenge*: Highly complex geometries

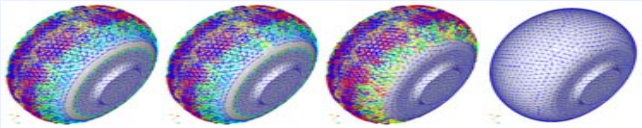
*Impact*: Mesquite enables PNNL to create good quality meshes for computational biology.









*Application*: Shape optimization for accelerator cavities to minimize losses

*Challenge*: Rapidly and smoothly update the mesh to conform to trial geometries

*Impact*: Used the deforming mesh metric to prototype geometry & mesh update model for potential use in SLAC accelerator design studies.

# Some Future Directions

- Post processing approach will be applied to a wider array of applications
  - design optimization,
  - deforming mesh smoothing
  - ALE applications
- Node movement schemes will be used in adaptive, a posteriori settings
- More sophisticated solvers will be developed
  - automatic mesh boundary improvement
  - improve efficiency
  - take advantage of the structure of particular problems

# Mesquite Software Availability

- User Documentation

  http://www.cs.sandia.gov/optimization/knupp/Mesquite.html

- Beta release software with limited support available

  http://www.cs.sandia.gov/~web9200/9200_download.html

- ITAPS Interfaces

  http://www.itaps-scidac.gov

# Contact Information

Lori Freitag Diachin
Center for Applied Scientific Computing
L-561
Lawrence Livermore National Laboratory
Livermore, CA 94551
diachin2@llnl.gov

Patrick Knupp
Optimization and Uncertainty Estimation Department
MS 1111
Sandia National Laboratories
Albuquerque, NM 87185
pknupp@sandia.gov

# Acknowledgments


- The Mesquite team (past and present)
  - Michael Brewer (SNL)
  - Lori Diachin (LLNL)
  - Patrick Knupp (SNL)
  - Jason Kraftcheck (Wisconsin)
  - Thomas Leurent (ANL)
  - Darryl Melander (SNL)
  - Martin Isenburg (LLNL)
- Funding

SciDAC
Scientific Discovery through Advanced Computing