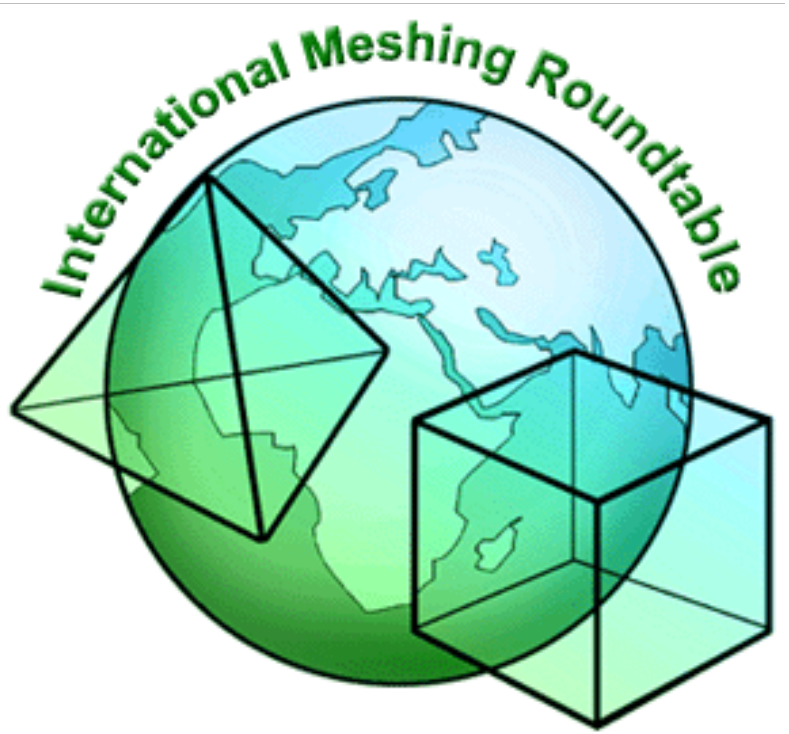
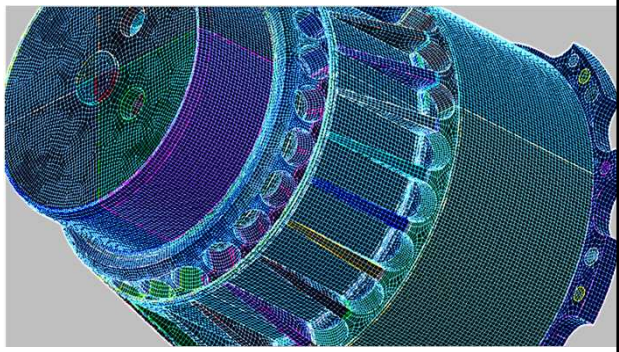
SAND2016-9194C

Exceptional service in the national interest

Sandia National Laboratories





An Introduction to Automatic Mesh Generation Algorithms

Short Course, September 26, 2016
Washington, DC

Steven Owen
Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY | NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Agenda

Sandia National Laboratories

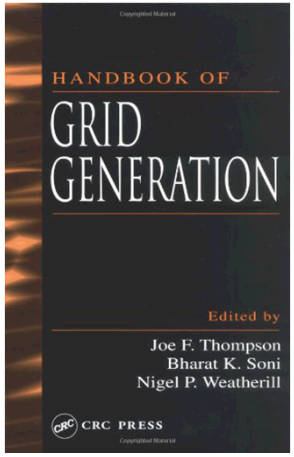
Part I 8:00-9:30AM

- The Simulation Process
- Geometry Basics
- Mesh Representations
- Mesh Generation Methods
- Tet/Tri Meshing Methods
- Surface Meshing Basics
- Smoothing

Part II 10:00-11:30AM

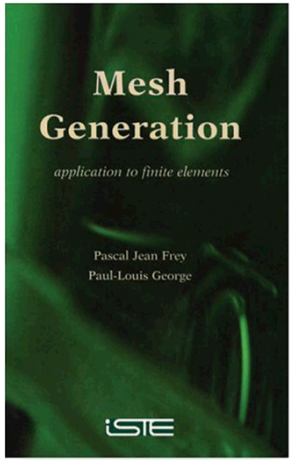
- Tet vs. Hex Meshing
- Structured vs. Unstructured
- Structured Hex Methods
- Unstructured Hex Methods
- Hex Dual Representations
- Overlay Grid
- Automatic Block Decomposition
- Hybrid Methods

Classical References



J. F. Thompson, B. K. Soni, and N. P. Weatherill, eds., **Handbook of Grid Generation**, CRC Press, 1998.
Block-structured grids:

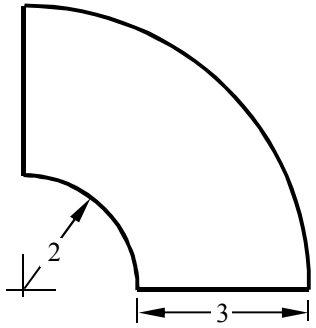
- Transfinite-interpolation, Elliptic and hyperbolic PDE systems, Harmonic mappings, ...
- Unstructured grids: Quadtrees and Octrees methods,
- Advancing-front methods, Delaunay-Voronoi methods,
- Anisotropic Grid generation, ...



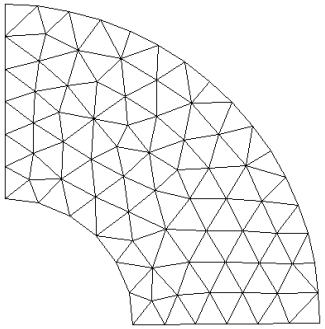
P. J. Frey and P.-L. George, **Mesh Generation - Application to Finite Elements**, Hermes Science Publishing, Oxford, UK, 1st ed., 2000, 2nd ed. 2008.

- A comprehensive survey of Tetrahedral mesh generation methods: Quadtree-octree methods, Advancing-front methods, Delaunay-based methods, ...

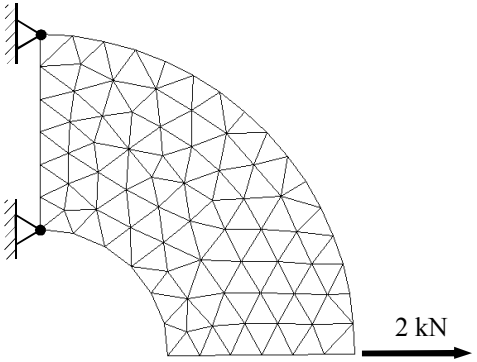
Simulation Process



1. Build CAD Model

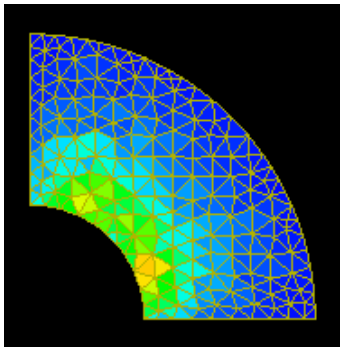


2. Mesh



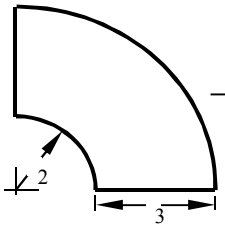
3. Apply Loads and Boundary Conditions

4. Computational Analysis

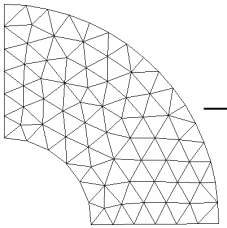


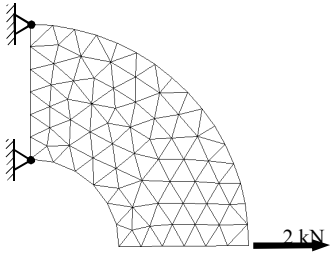
5. Visualization

Adaptive Simulation Process



User supplies meshing parameters





1. Build CAD Model

2. Mesh

3. Apply Loads and Boundary Conditions

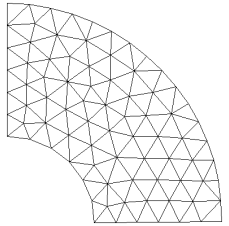
Adaptivity Loop

4. Computational Analysis

5. Error Estimation

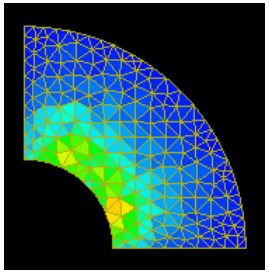
Error > ε

Analysis Code supplies meshing parameters



6. Remesh/Refine/Improve

Error < ε

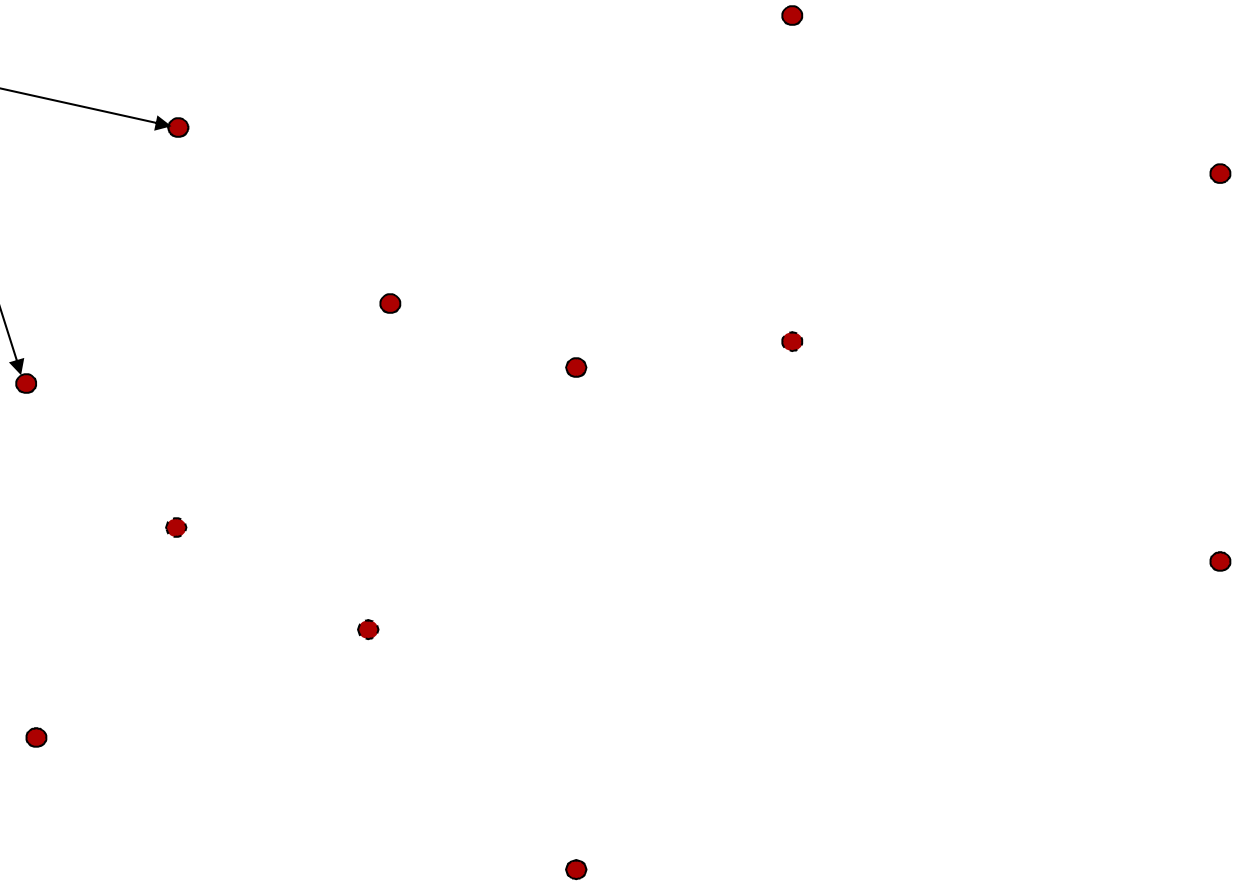


7. Visualization

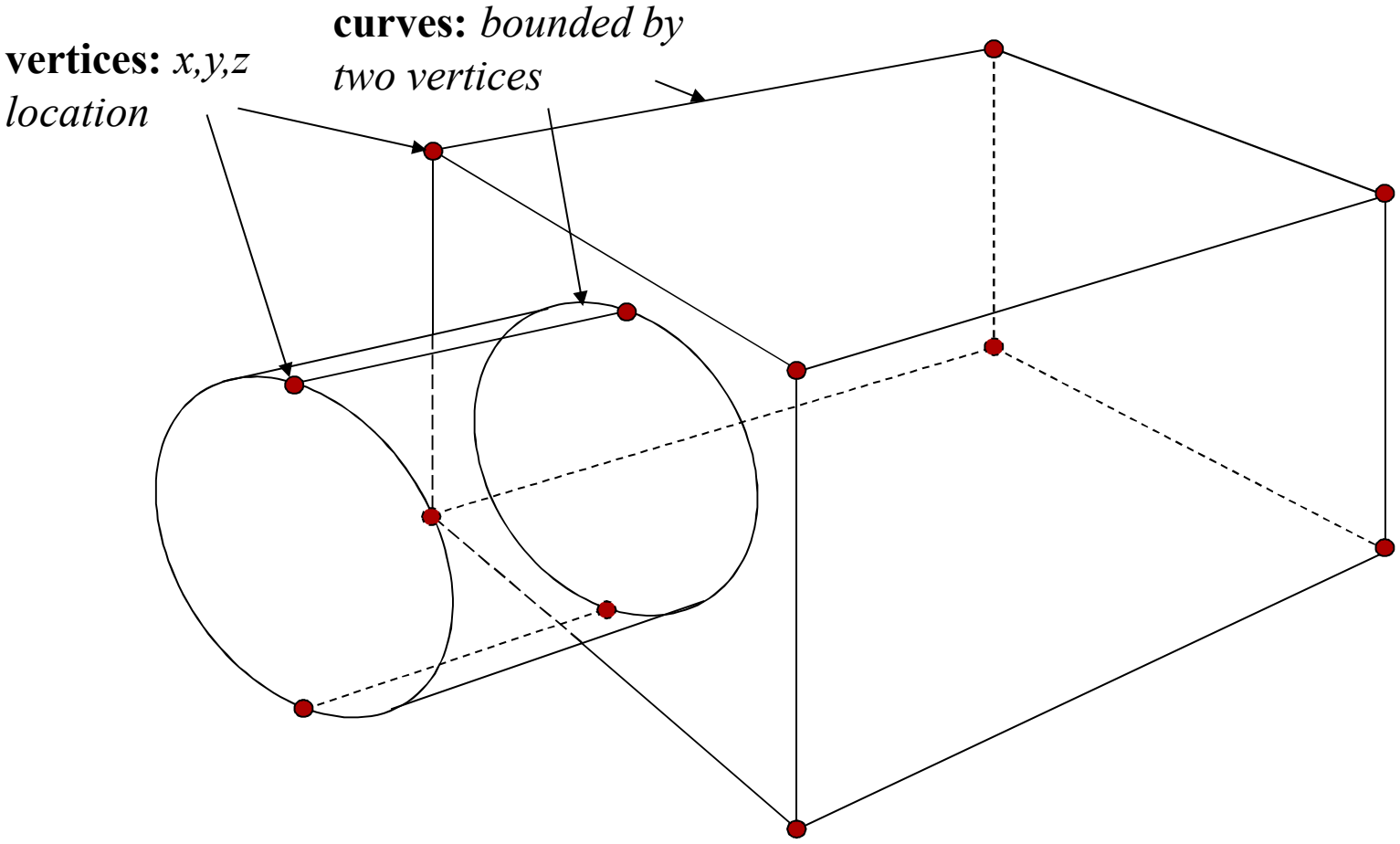
Geometry

Sandia National Laboratories

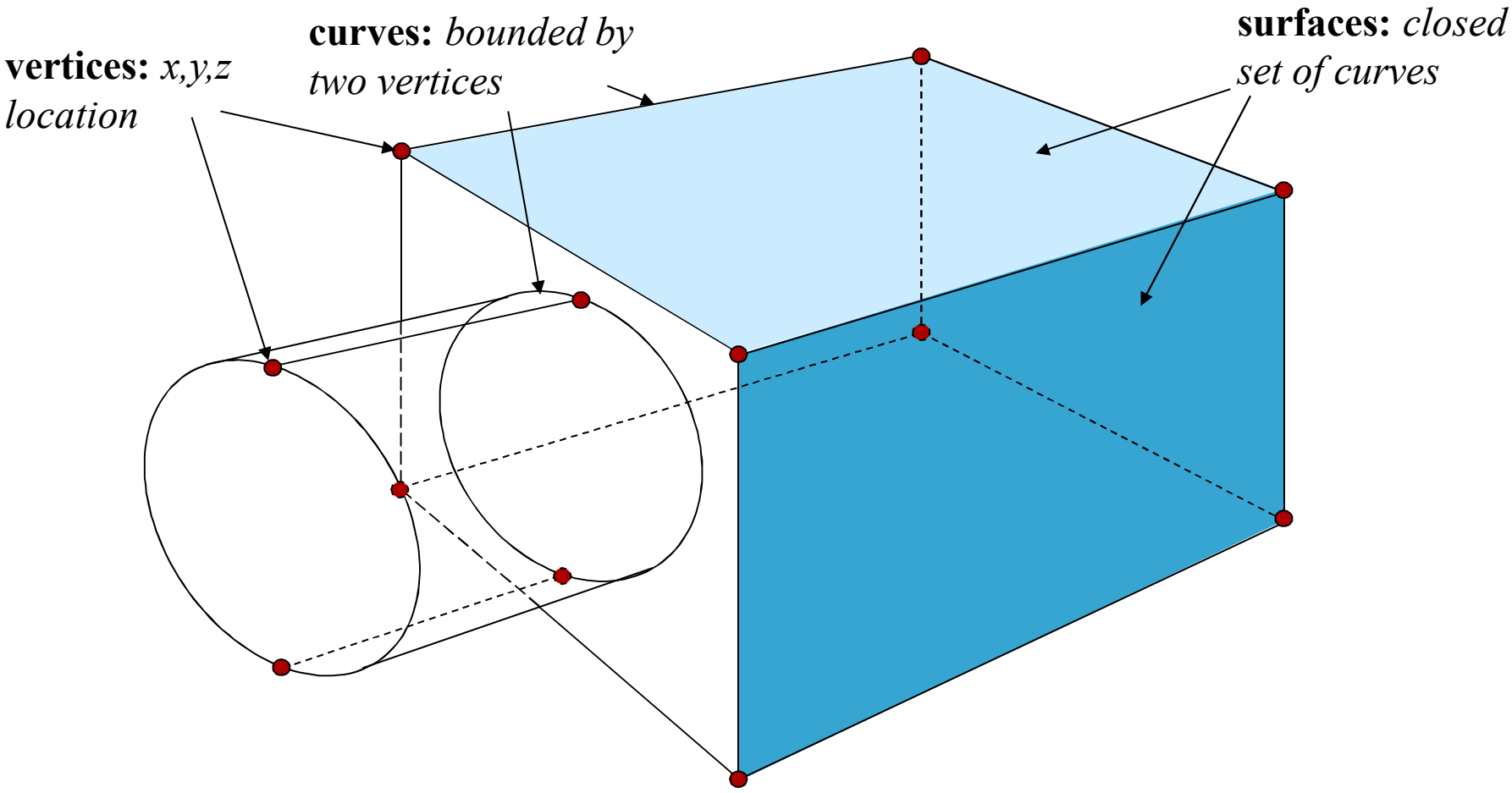
vertices: *x,y,z*
location



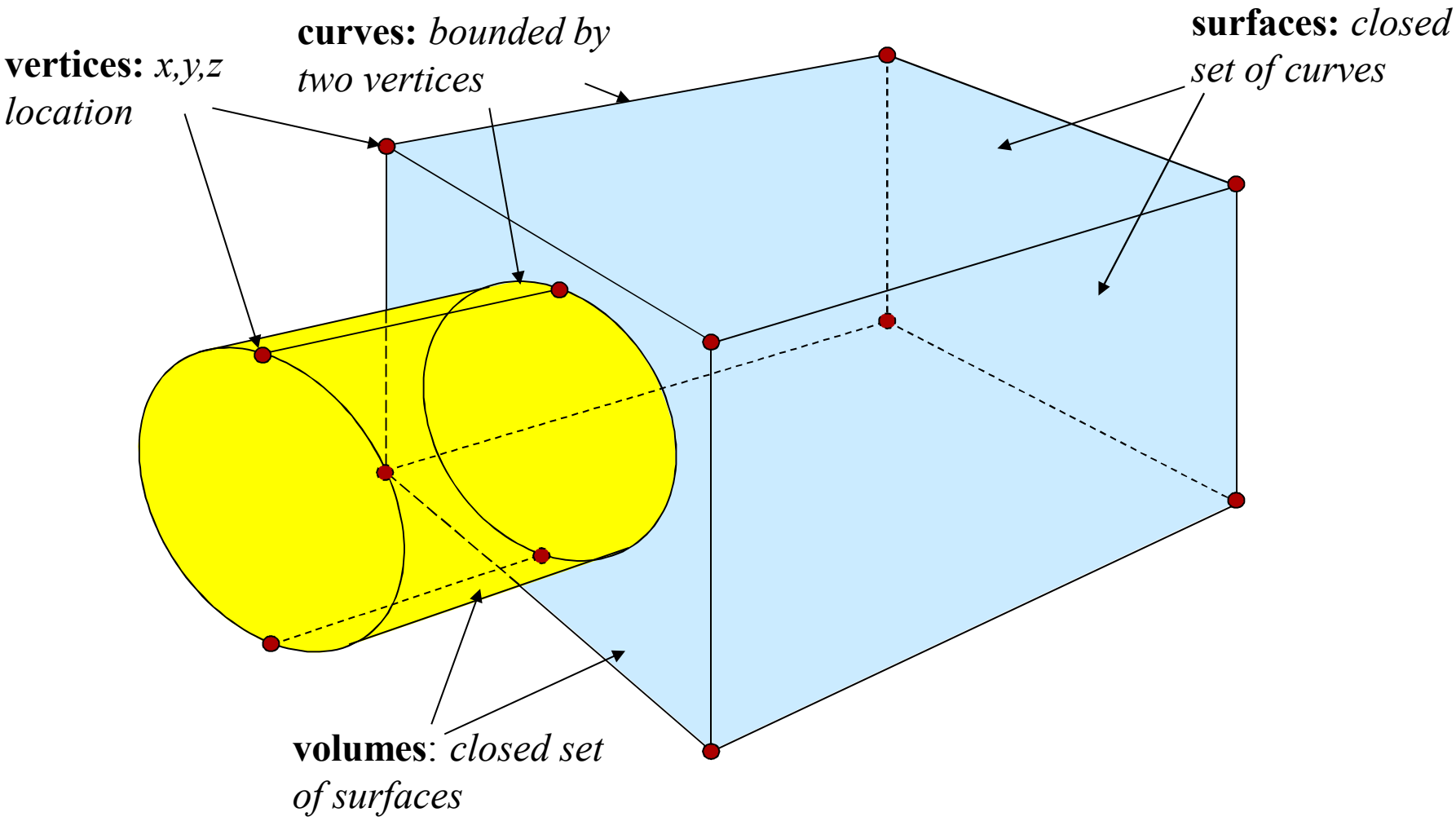
Geometry



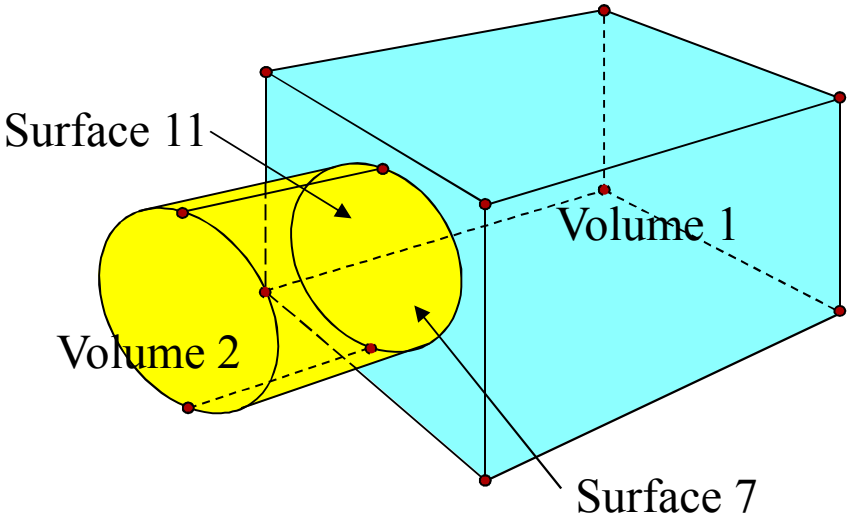
Geometry


vertices: *x,y,z location*
curves: *bounded by two vertices*
surfaces: *closed set of curves*

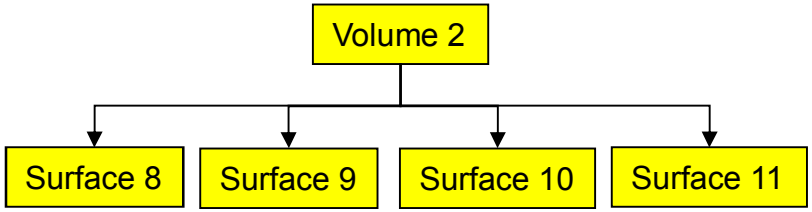
Geometry

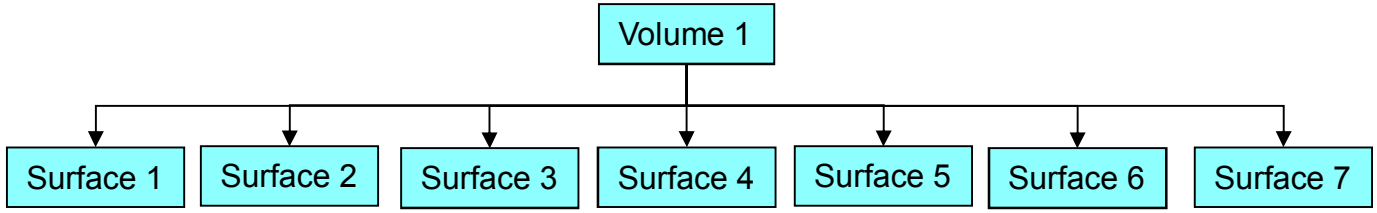

vertices: *x,y,z location*
curves: *bounded by two vertices*
surfaces: *closed set of curves*
volumes: *closed set of surfaces*

Geometry

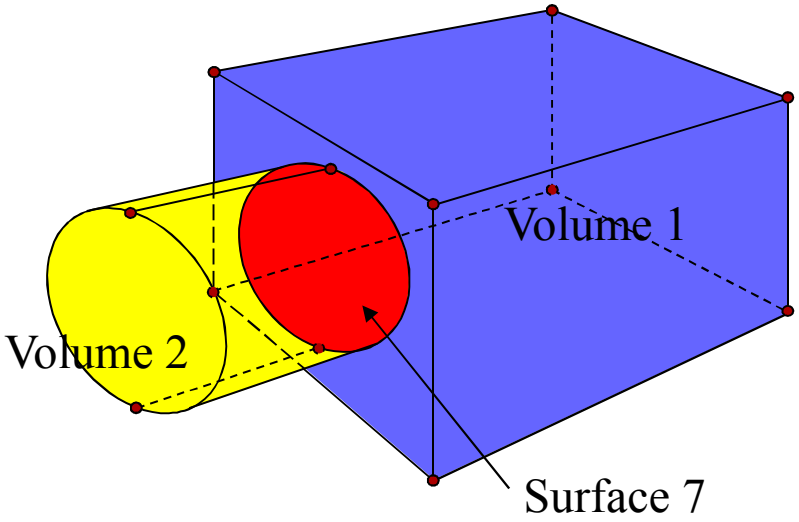


Manifold Geometry:
Each volume maintains its own set of unique surfaces

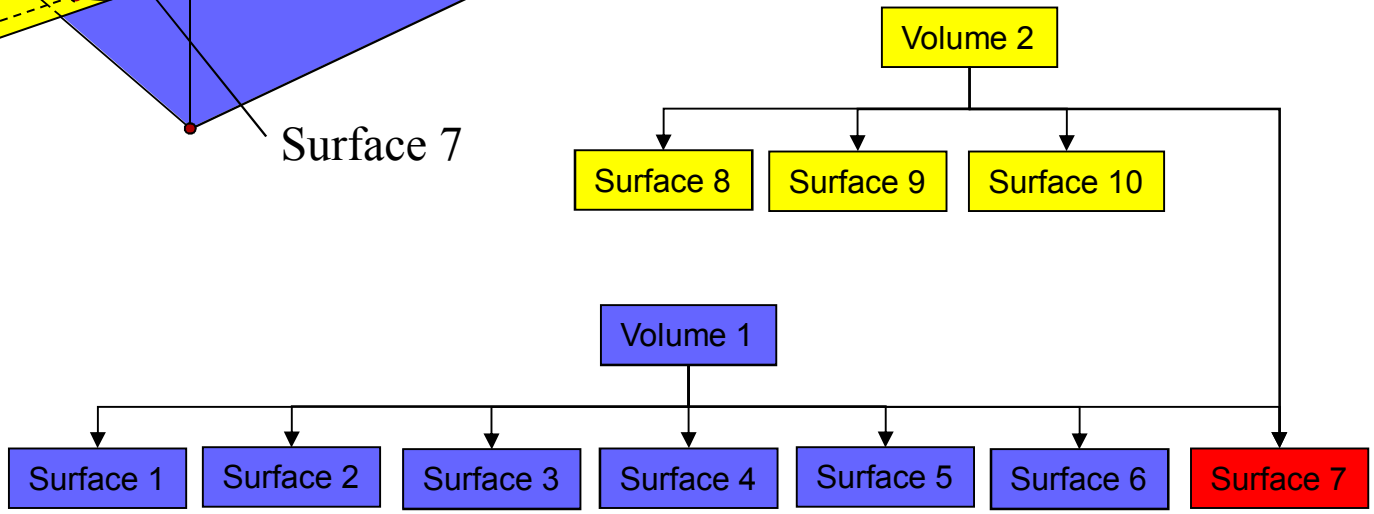




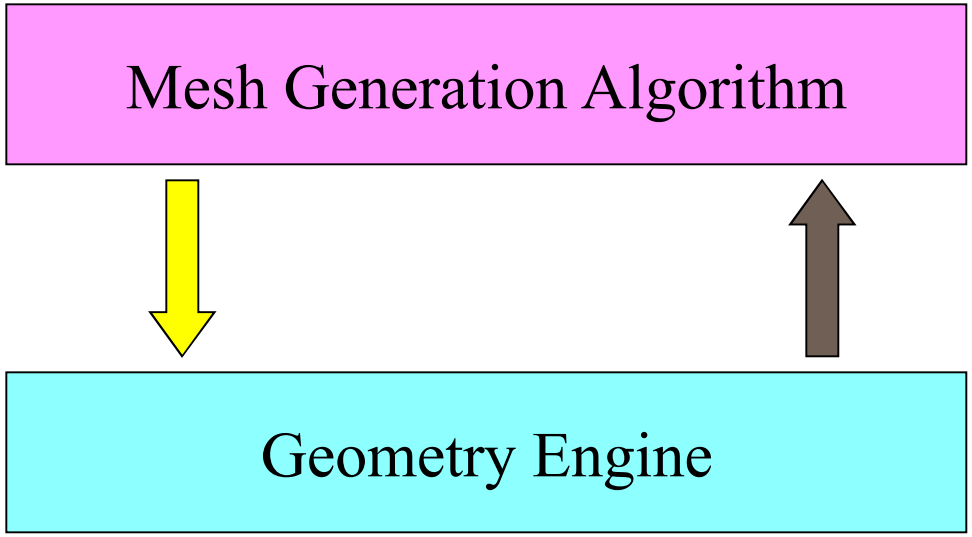
Geometry



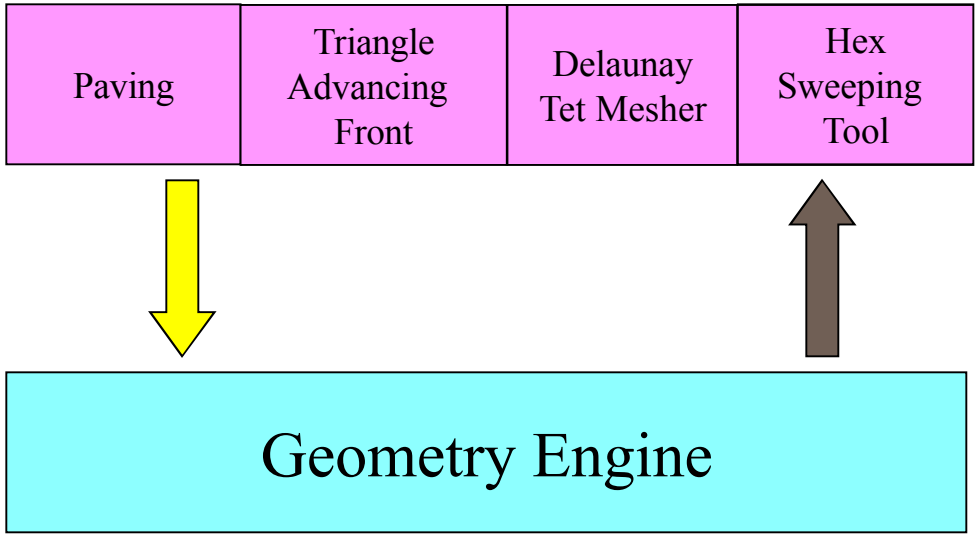
Non-Manifold Geometry: *Volumes share matching surfaces*



Geometry



Geometry


Paving
Triangle Advancing Front
Delaunay Tet Mesher
Hex Sweeping Tool
Geometry Engine

Geometry

Sandia National Laboratories

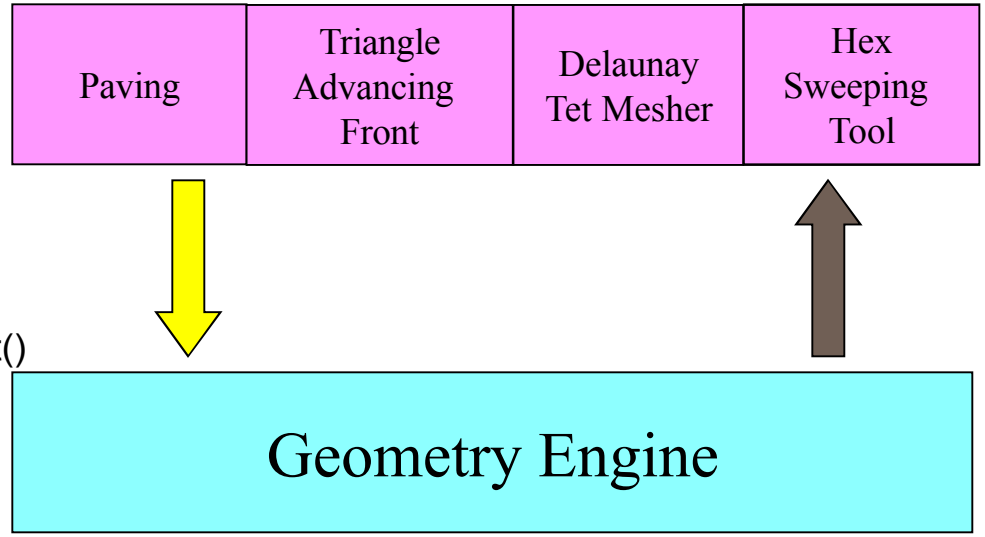
Geometry Query

Curve::project_to()

Surface::project_to()

Surface::normal_at()

Surface::curvature_at()



Topology Query

Surface::curves()

Curve::vertices()

Volume::surfaces()

Vertex::curves()

Geometry

Sandia National Laboratories

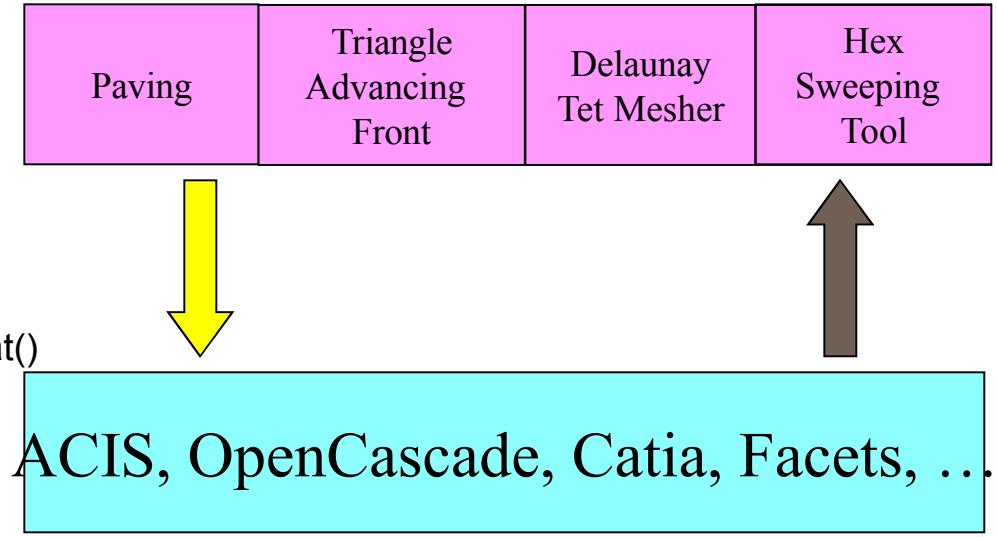
Geometry Query

Curve::project_to()

Surface::project_to()

Surface::normal_at()

Surface::curvature_at()



Topology Query

Surface::curves()

Curve::vertices()

Volume::surfaces()

Vertex::curves()

Geometry

Sandia National Laboratories

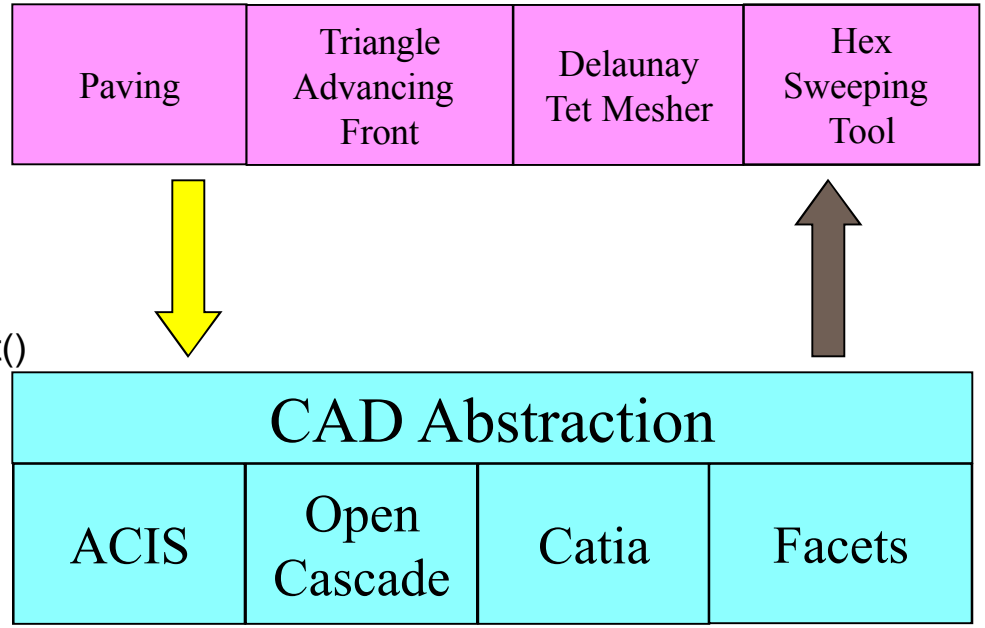
Geometry Query

Curve::project_to()

Surface::project_to()

Surface::normal_at()

Surface::curvature_at()



Topology Query

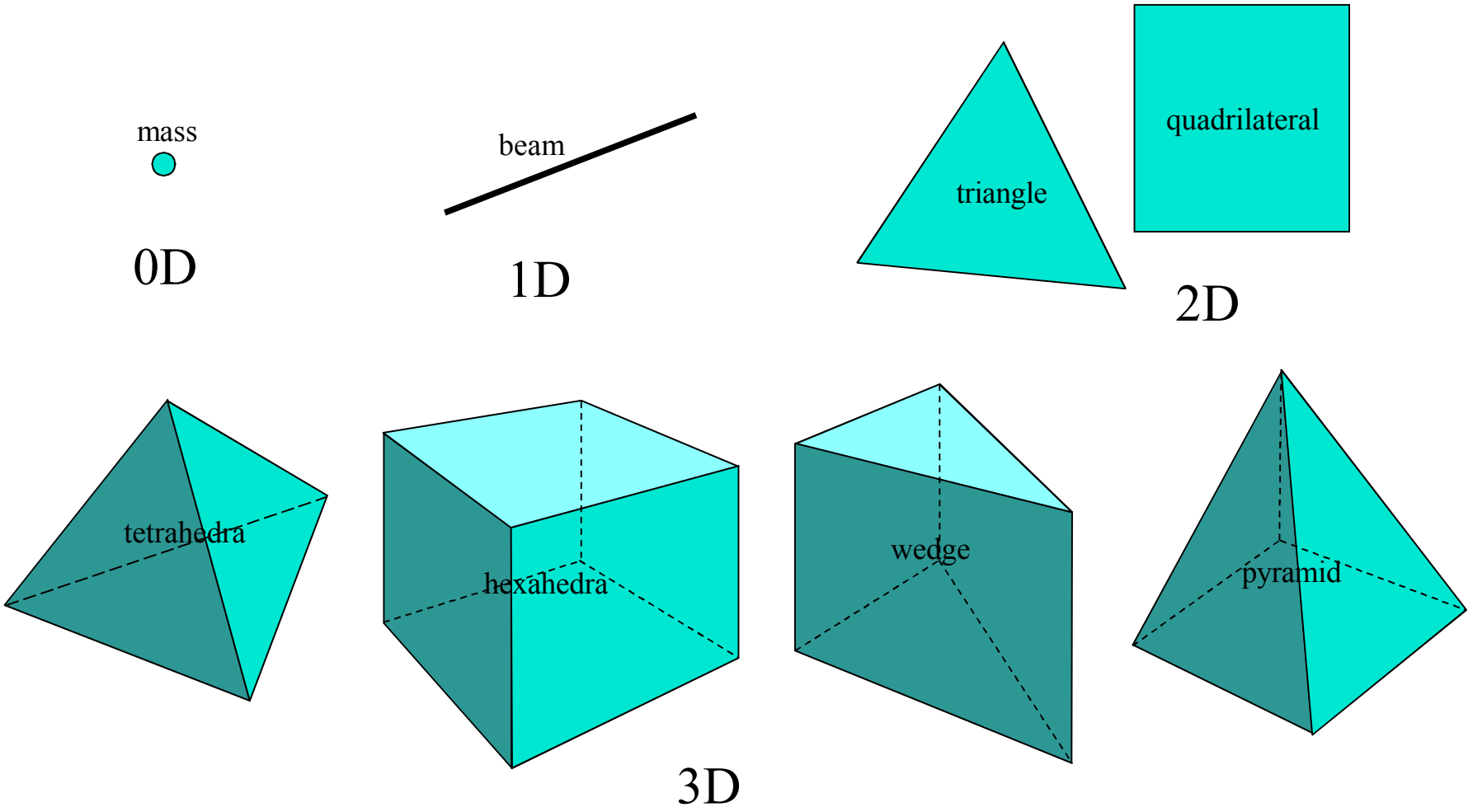
Surface::curves()

Curve::vertices()

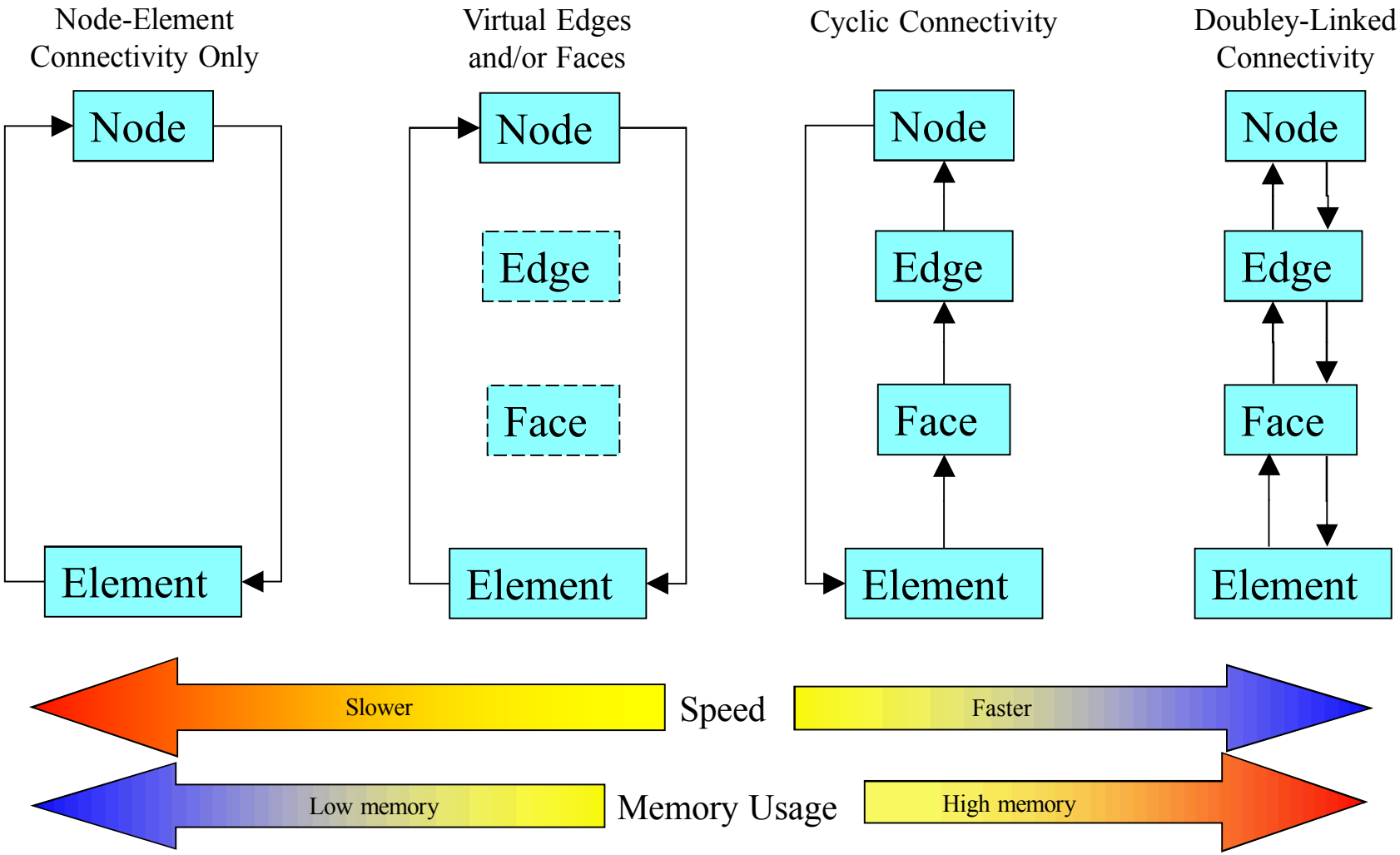
Volume::surfaces()

Vertex::curves()

Mesh Representation



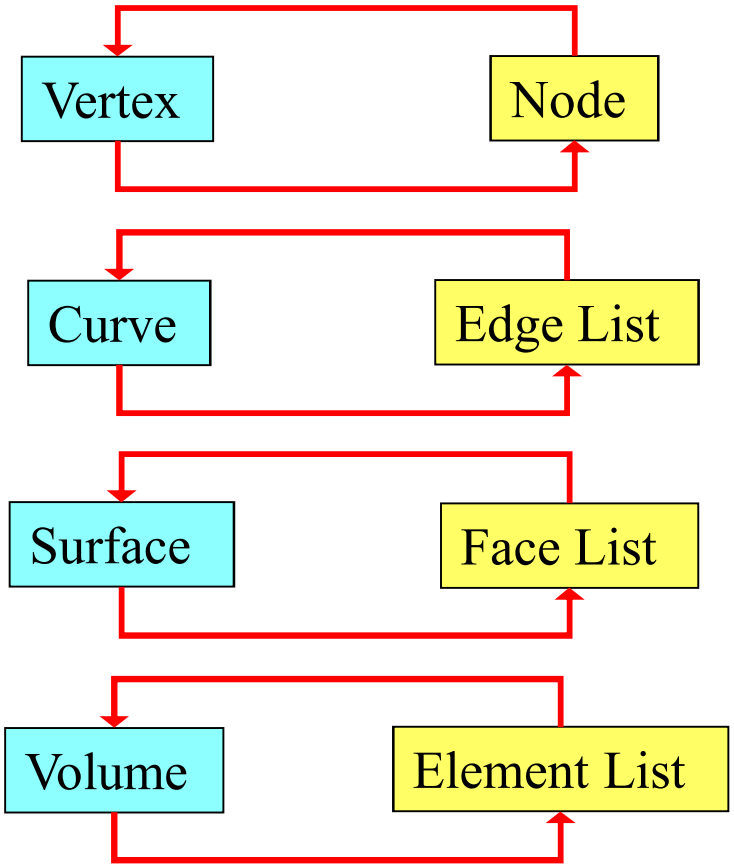
Mesh Representation



Mesh Representation

Sandia National Laboratories

Geometry Associativity

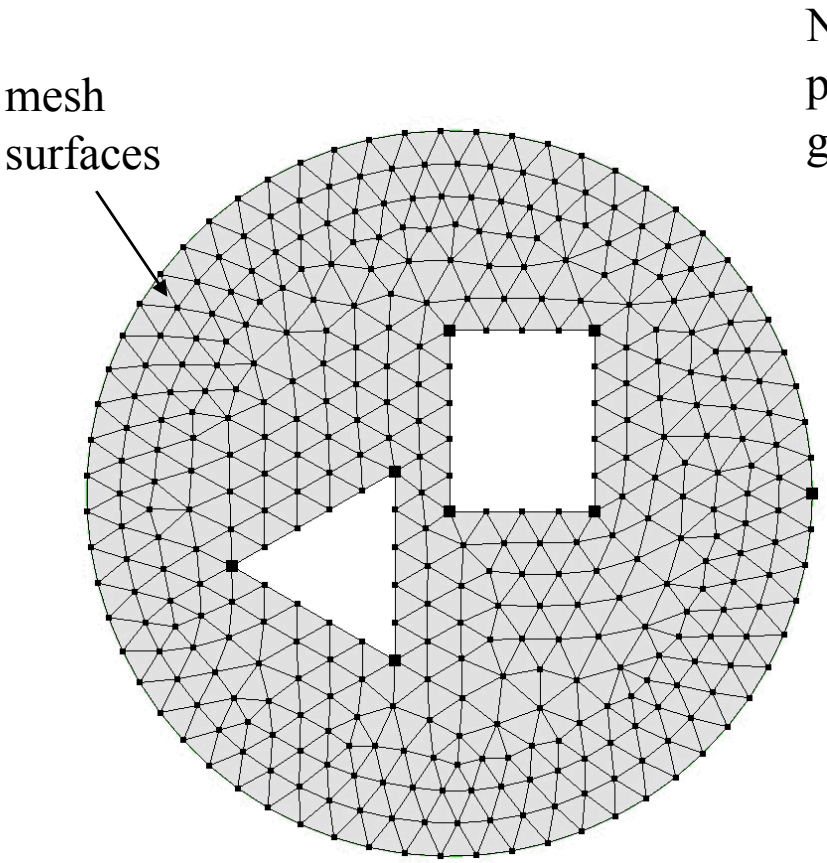


Boundary Conditions applied to geometric entities.

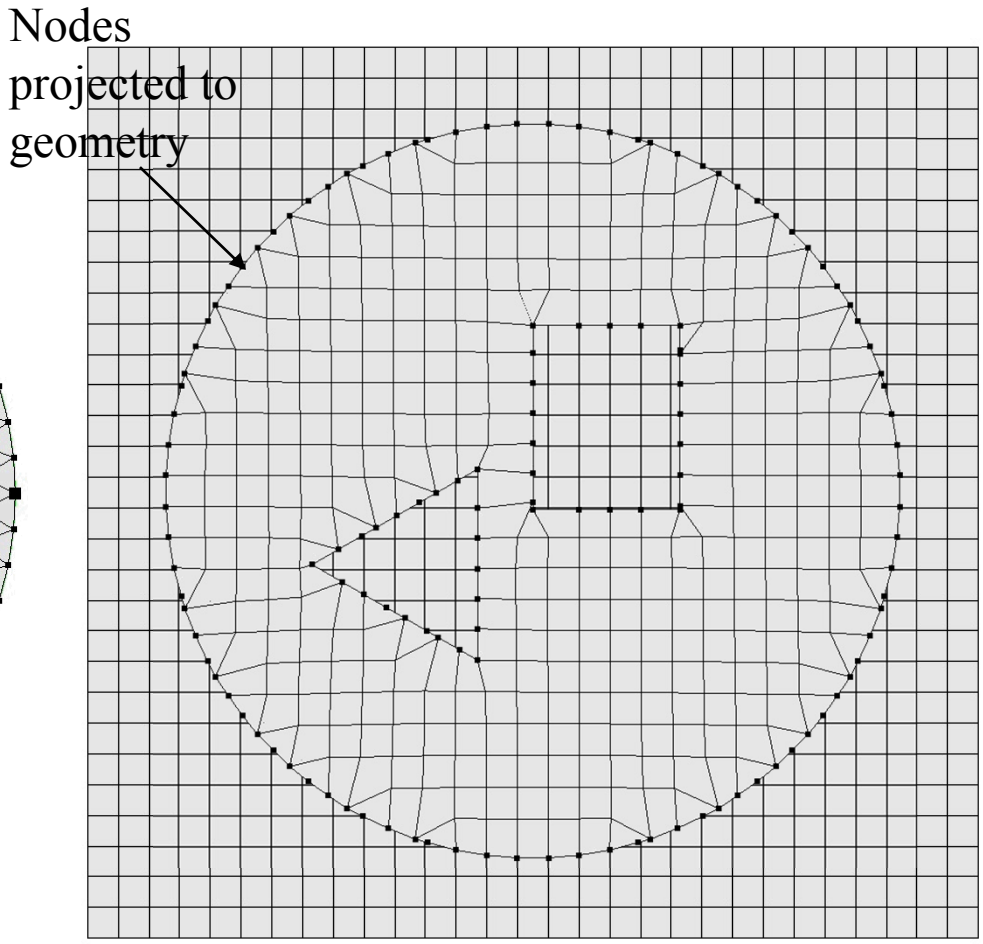
Delete/Modify mesh defined on geometric entity

Operations on mesh entities require conformity to geometry

Mesh Generation Methods



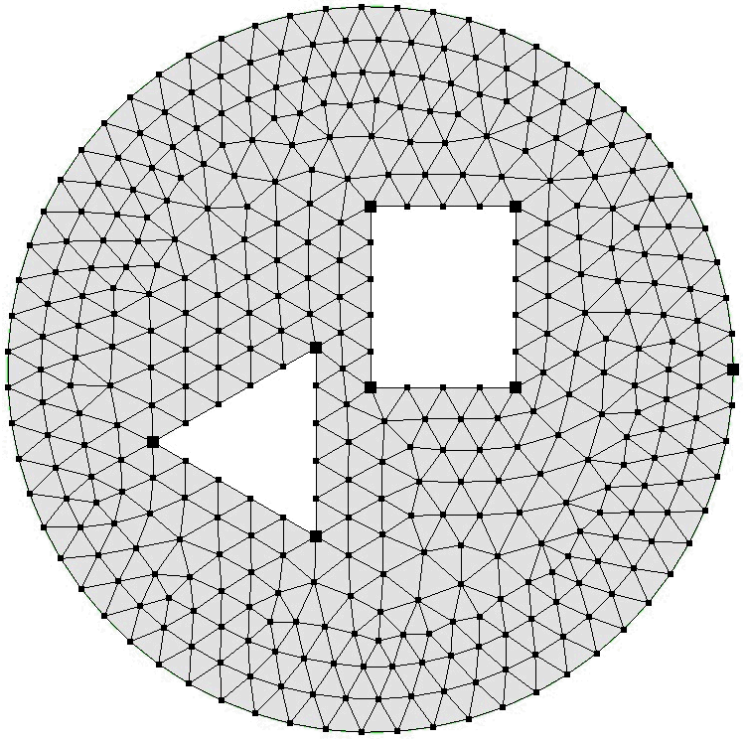
Geometry First



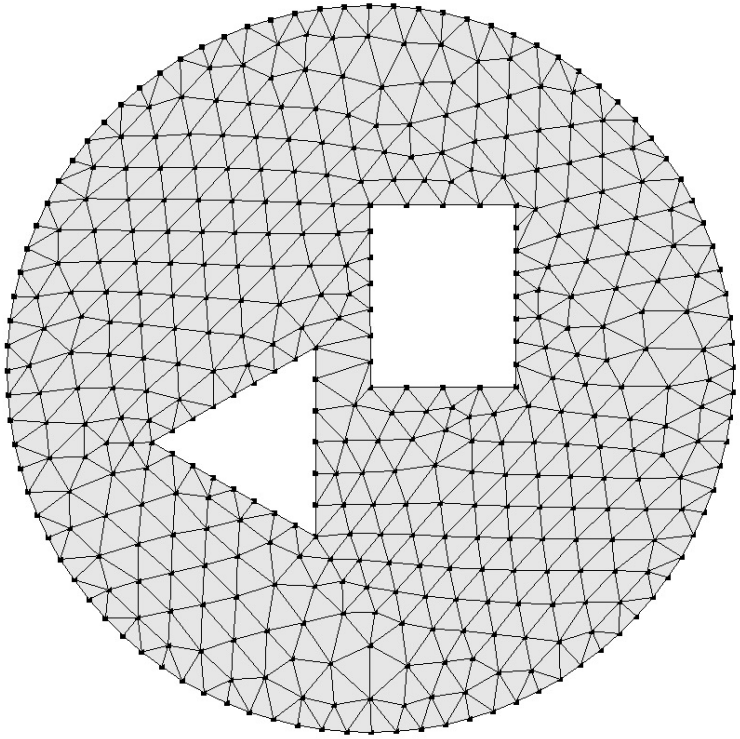
Mesh First

Mesh Generation Methods

Sandia National Laboratories



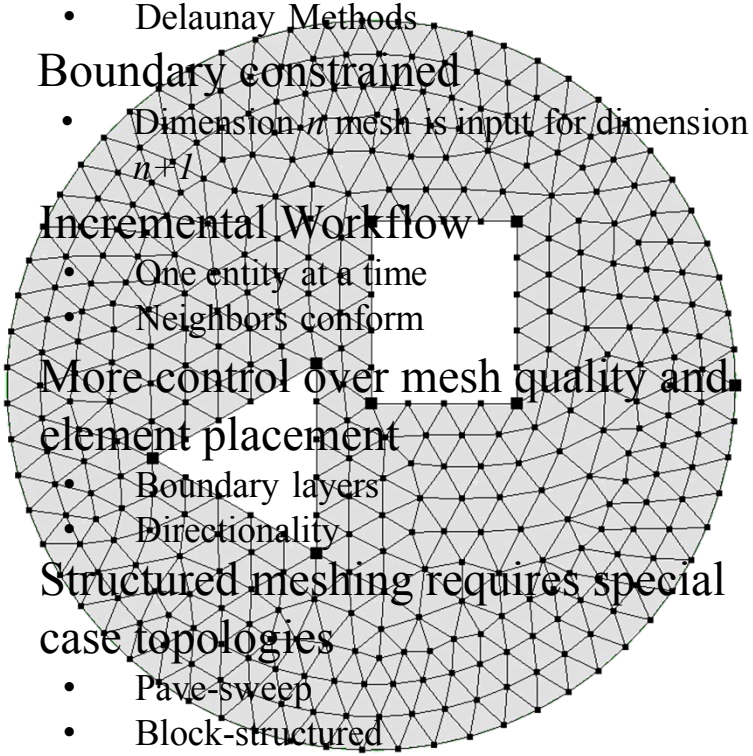
Geometry First



Mesh First

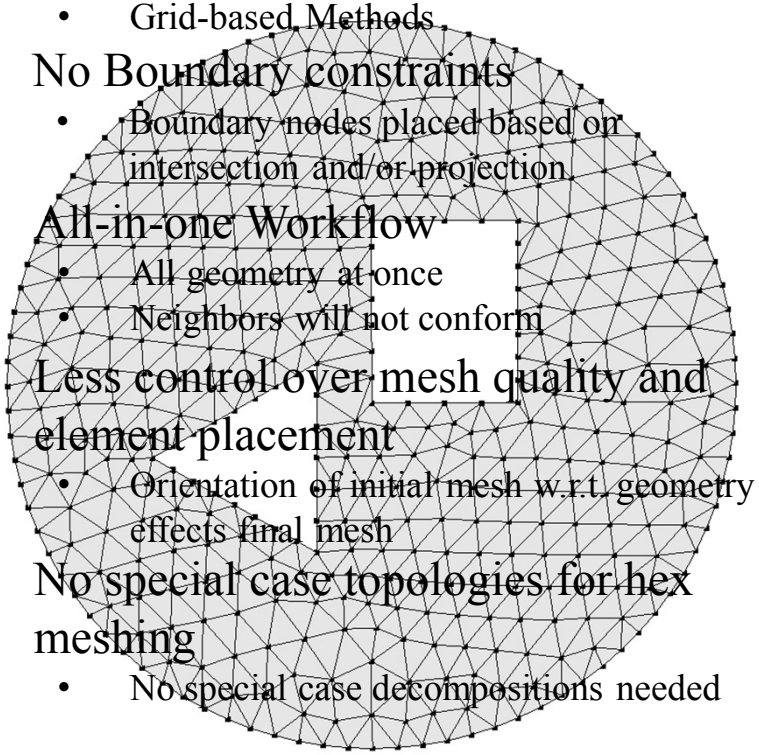
Mesh Generation Methods

- AKA. Bottom-up Meshing
  - Advancing Front
  - Delaunay Methods
- Boundary constrained
  - Dimension *n* mesh is input for dimension *n+1*
- Incremental Workflow
  - One entity at a time
  - Neighbors conform
- More control over mesh quality and element placement
  - Boundary layers
  - Directionality
- Structured meshing requires special case topologies
  - Pave-sweep
  - Block-structured
  - Manual/Automated decomposition required



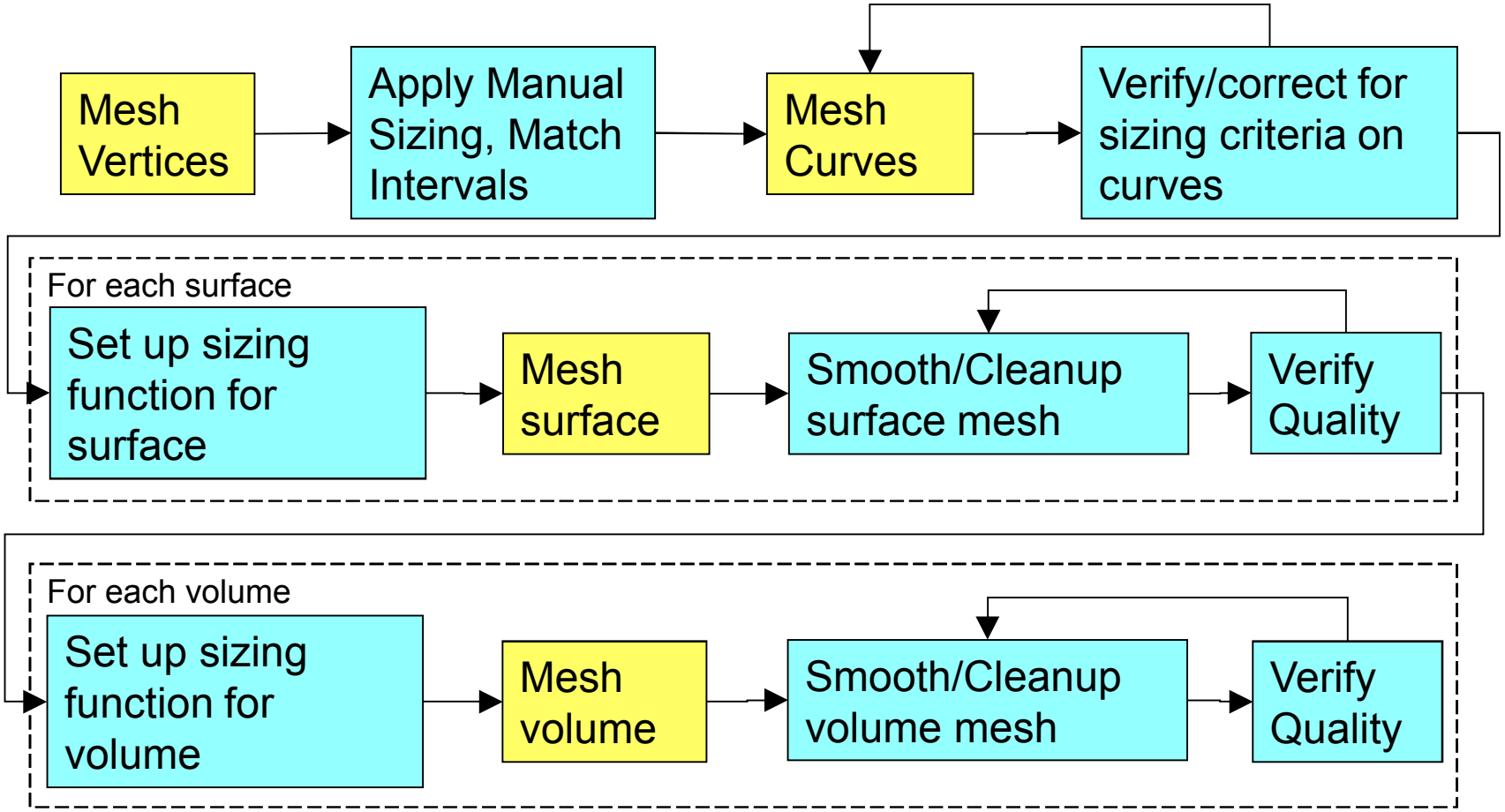
Geometry First

- AKA. Overlay Grid
  - Octree Methods
  - Grid-based Methods
- No Boundary constraints
  - Boundary nodes placed based on intersection and/or projection
- All-in-one Workflow
  - All geometry at once
  - Neighbors will not conform
- Less control over mesh quality and element placement
  - Orientation of initial mesh w.r.t. geometry effects final mesh
- No special case topologies for hex meshing
  - No special case decompositions needed



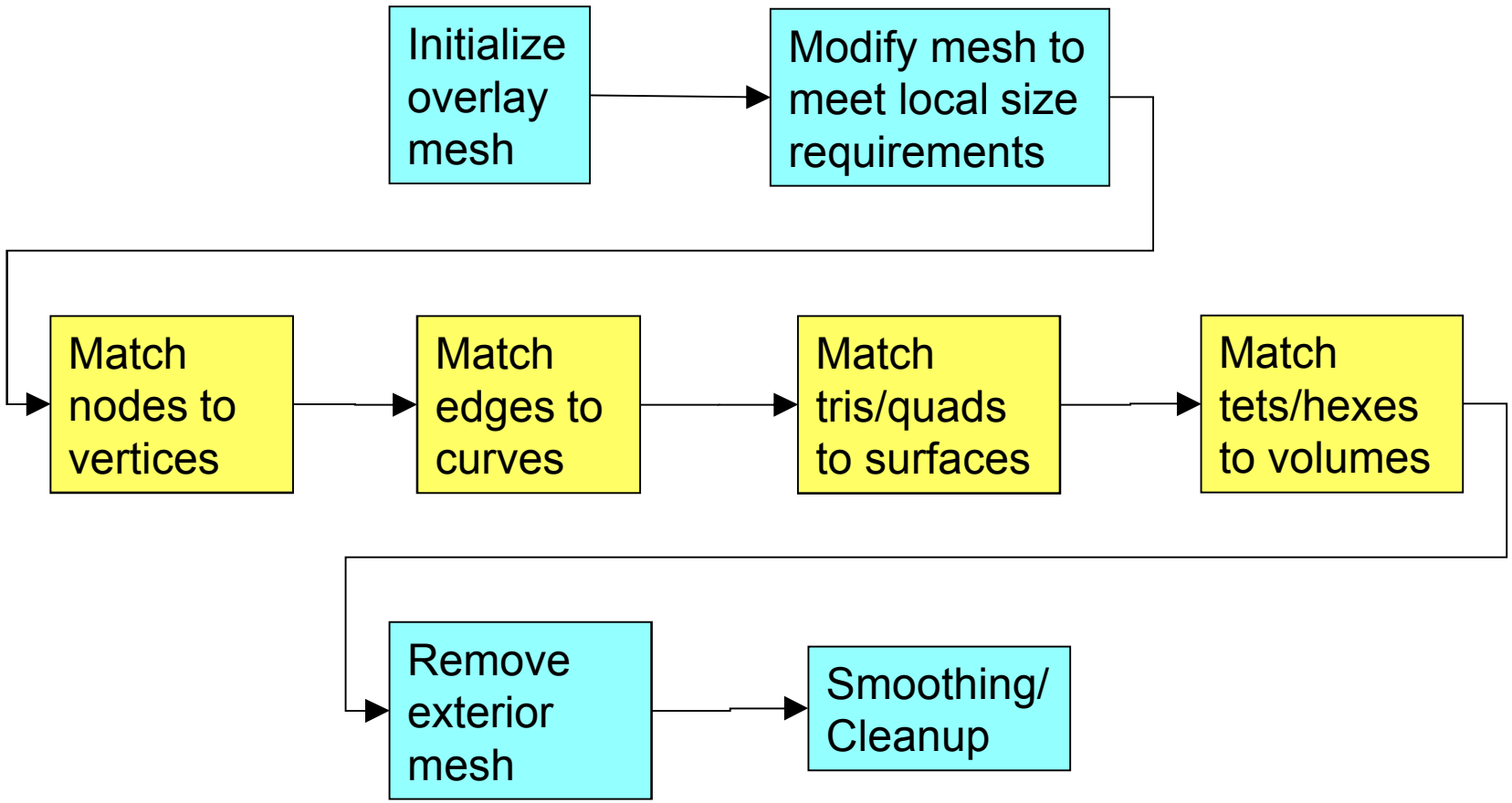
Mesh First

Mesh Generation Process

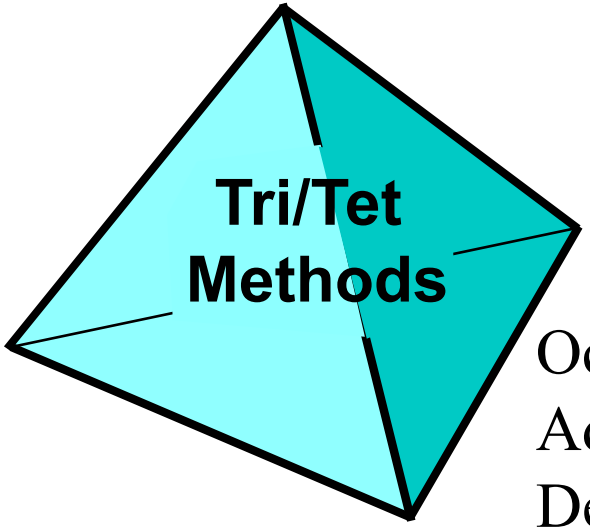


Geometry-First Mesh Generation

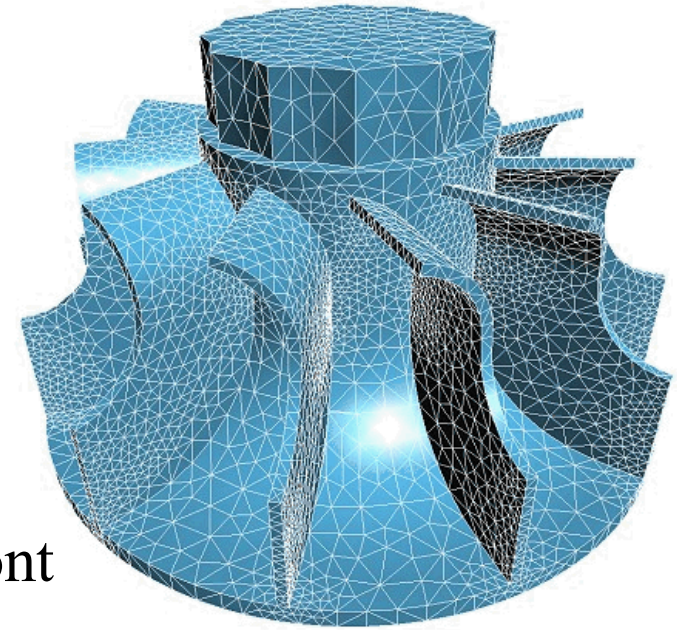
Mesh Generation Process



Mesh-First Mesh Generation

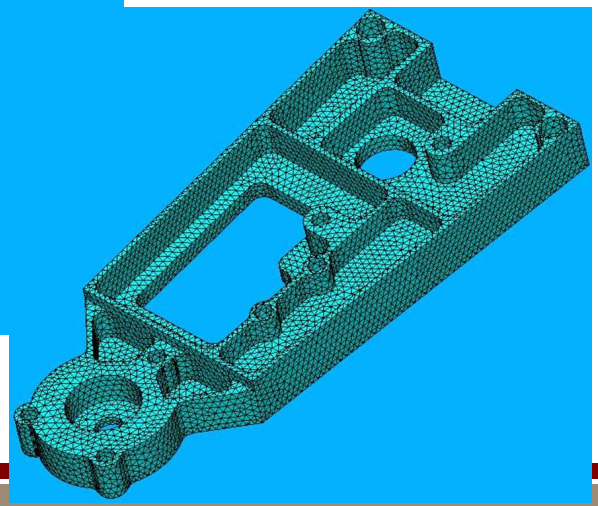


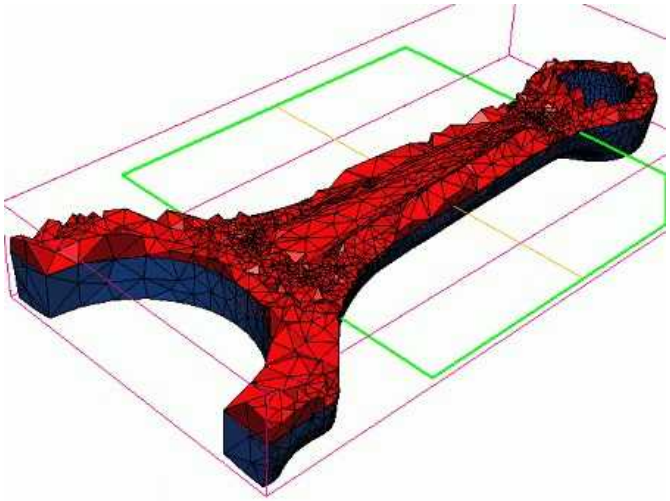
Octree
Advancing Front
Delaunay



http://www.simulog.fr/mesh/gener2.htm

http://www.ansys.com





Tet Meshing Software

Sandia National Laboratories

A Representative Sample

Commercial

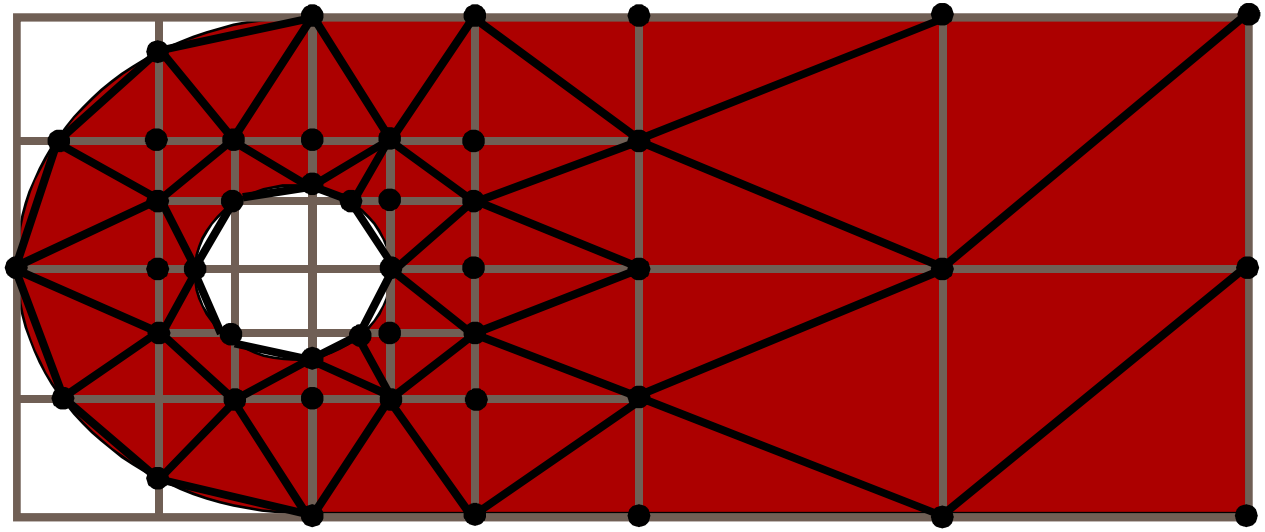
- Tetmesh-GHS3D, INRIA, Rocquencourt, Distene France.
- MeshSim, SCOREC, RPI, Simmetrix Inc. USA.
- SolidMesh, AFLR mesh generator, SimCenter, Mississippi State Uni., (Altair Hypermesh)

Open Source

- Netgen, TU Vienna.
- Gmsh, Uni. Liege & Uni. Catholique de Louvain.
- GRUMMP, University of British Columbia.
- Pyramid, UC Berkeley.
- CGALmesh, INRIA, Sophia-Antipolis.
- TetGen, Weierstrass Institute, Berlin.

Octree/Quadtree

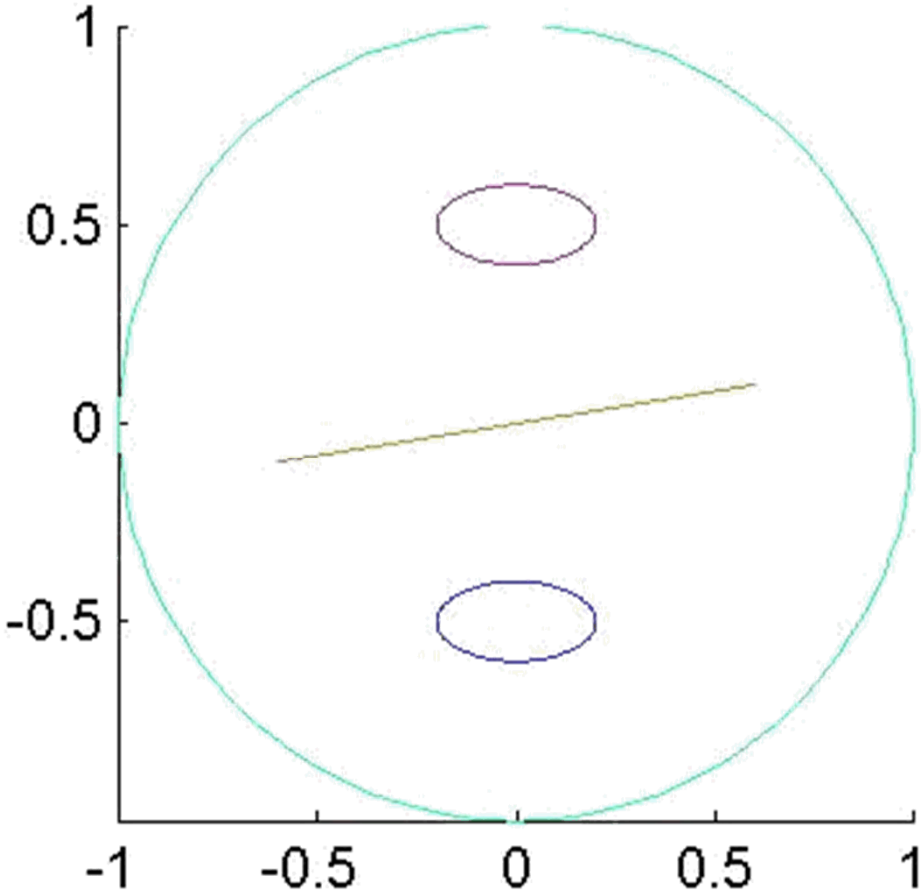
Sandia National Laboratories



- Define intial bounding box (*root* of quadtree)
- Recursively break into 4 *leaves* per *root* to resolve geometry
- Find intersections of leaves with geometry boundary
- Mesh each *leaf* using corners, side nodes and intersections with geometry
- Delete Outside
- (Yerry and Shephard, 84), (Shepherd and Georges, 91)

Octree/Quadtree

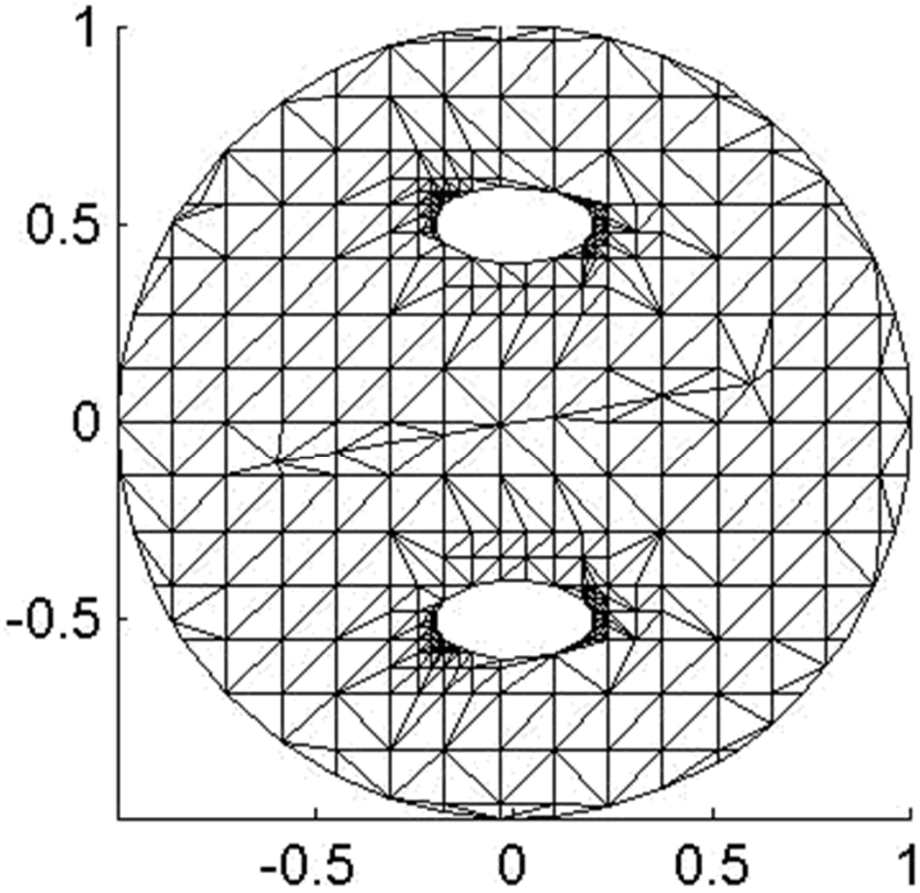
Sandia National Laboratories



QMG,
Cornell University

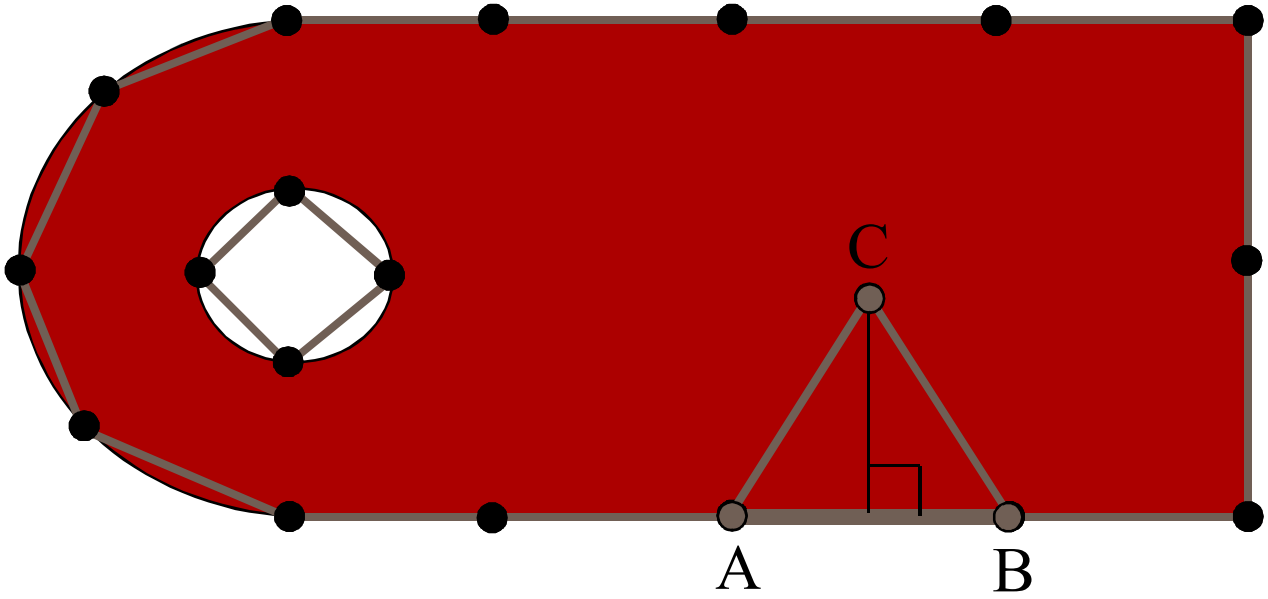
Octree/Quadtree

Sandia National Laboratories



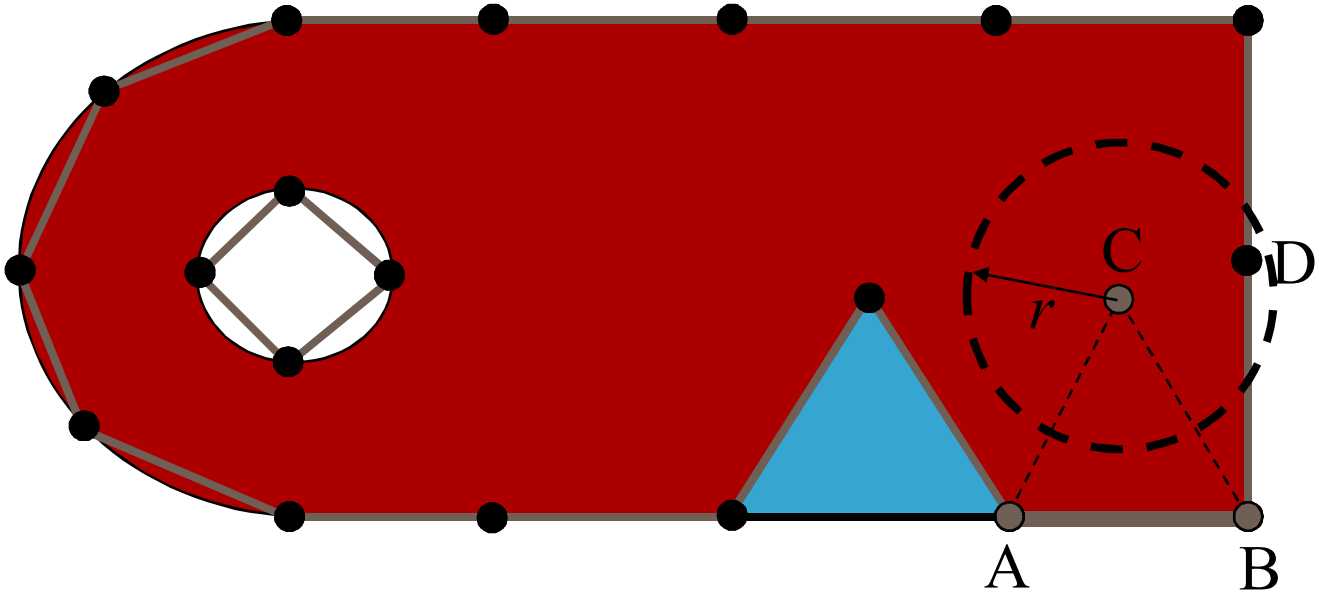
QMG,
Cornell University

Advancing Front



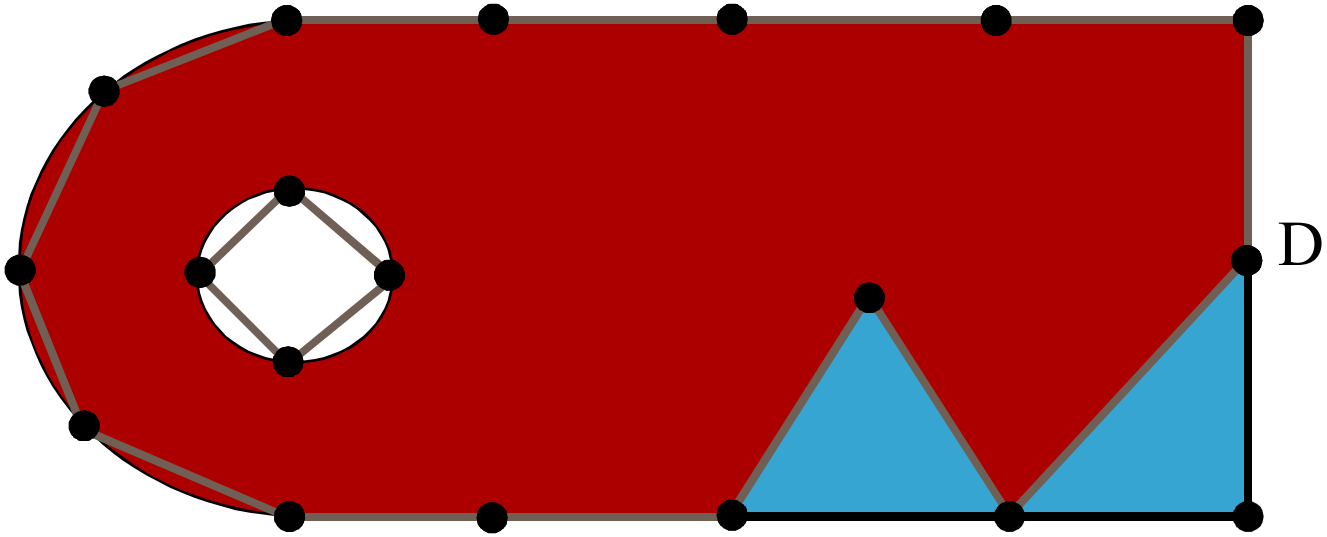
- Begin with boundary mesh - define as initial *front*
- For each edge (face) on front, locate ideal node C based on front AB

Advancing Front



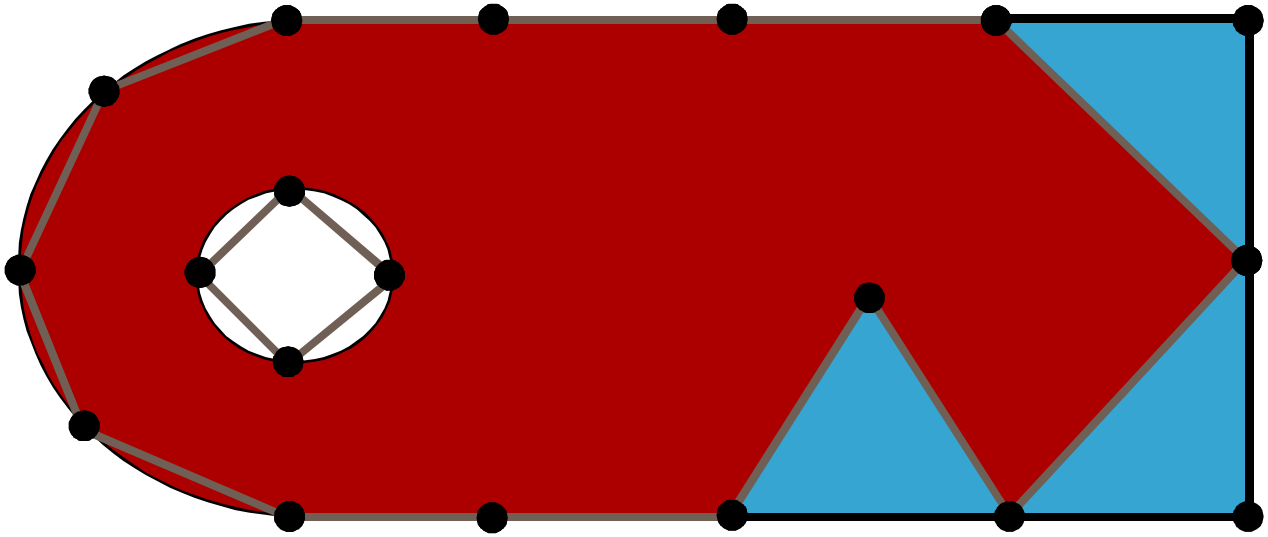
•Determine if any other nodes on current front are within search radius *r* of ideal location C (Choose D instead of C)

Advancing Front



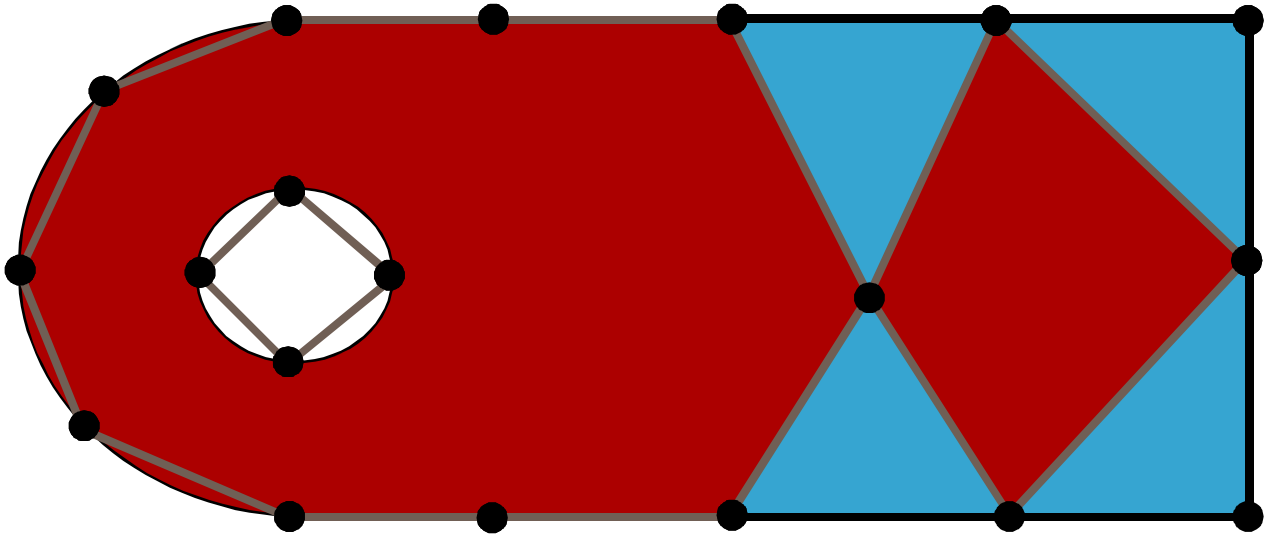
•Book-Keeping: New *front edges* added and deleted from *front* as triangles are formed
•Continue until no *front edges* remain on *front*

Advancing Front



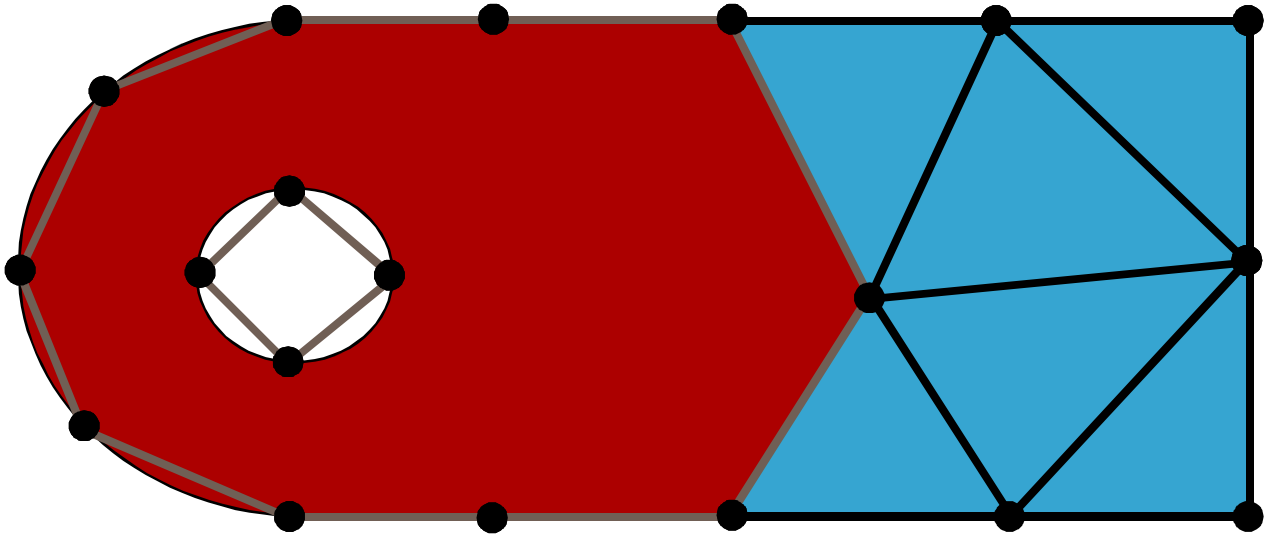
•Book-Keeping: New *front edges* added and deleted from *front* as triangles are formed
•Continue until no *front edges* remain on *front*

Advancing Front



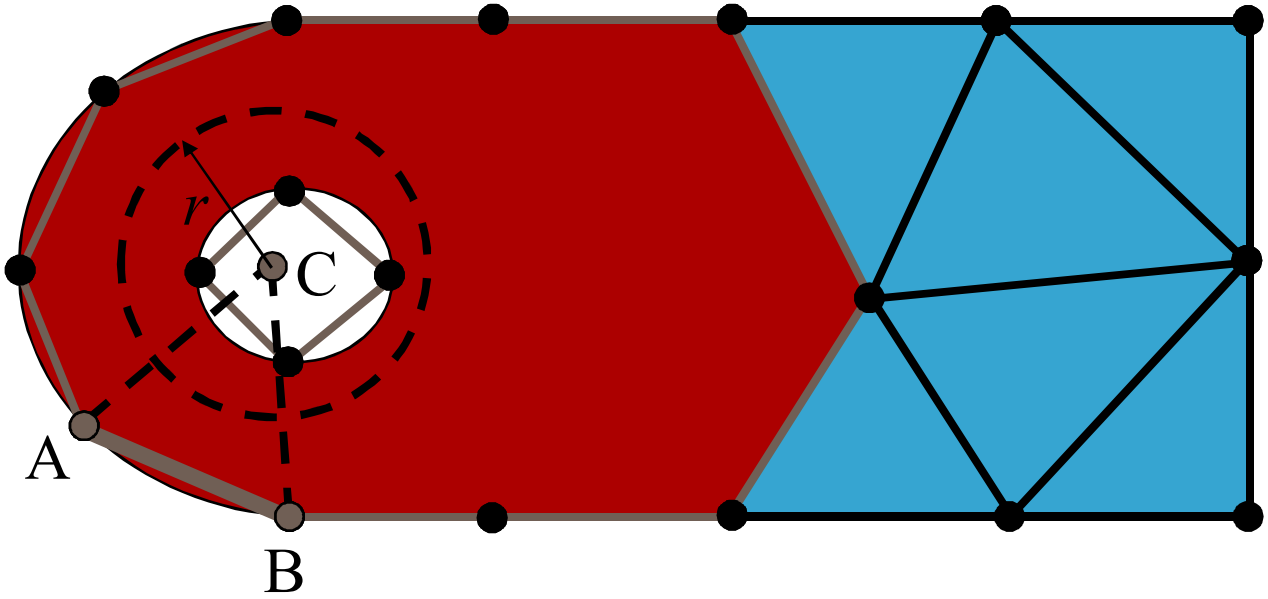
•Book-Keeping: New *front edges* added and deleted from *front* as triangles are formed
•Continue until no *front edges* remain on *front*

Advancing Front



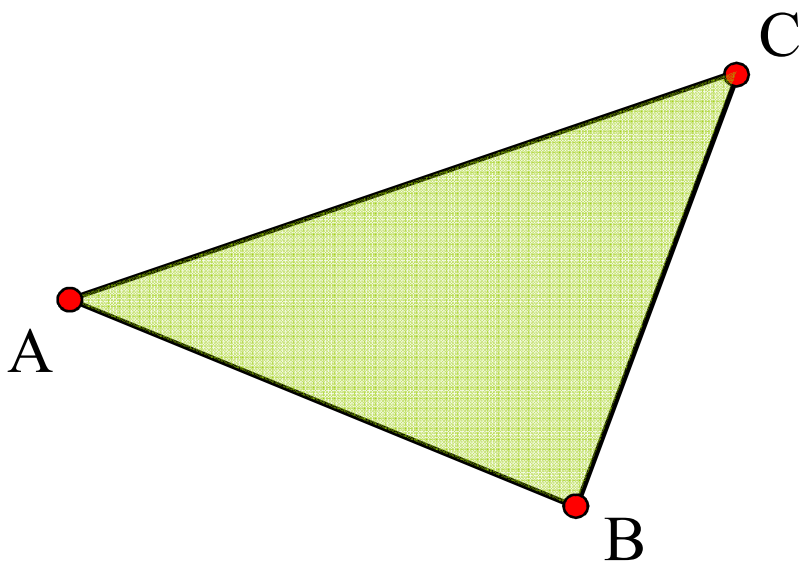
•Book-Keeping: New *front edges* added and deleted from *front* as triangles are formed
•Continue until no *front edges* remain on *front*

Advancing Front

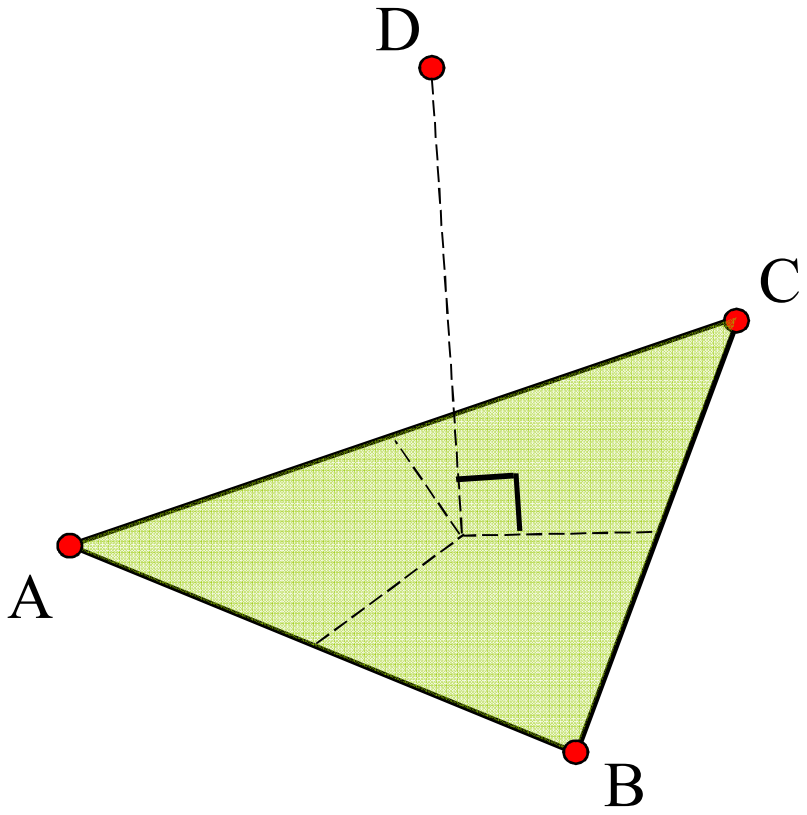


- Where multiple choices are available, use best quality (closest shape to equilateral)
- Reject any that would intersect existing front
- Reject any inverted triangles ($|AB \times AC| > 0$)
- (Lohner,88;96)(Lo,91)

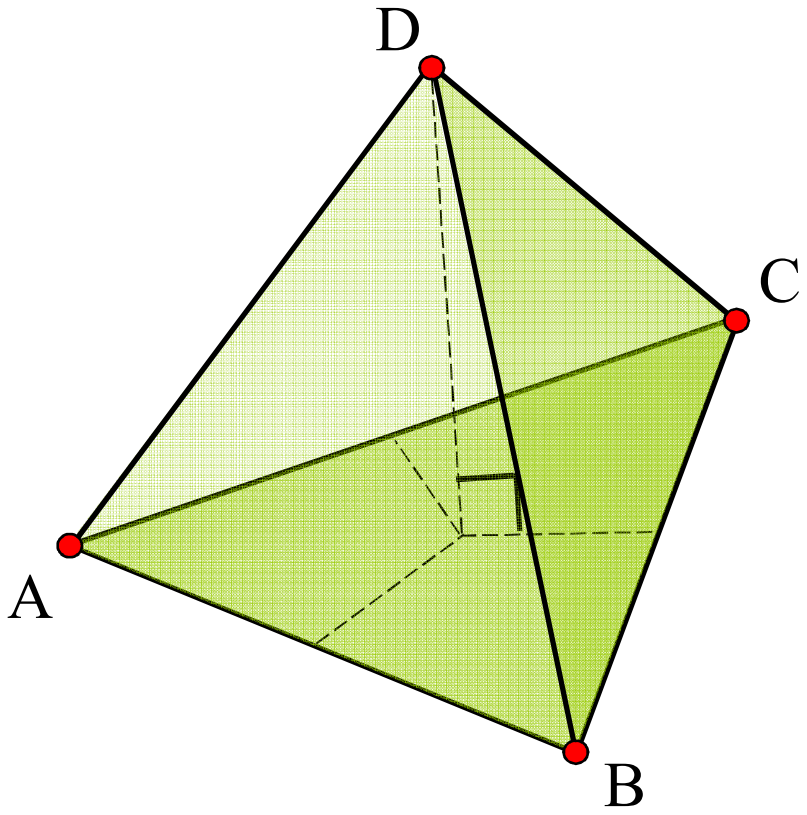
Advancing Front



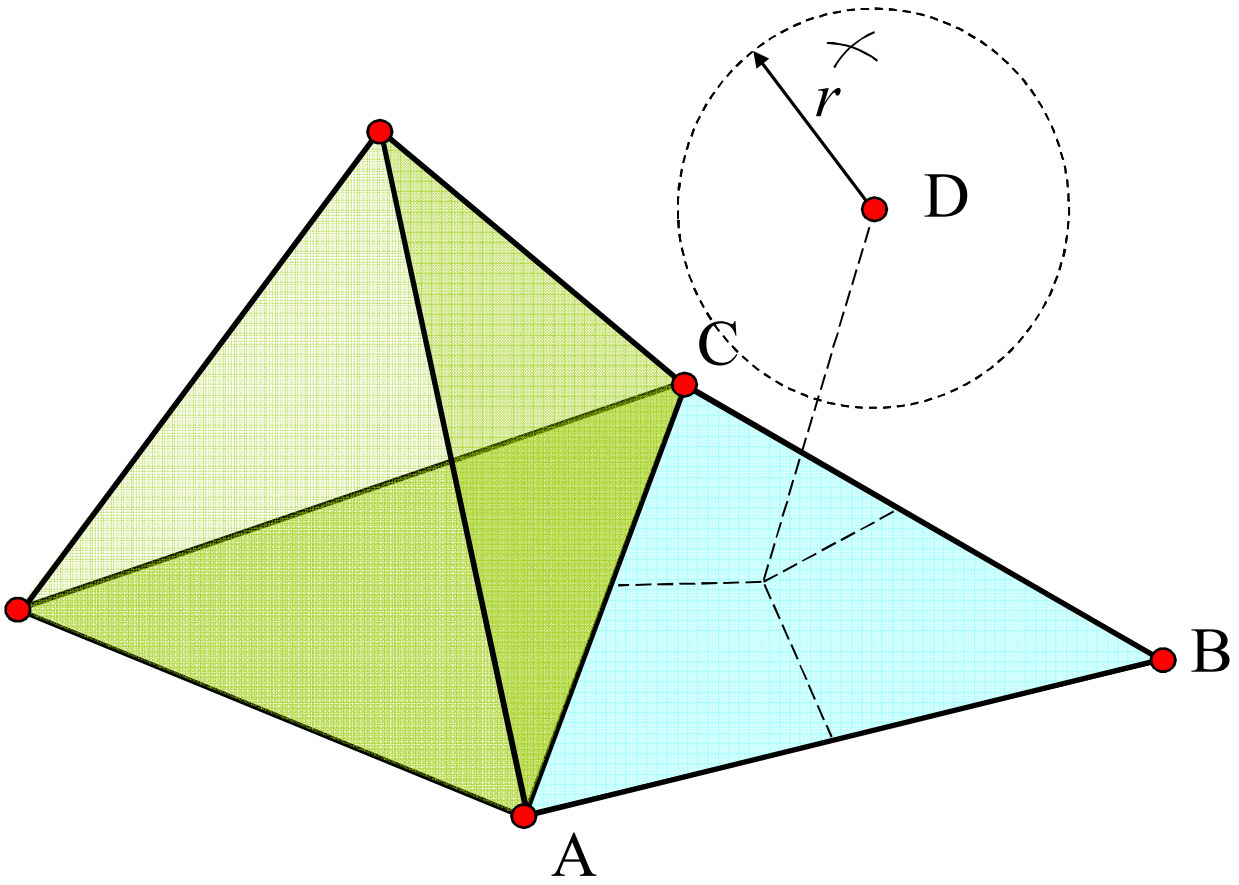
Advancing Front



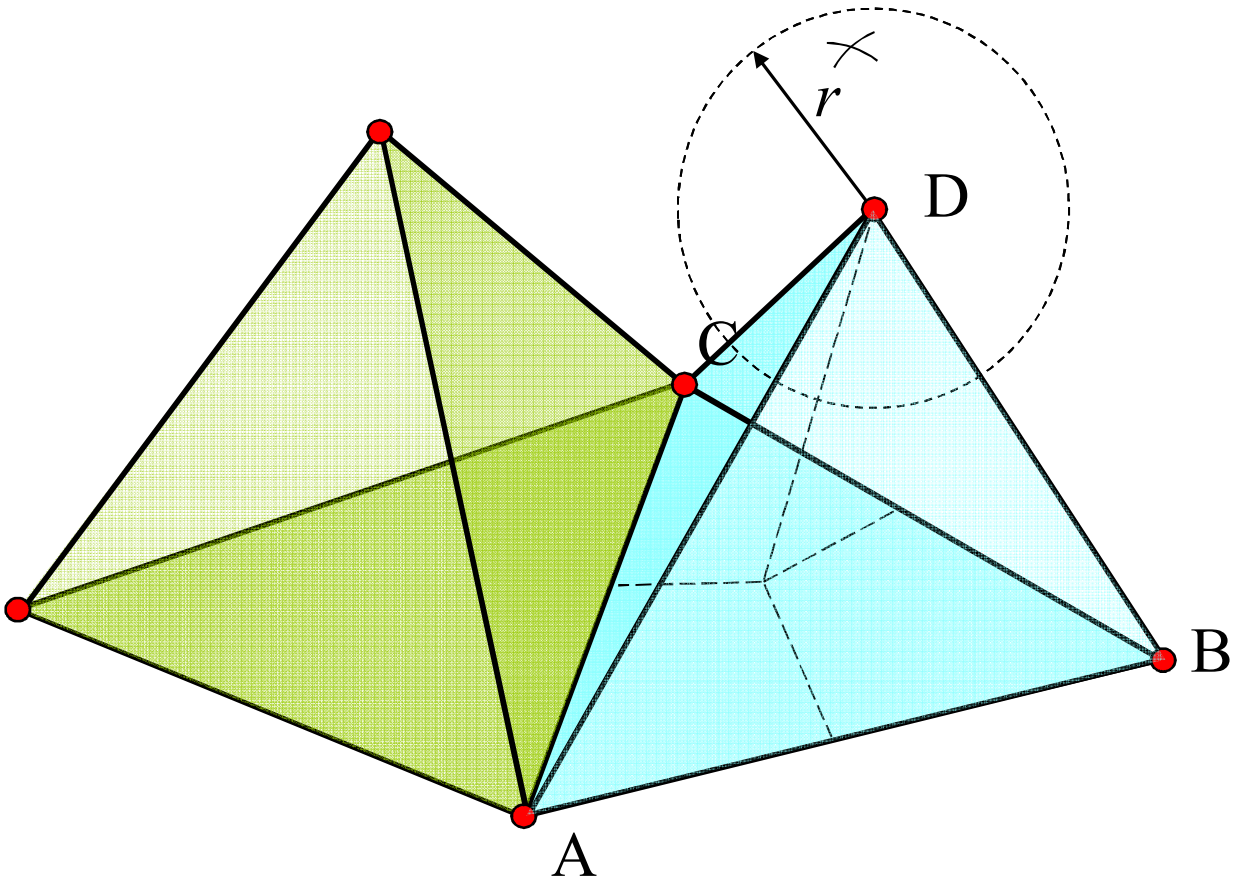
Advancing Front



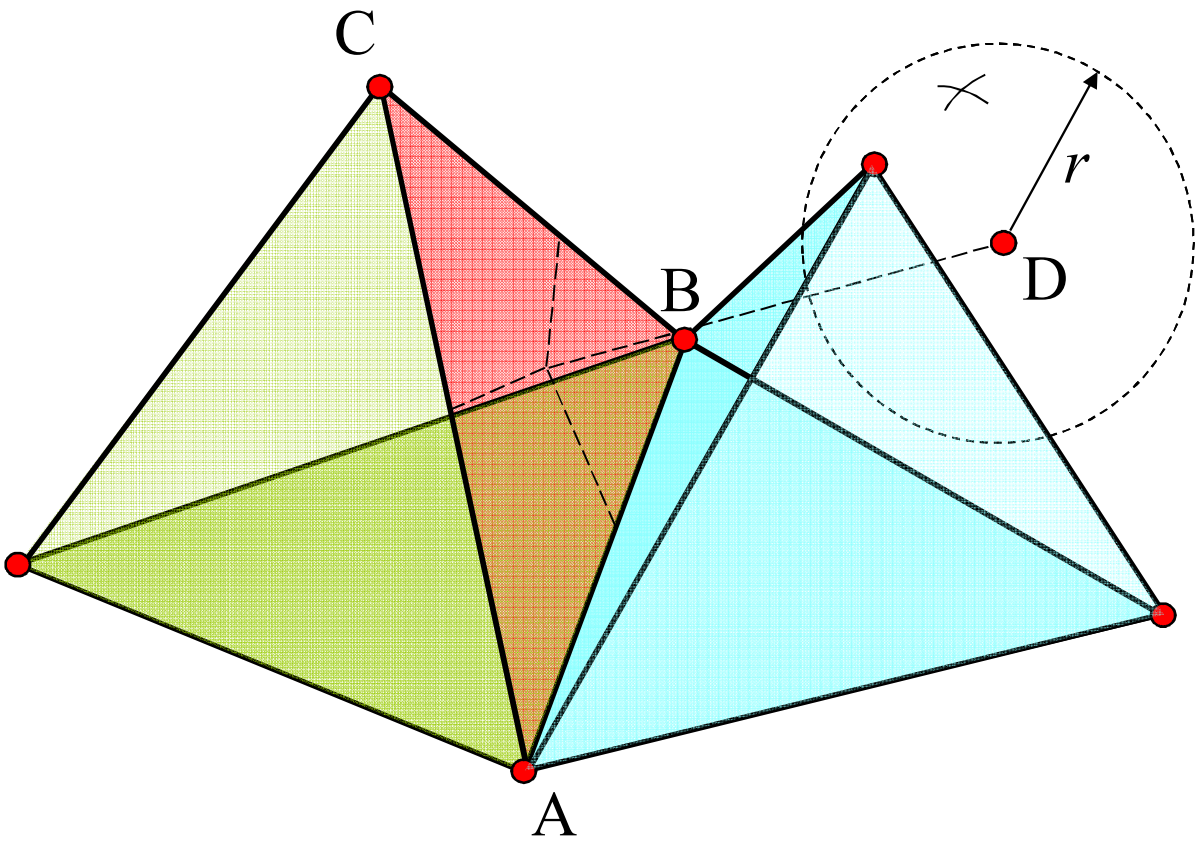
Advancing Front



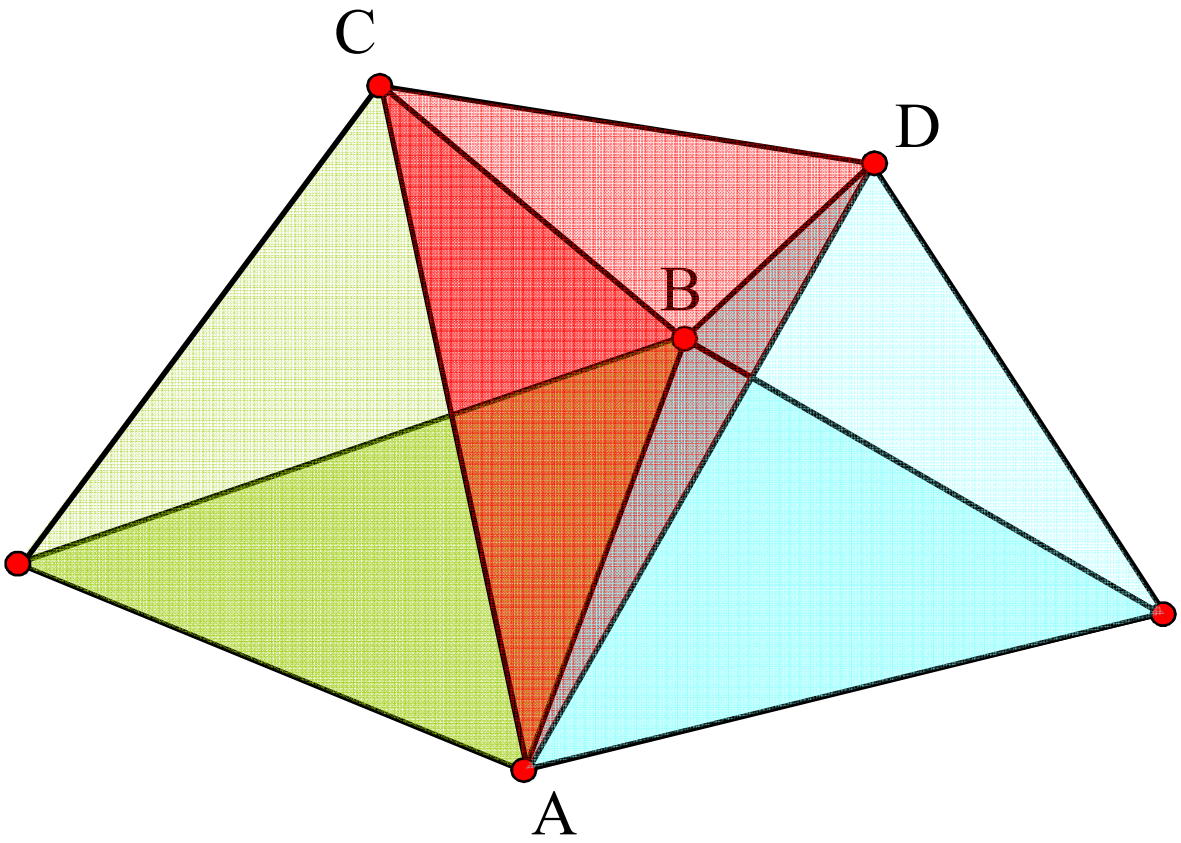
Advancing Front



Advancing Front

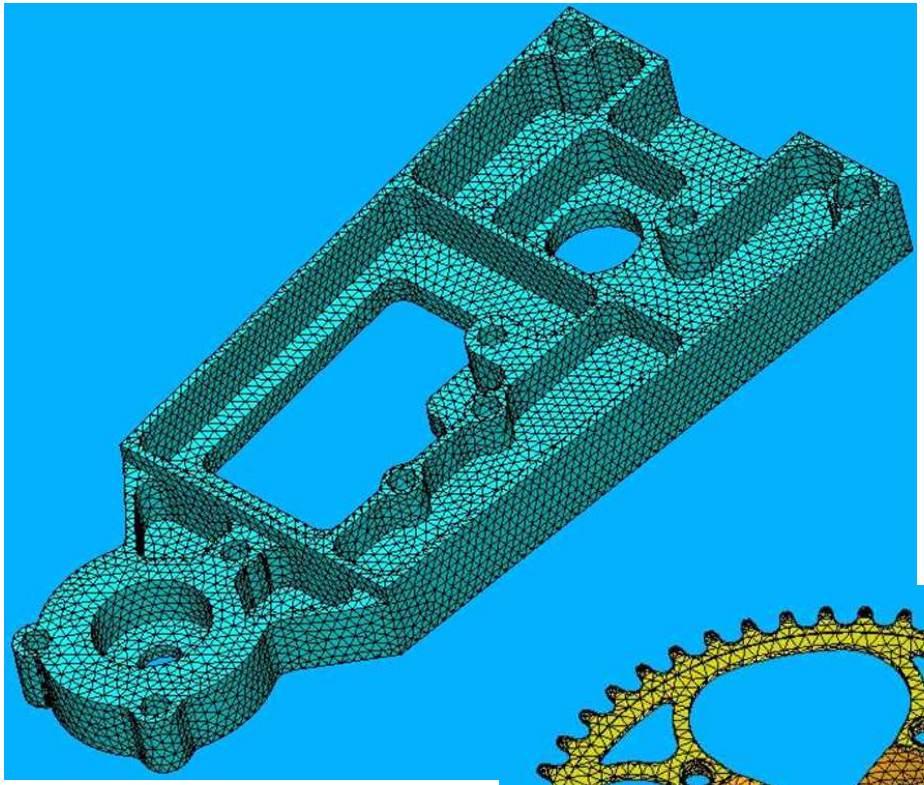


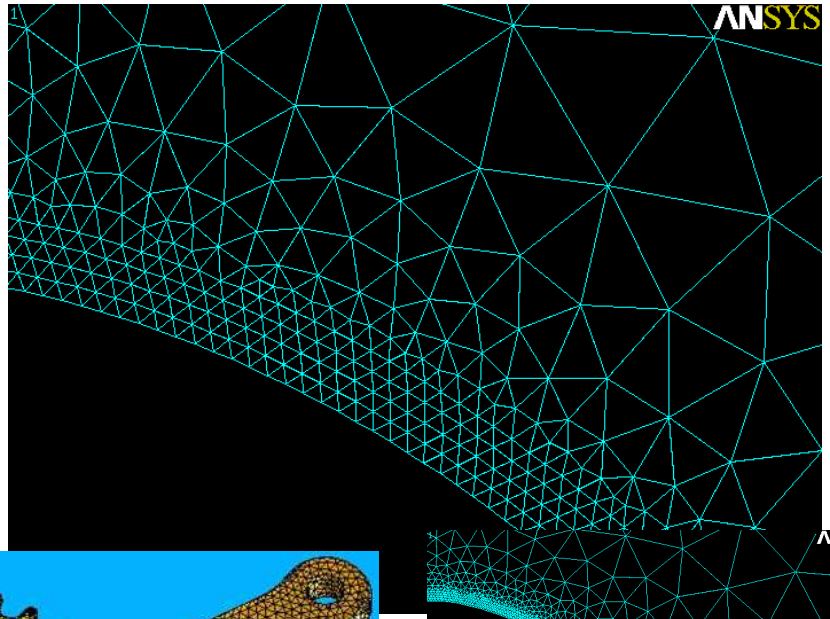
Advancing Front

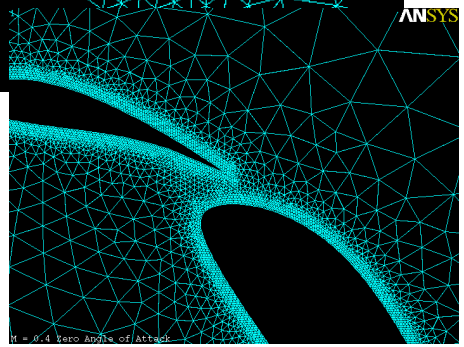


Advancing Front

Sandia National Laboratories

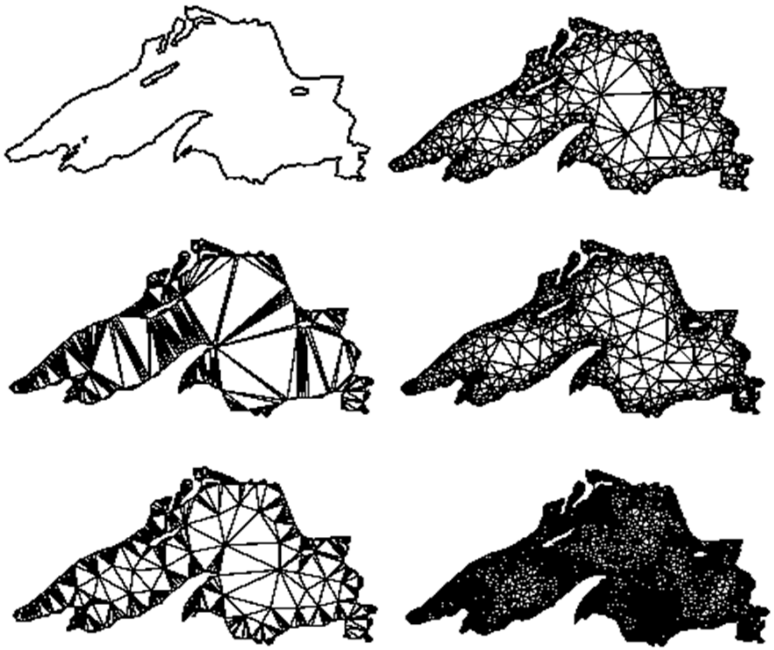




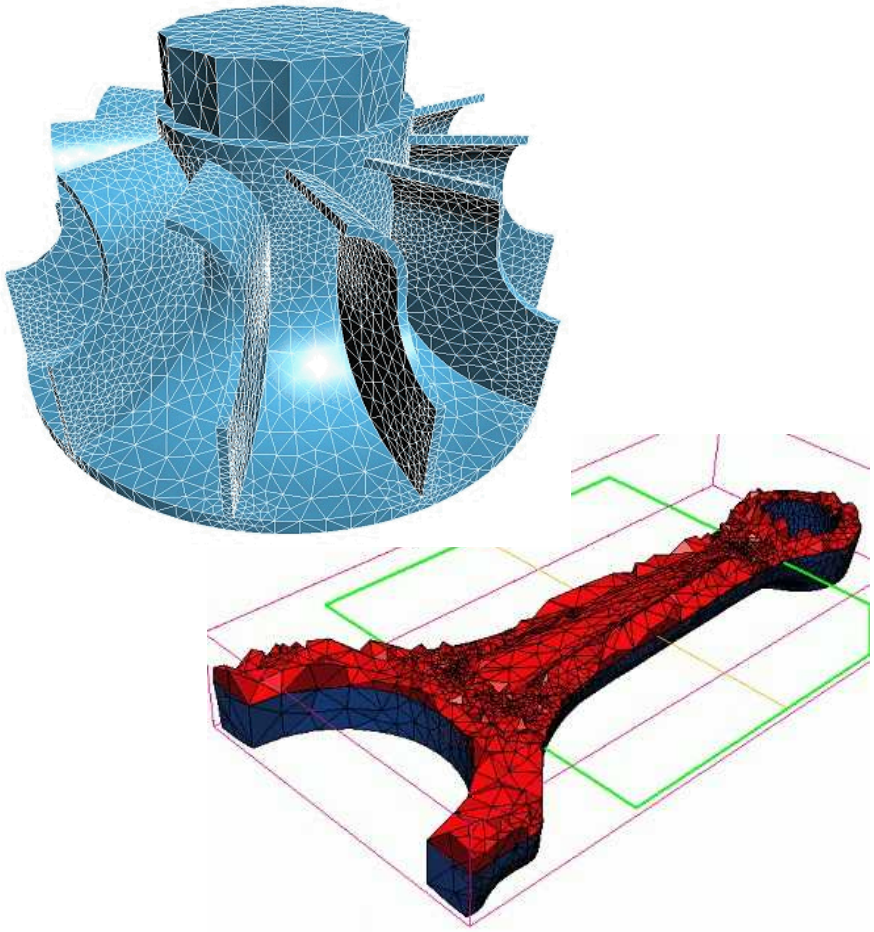


Ansys, Inc.
www.ansys.com

Delaunay

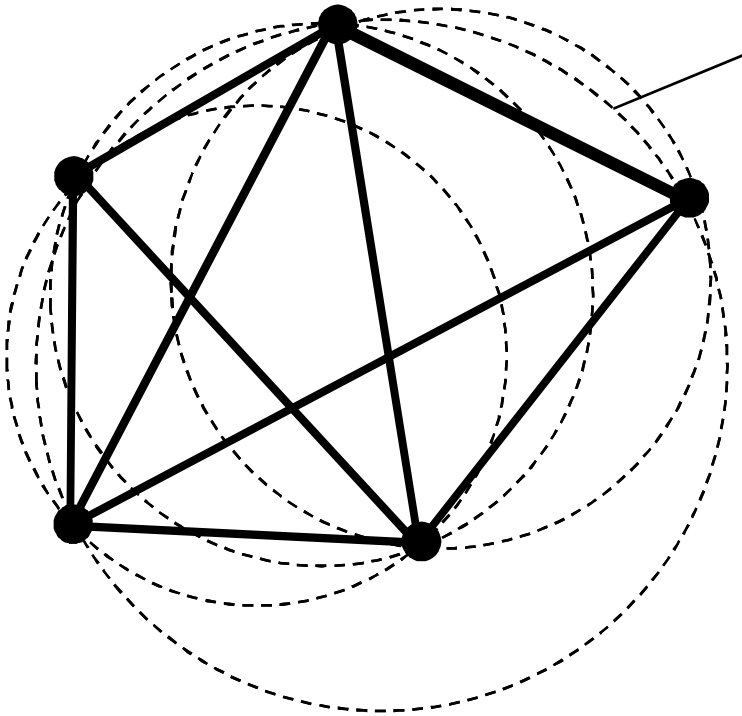


Triangle
Jonathon Shewchuk
http://www-2.cs.cmu.edu/~quake/triangle.html



Tetmesh-GHS3D
INRIA, France
http://www.simulog.fr/tetmesh/

Delaunay



Empty Circle Property:
No other vertex is contained within the circumcircle of any triangle

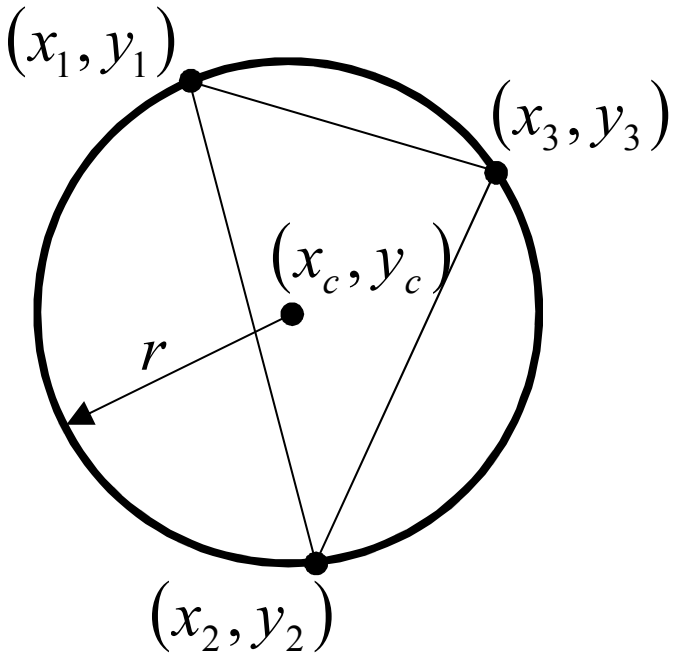
Delaunay

Computing the Circumcircle center and radius

$$Ax + By + C = x^2 + y^2$$

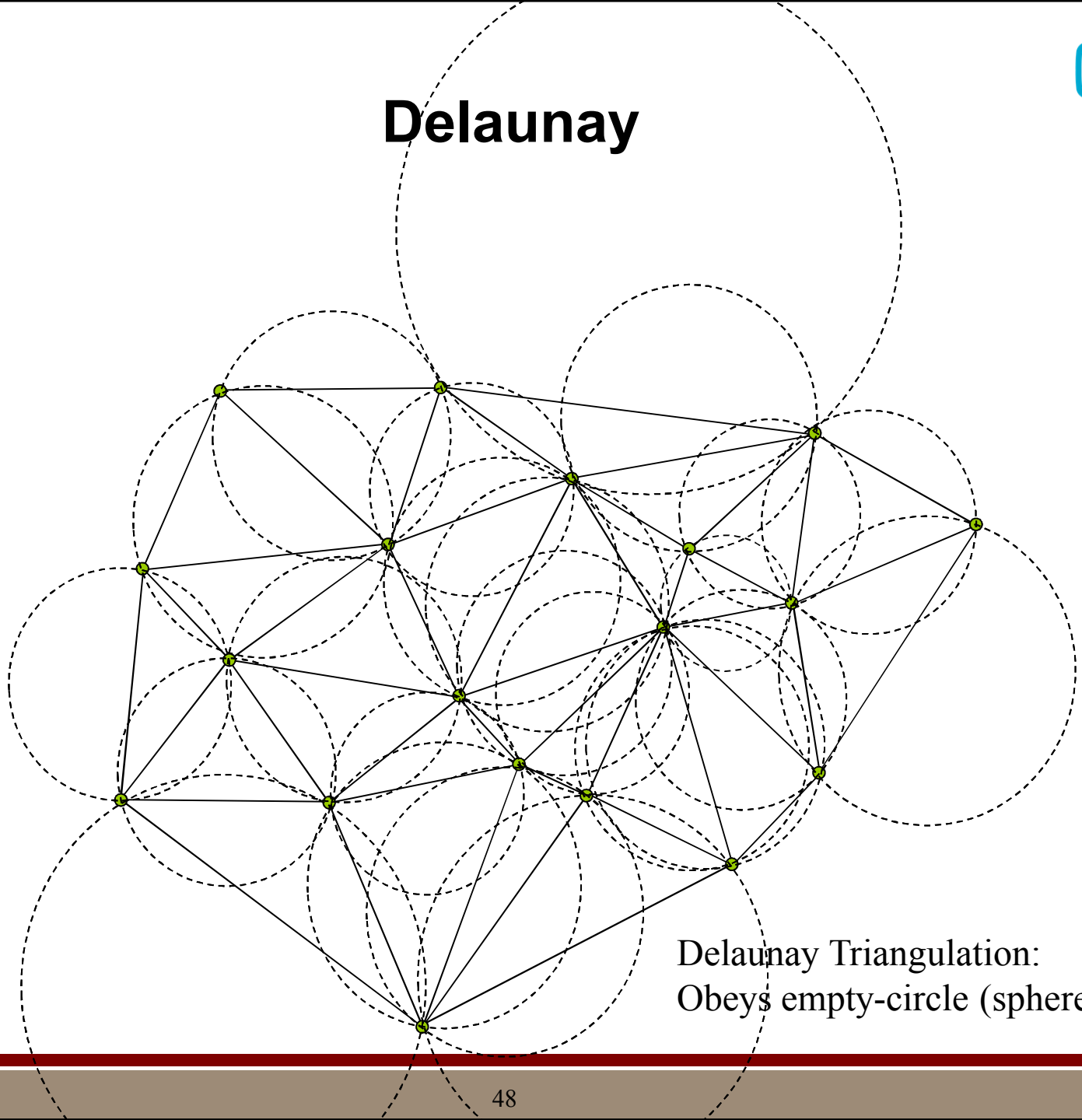
$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \begin{Bmatrix} A \\ B \\ C \end{Bmatrix} = \begin{bmatrix} x_1^2 + y_1^2 \\ x_2^2 + y_2^2 \\ x_3^2 + y_3^2 \end{bmatrix}$$

$$x_c = \frac{A}{2}, y_c = \frac{B}{2}, r = \sqrt{C + x_c^2 + y_c^2}$$



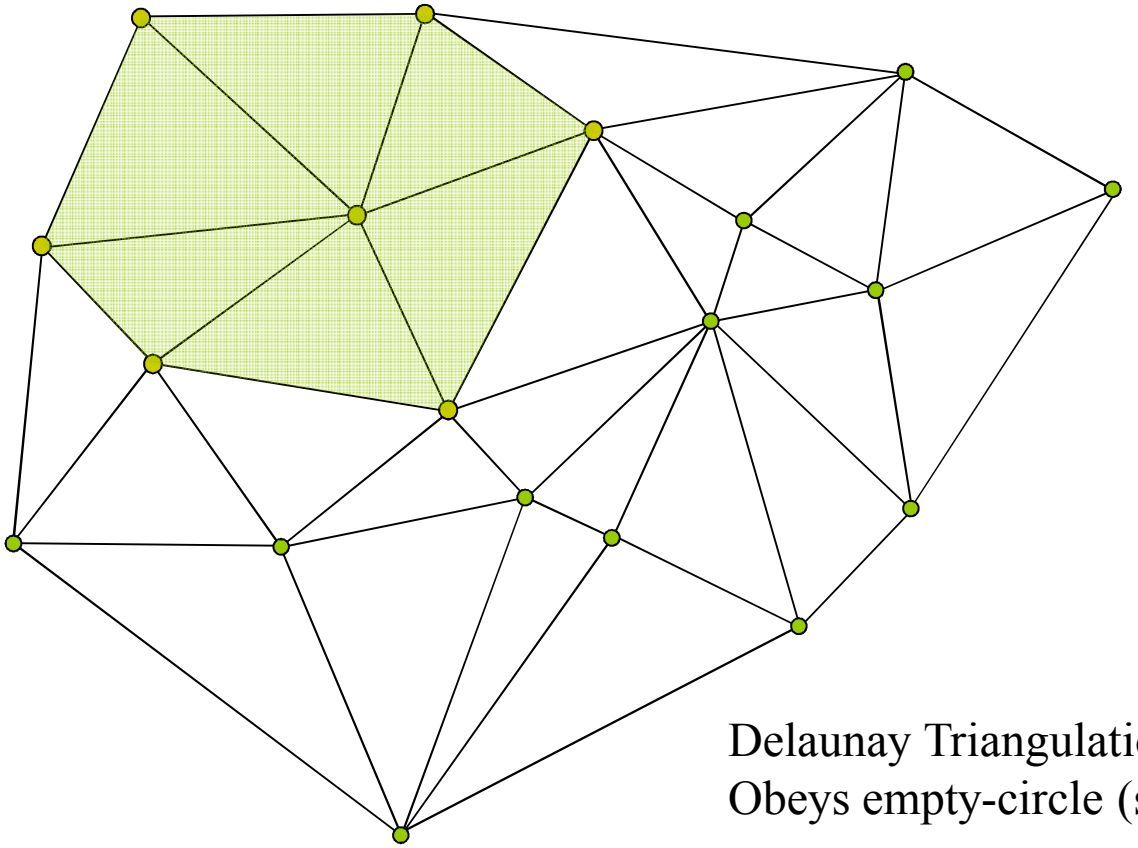
47

Delaunay



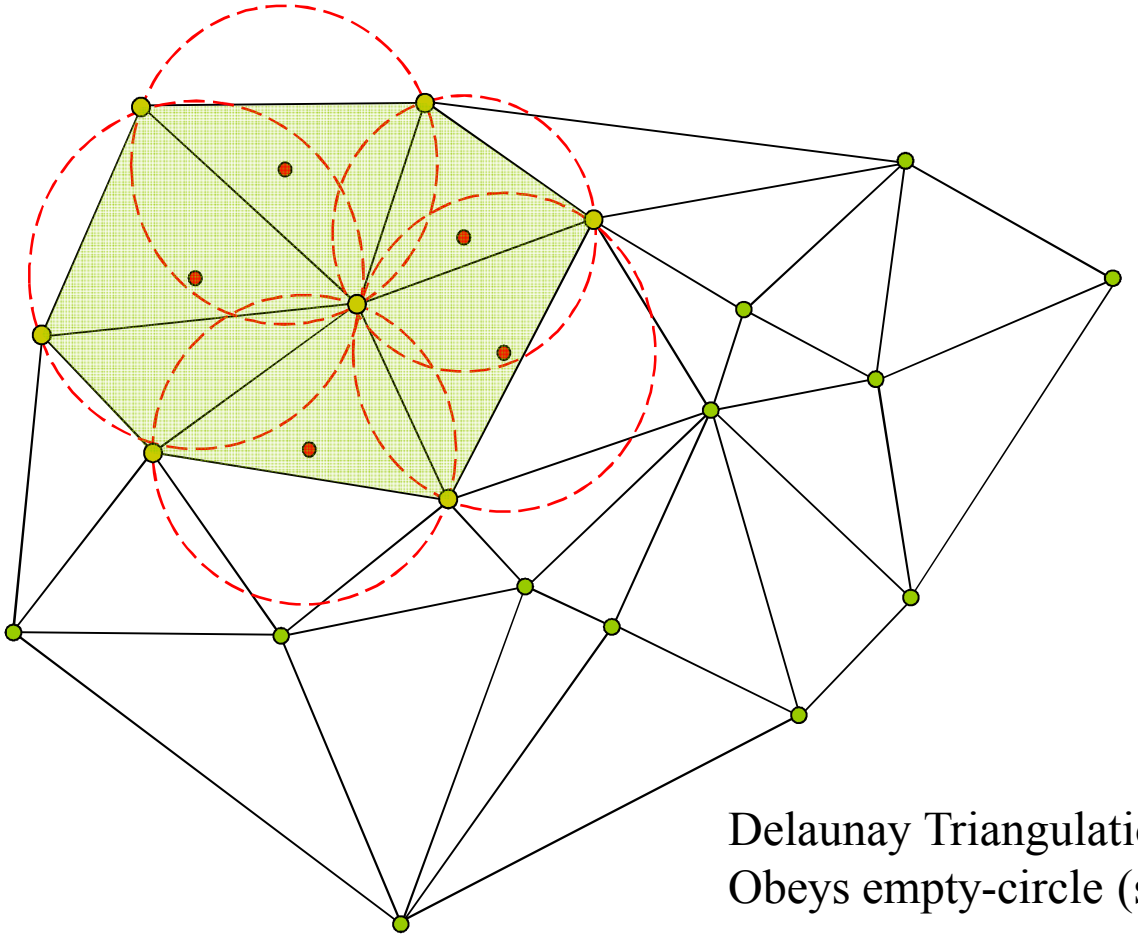
Delaunay Triangulation:
Obeys empty-circle (sphere) property

Delaunay



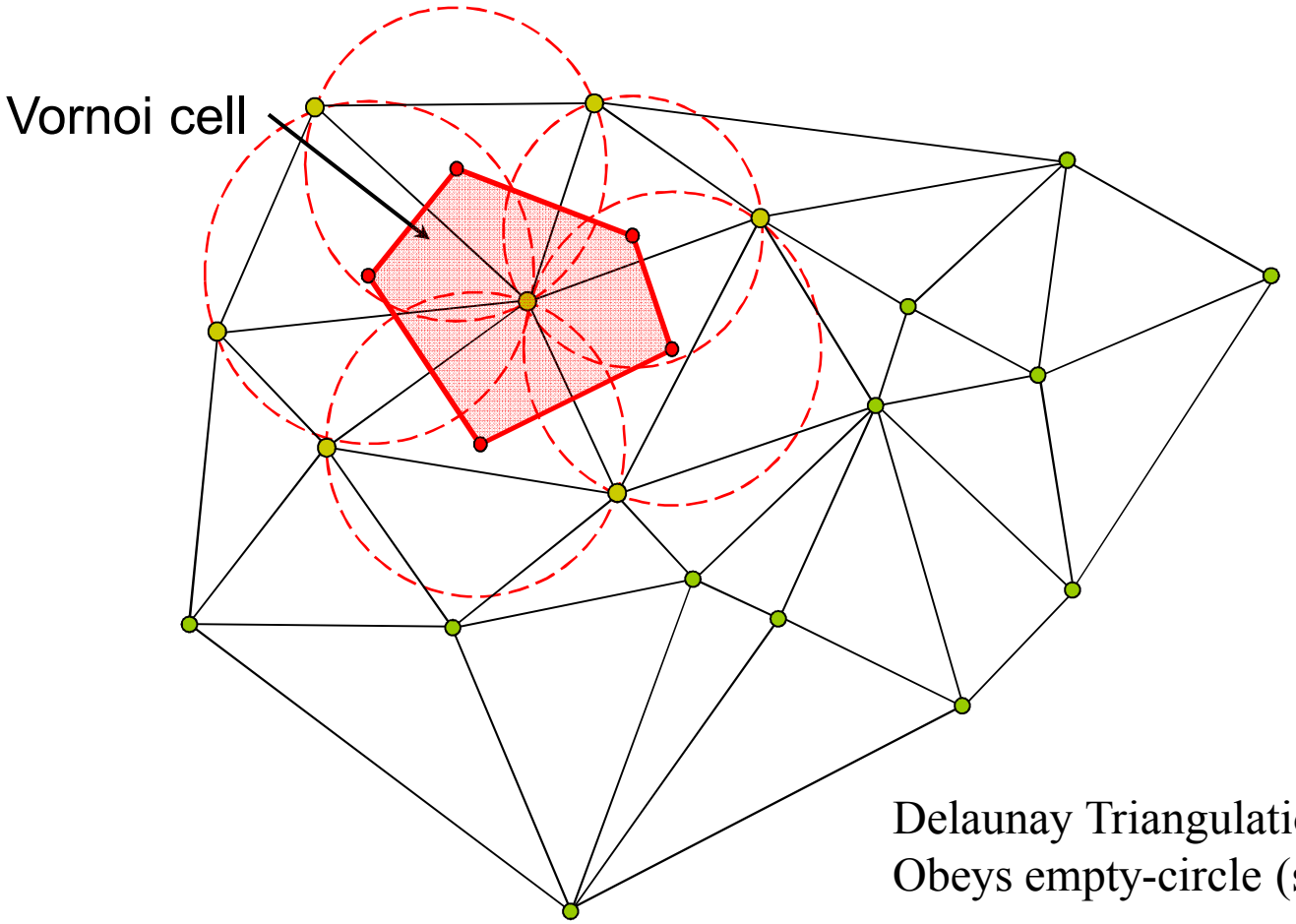
Delaunay Triangulation:
Obeys empty-circle (sphere) property

Delaunay



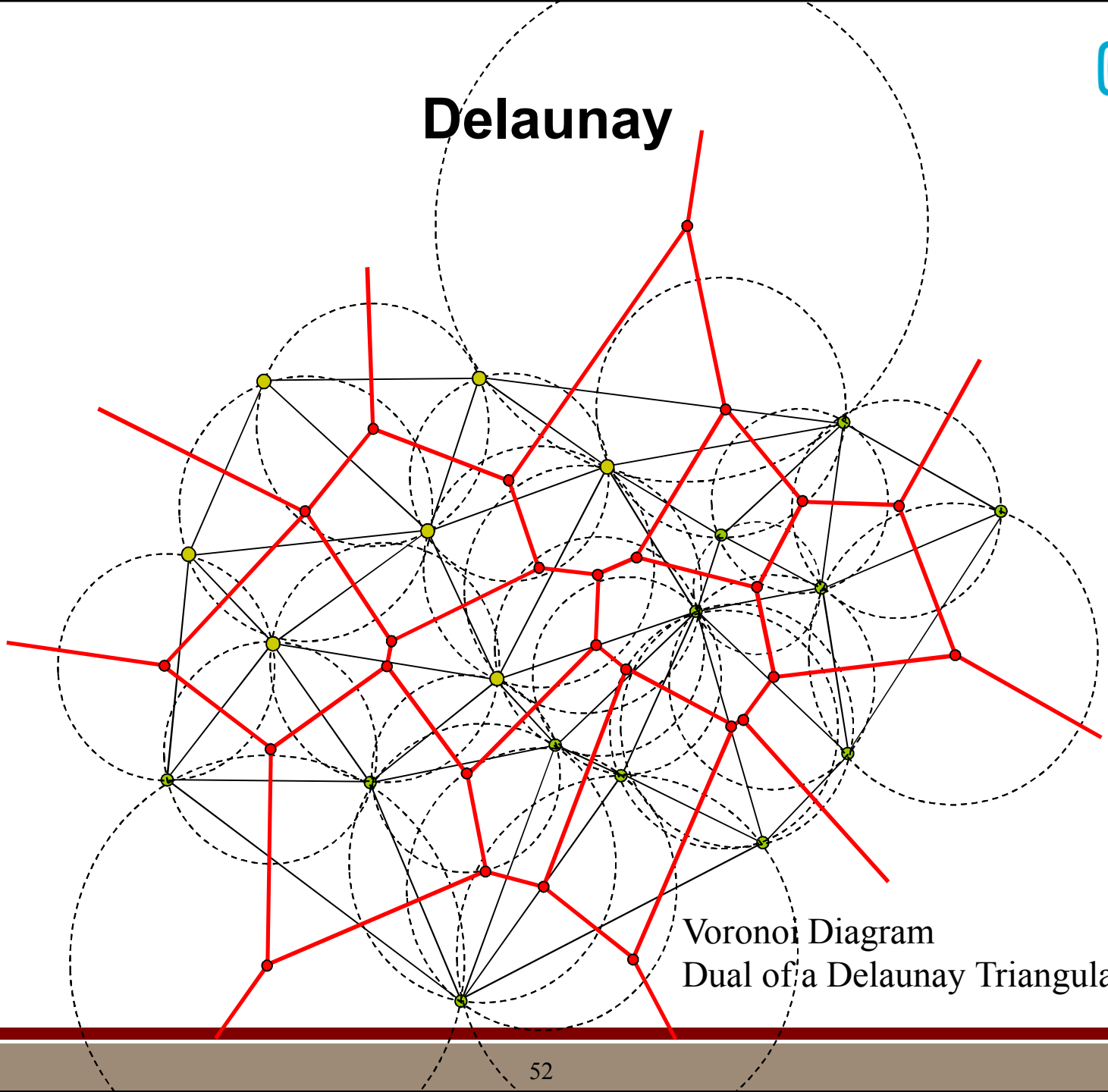
Delaunay Triangulation:
Obeys empty-circle (sphere) property

Delaunay



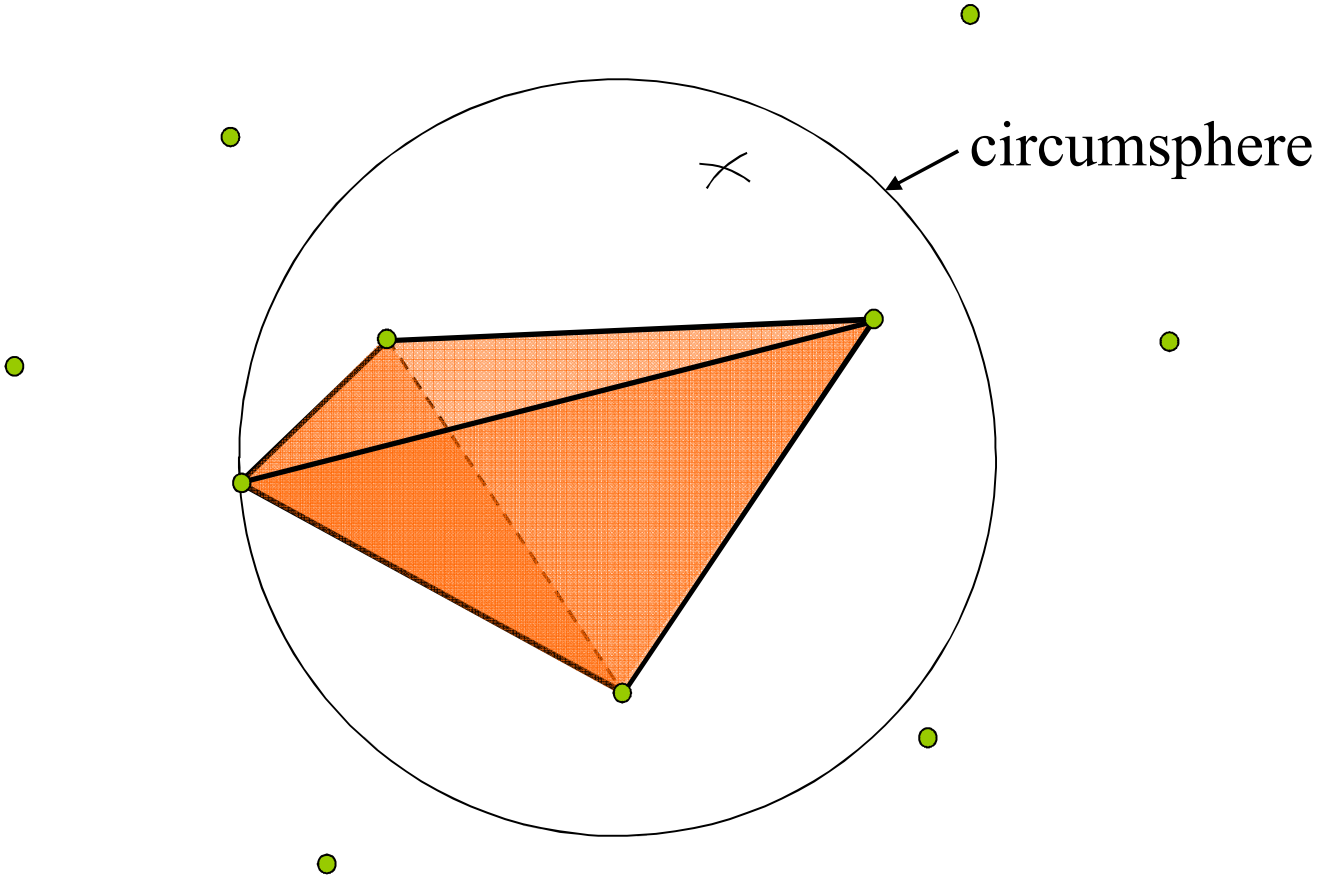
Delaunay Triangulation:
Obeys empty-circle (sphere) property

Delaunay



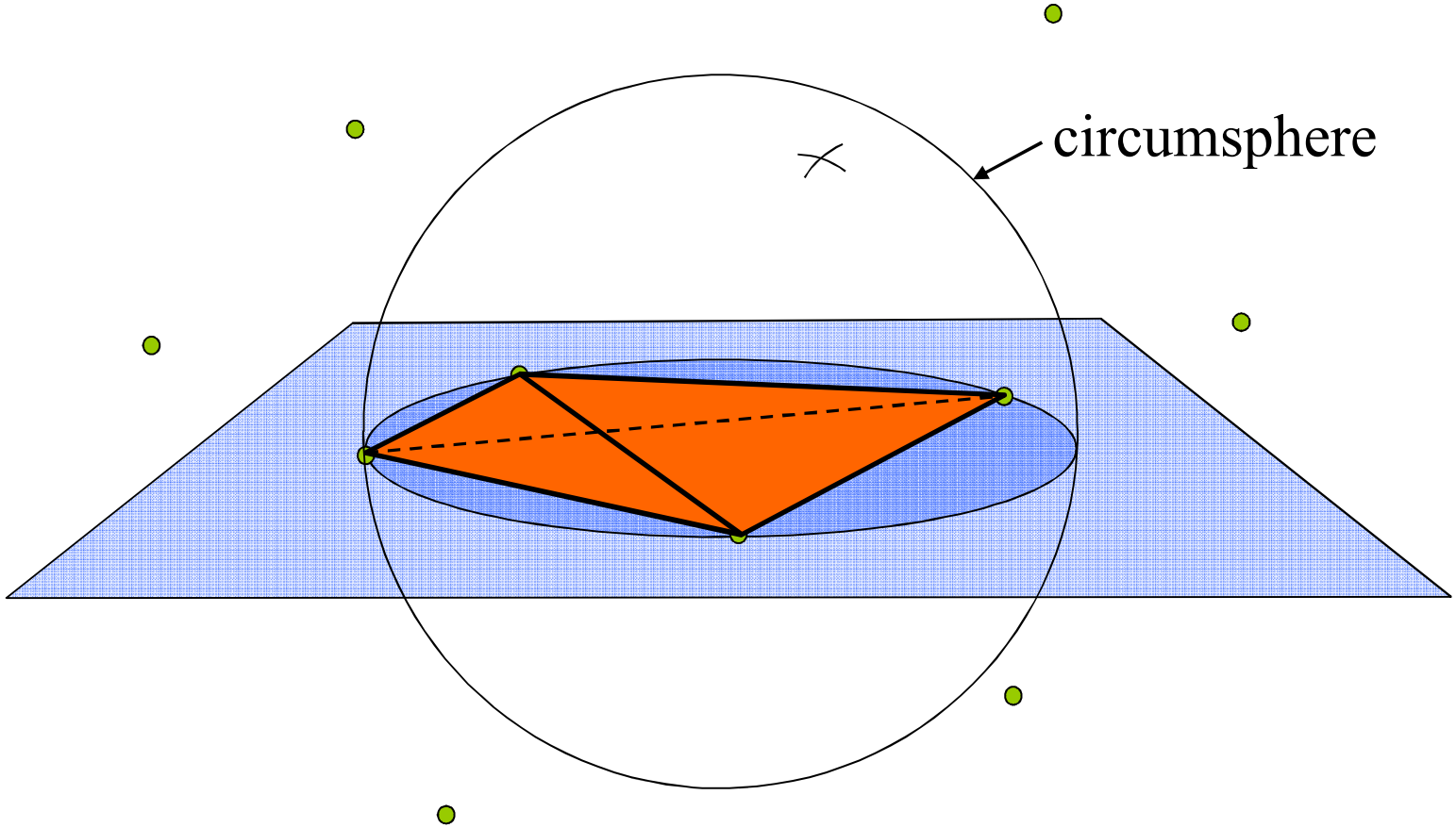
Voronoi Diagram
Dual of a Delaunay Triangulation

Delaunay



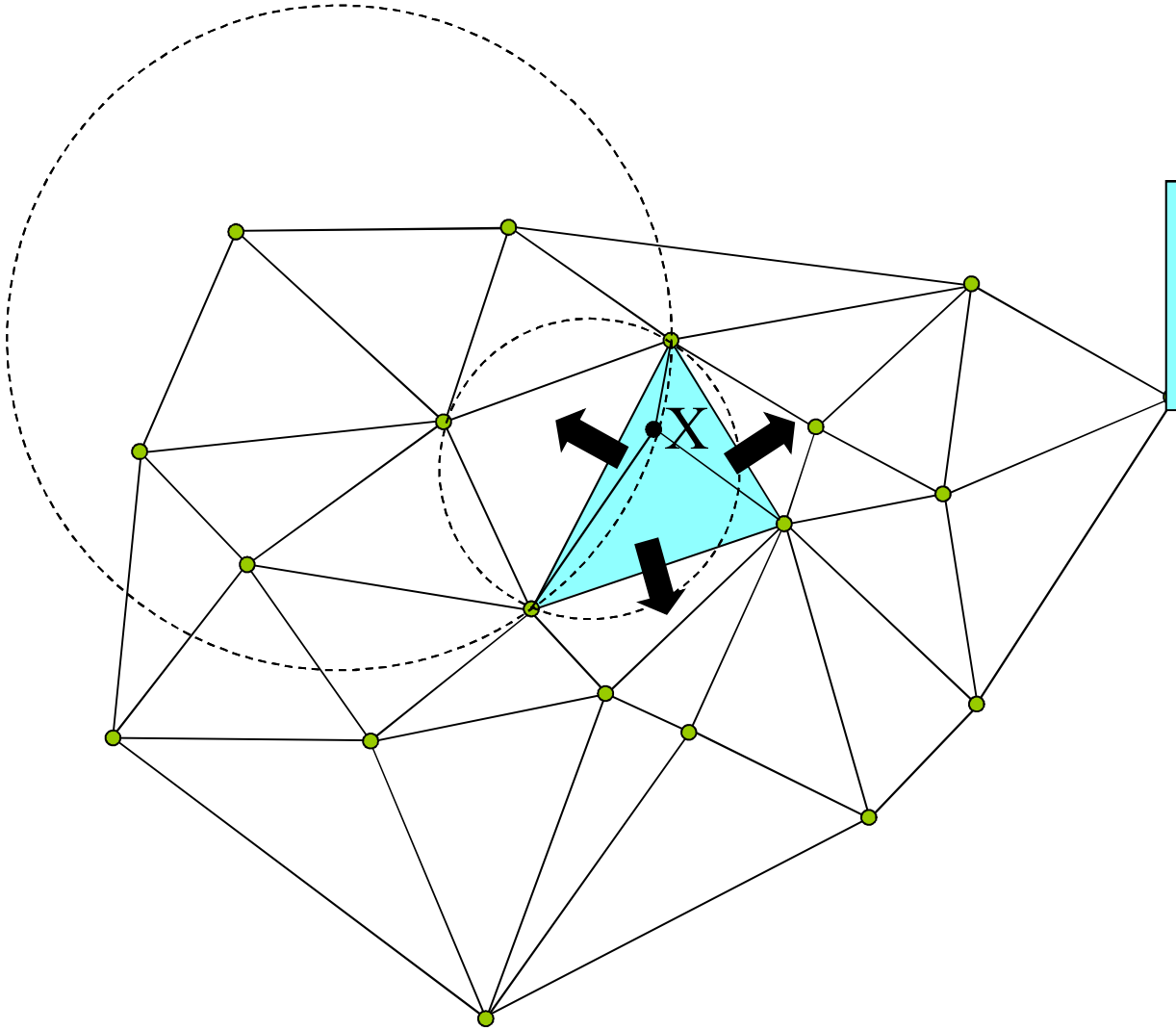
Empty Sphere Property:
No other vertex is contained within the circumsphere of any tetrahedron

Delaunay



Empty Sphere Property:
No other vertex is contained within the circumsphere of any tetrahedron

Delaunay

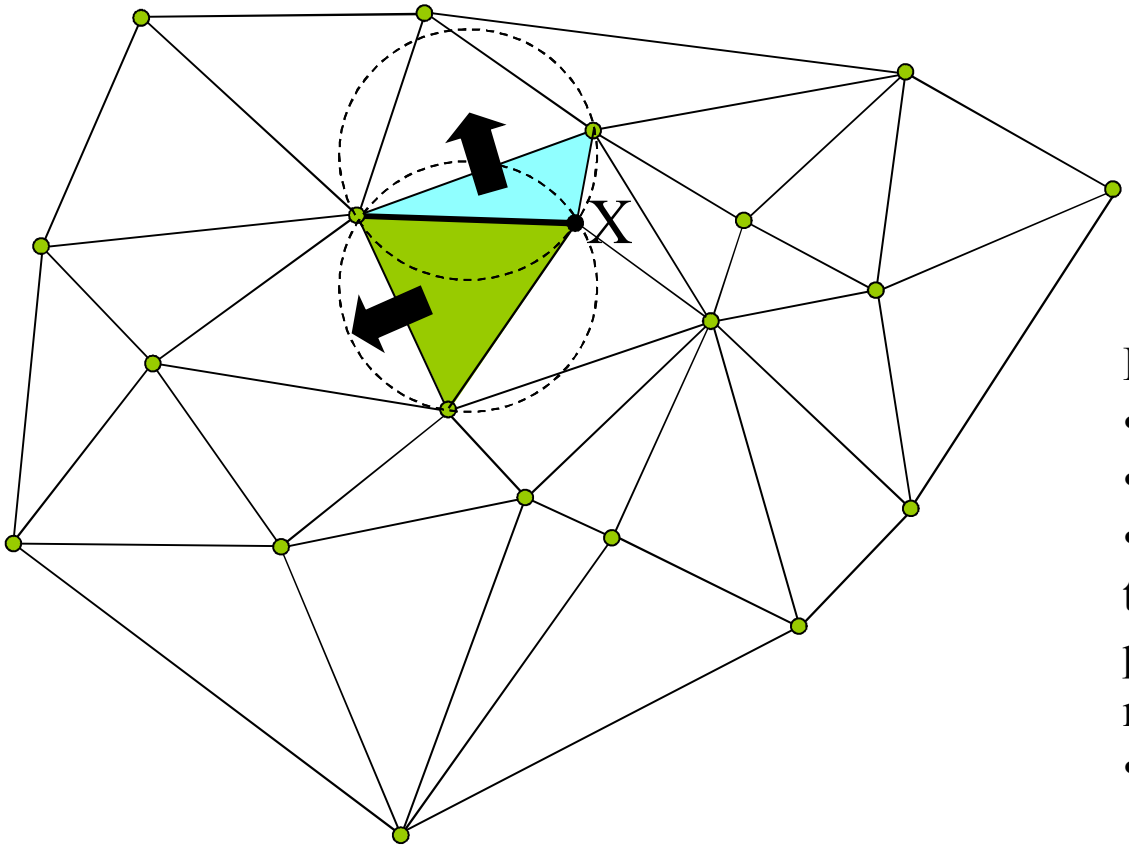


Given a Delaunay Triangulation of *n* nodes, How do I insert node *n+1* ?

Lawson Algorithm

- Locate triangle containing X
- Subdivide triangle
- Recursively check adjoining triangles to ensure empty-circle property. Swap diagonal if needed
- (Lawson,77)

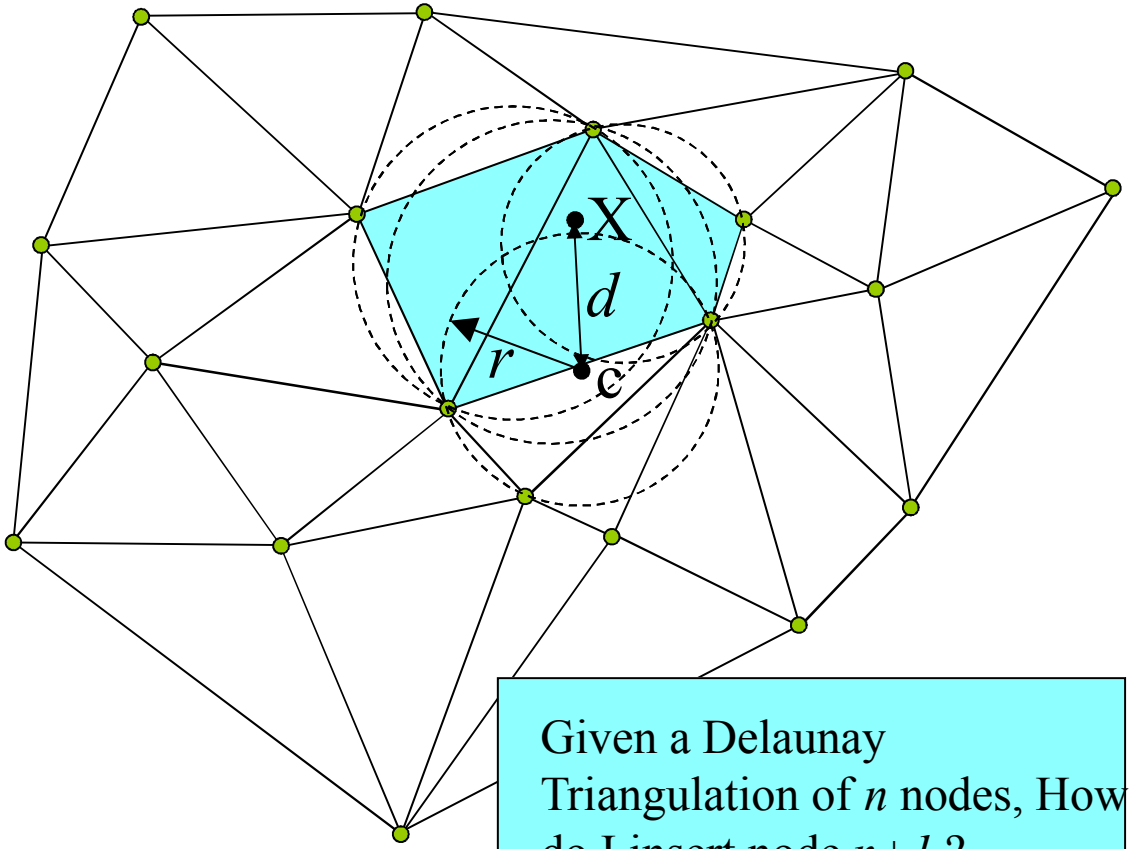
Delaunay



Lawson Algorithm

- Locate triangle containing X
- Subdivide triangle
- Recursively check adjoining triangles to ensure empty-circle property. Swap diagonal if needed
- (Lawson,77)

Delaunay

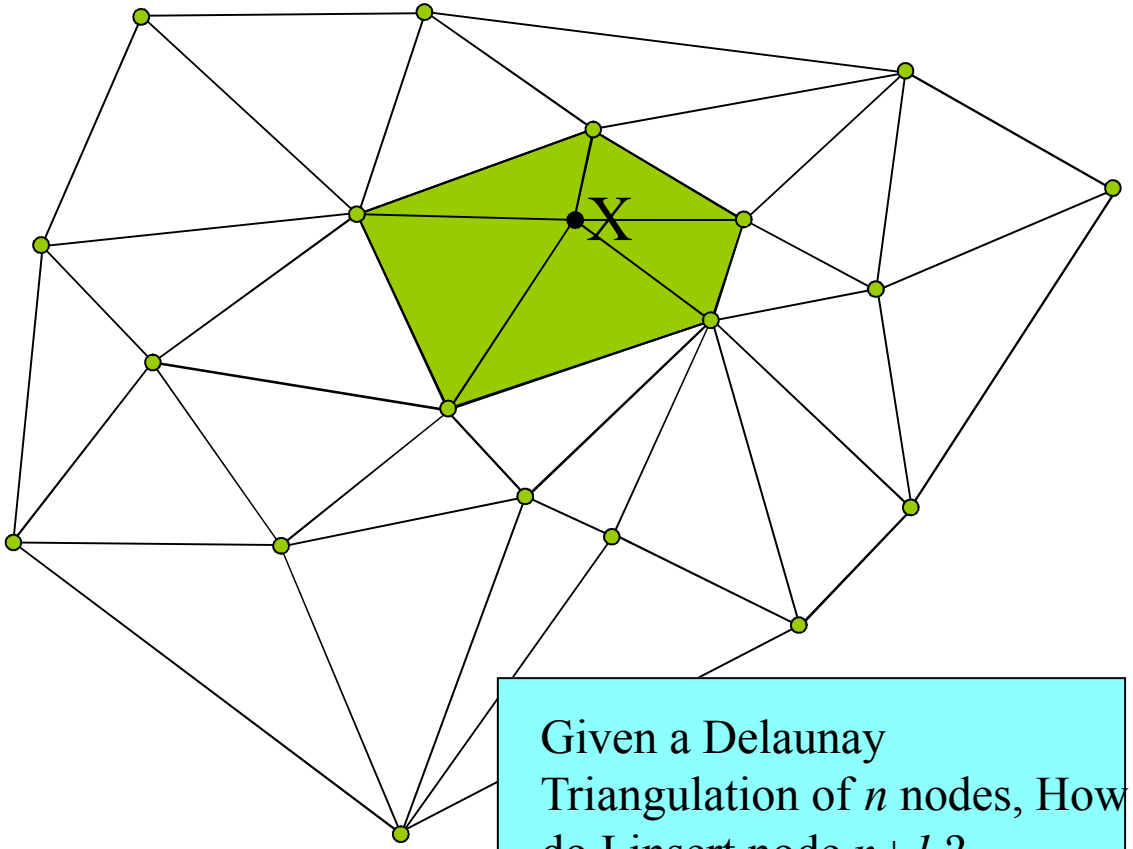


Given a Delaunay Triangulation of *n* nodes, How do I insert node *n+1* ?

Bowyer-Watson Algorithm

- Locate triangle that contains the point
- Search for all triangles whose circumcircle contain the point ($d<r$)
- Delete the triangles (creating a void in the mesh)
- Form new triangles from the new point and the void boundary
- (Watson,81)

Delaunay

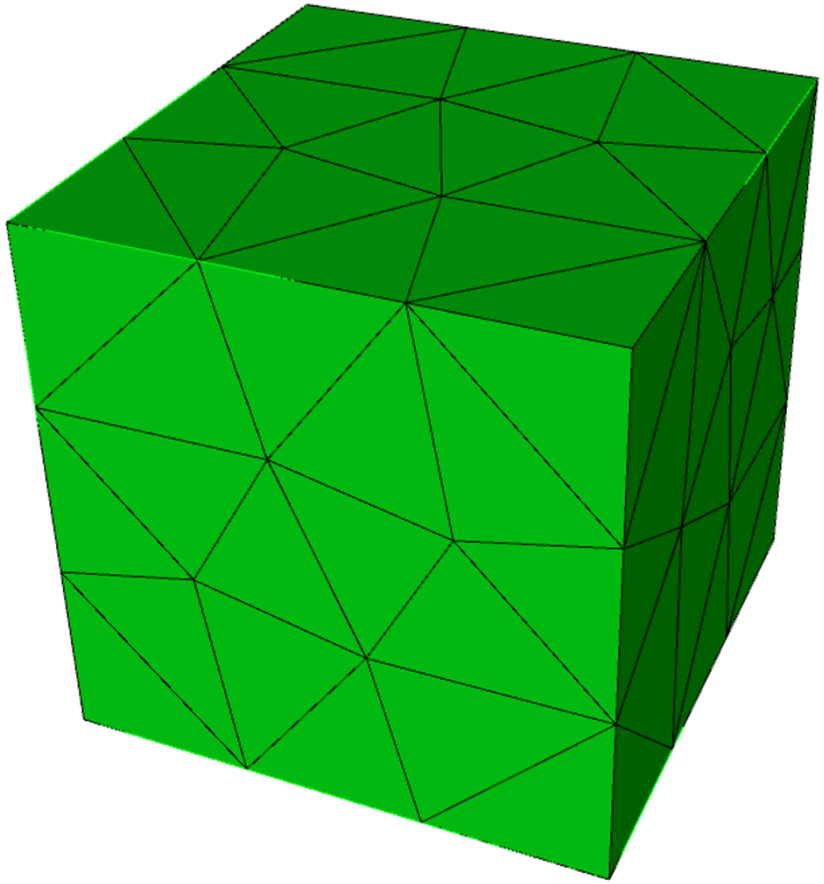


Given a Delaunay Triangulation of *n* nodes, How do I insert node *n+1* ?

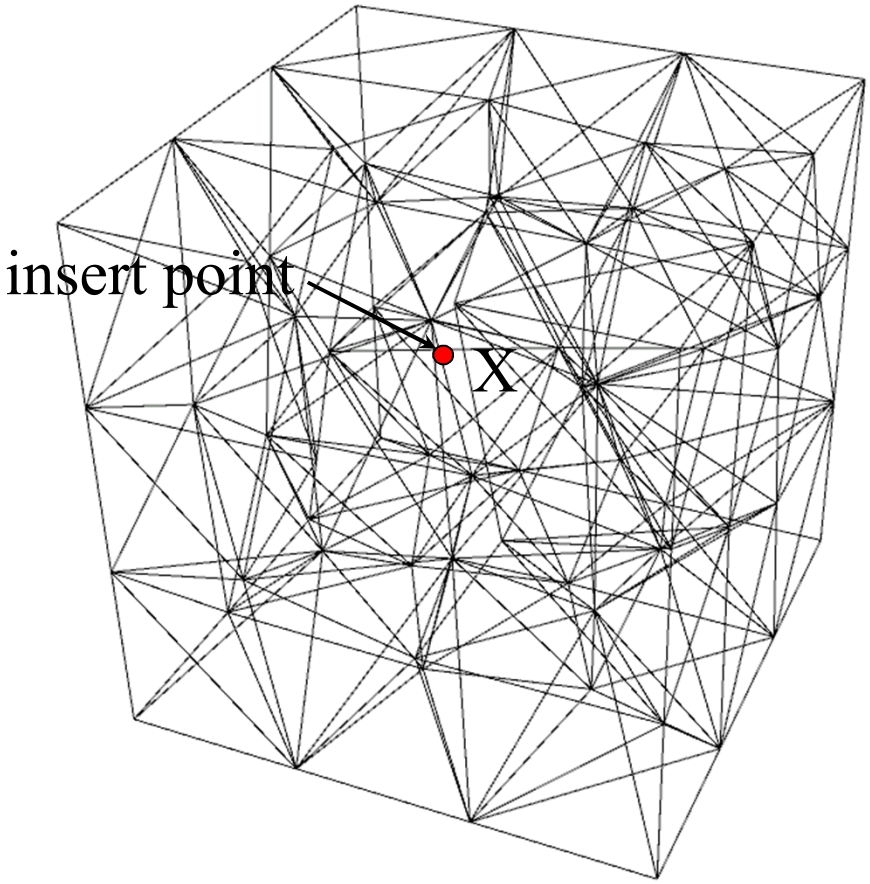
Bowyer-Watson Algorithm

- Locate triangle that contains the point
- Search for all triangles whose circumcircle contain the point ($d<r$)
- Delete the triangles (creating a void in the mesh)
- Form new triangles from the new point and the void boundary
- (Watson,81)

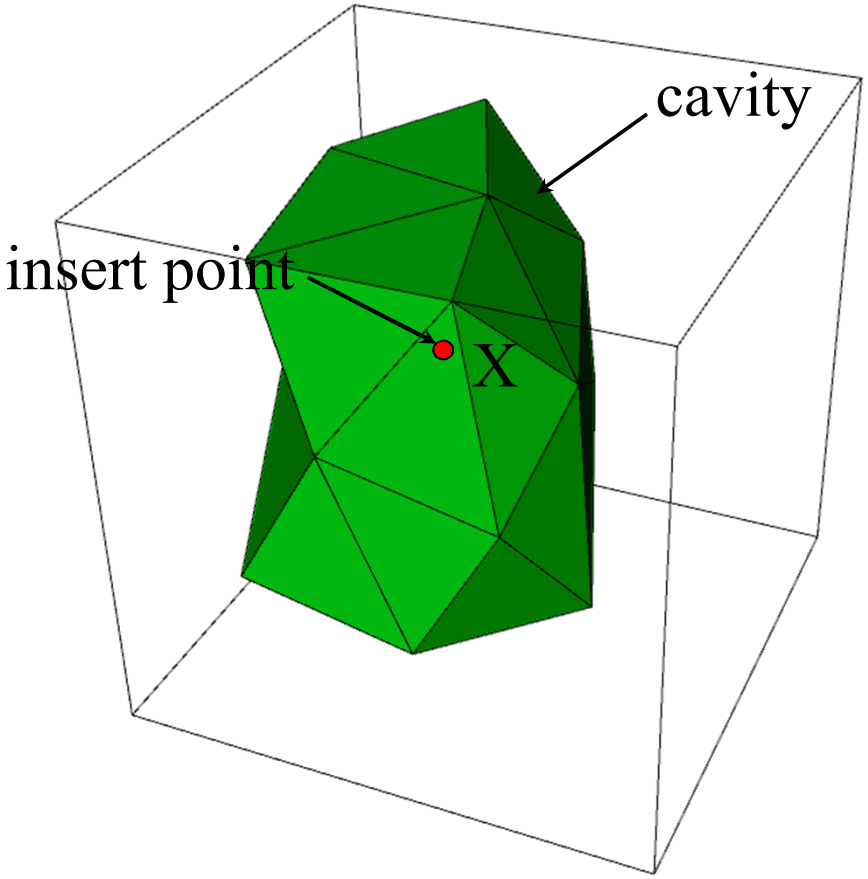
Delaunay



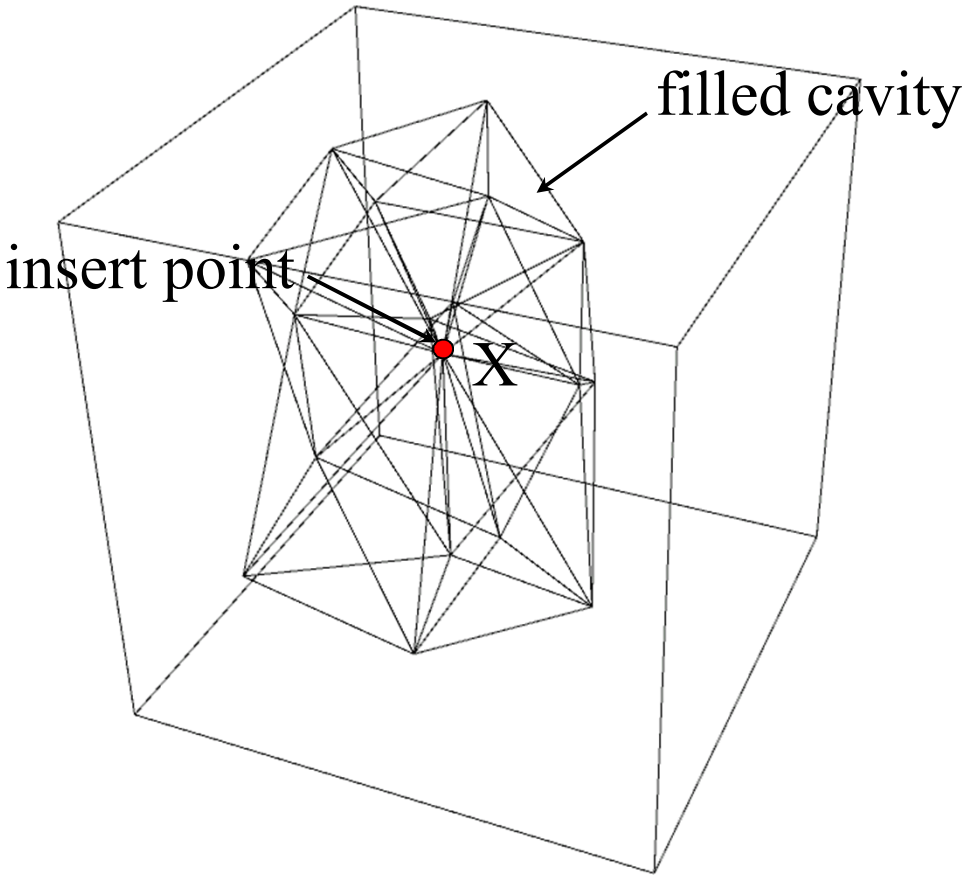
Delaunay



Delaunay

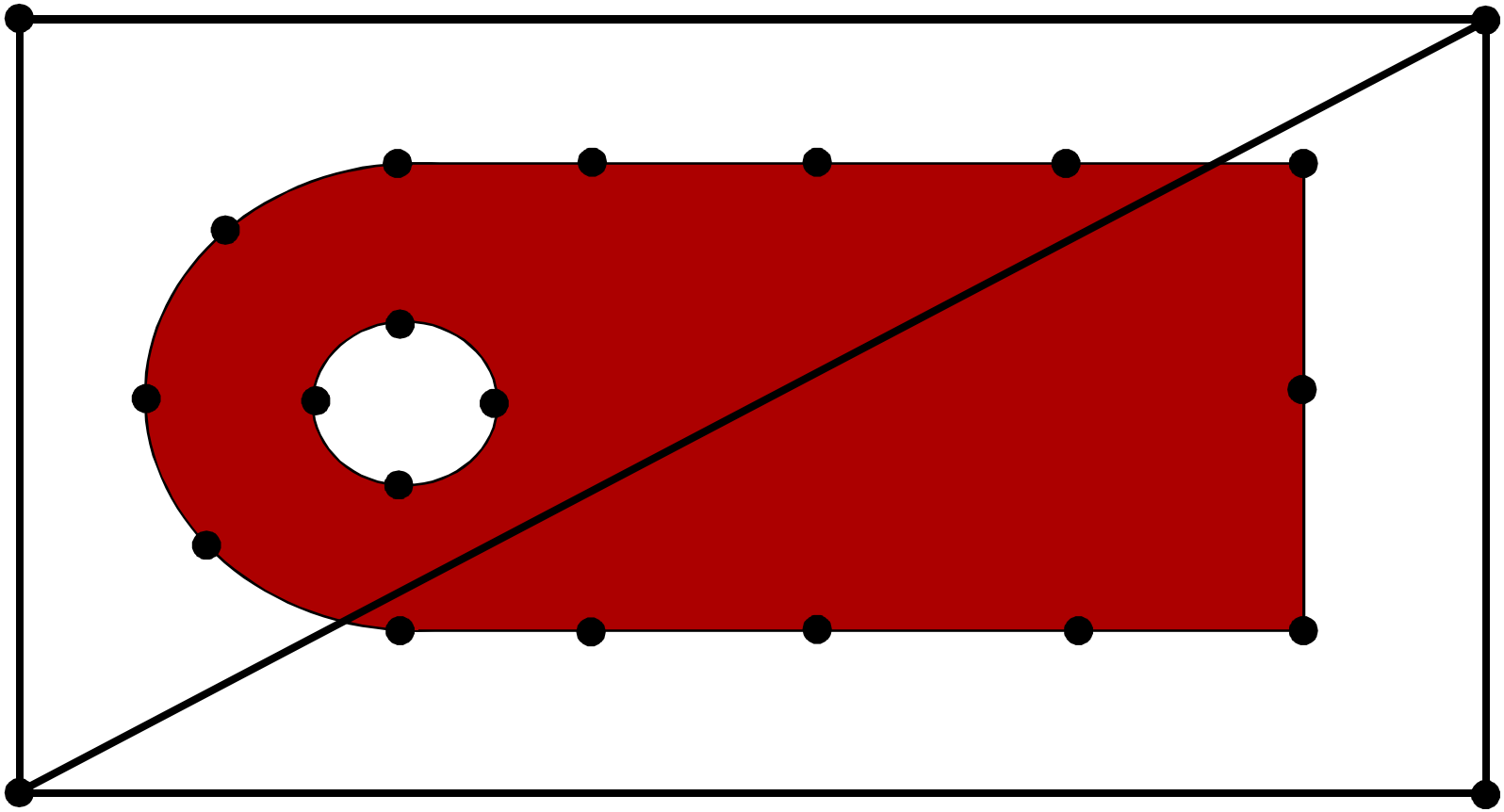


Delaunay



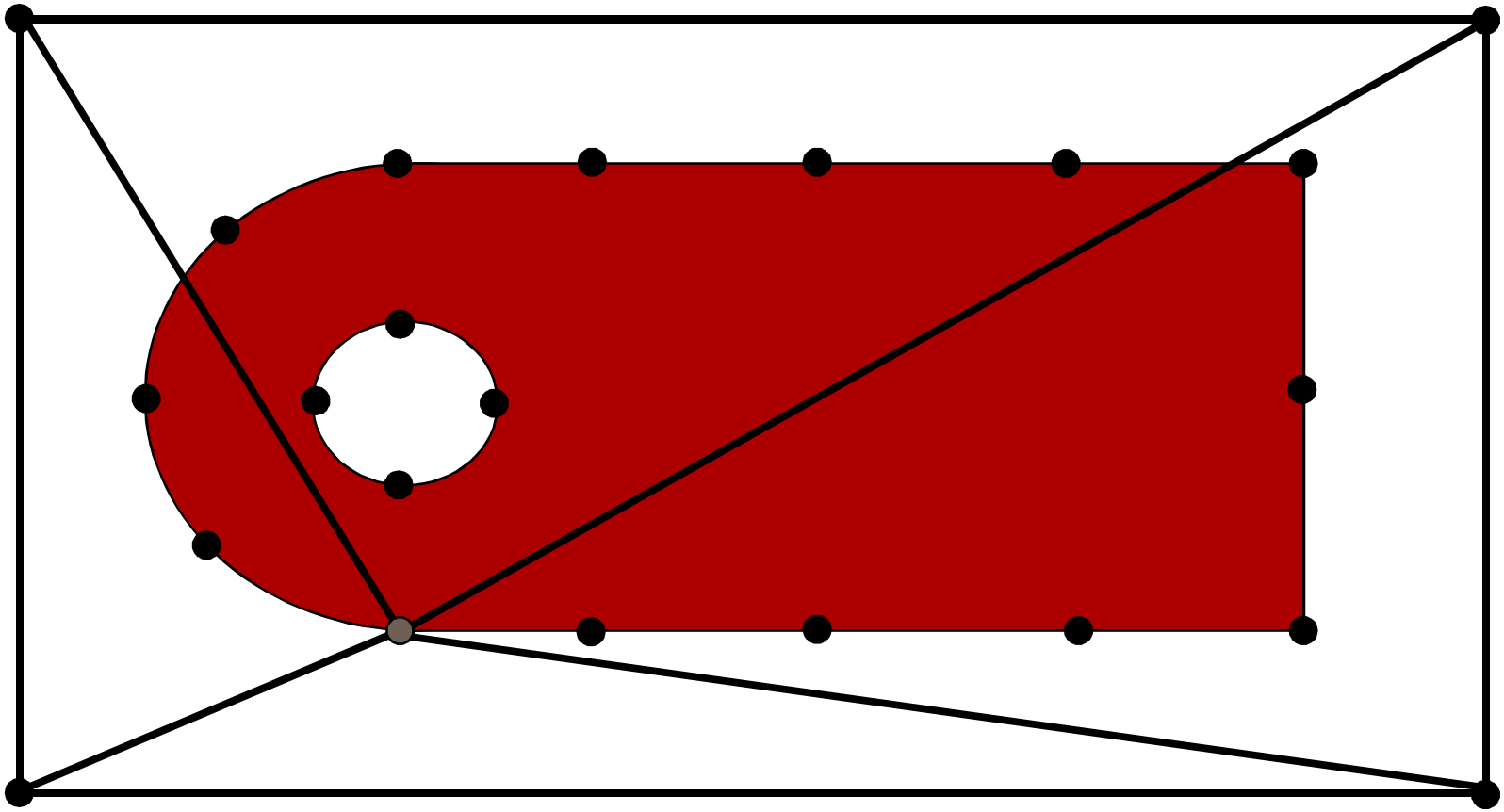
Delaunay

Sandia National Laboratories



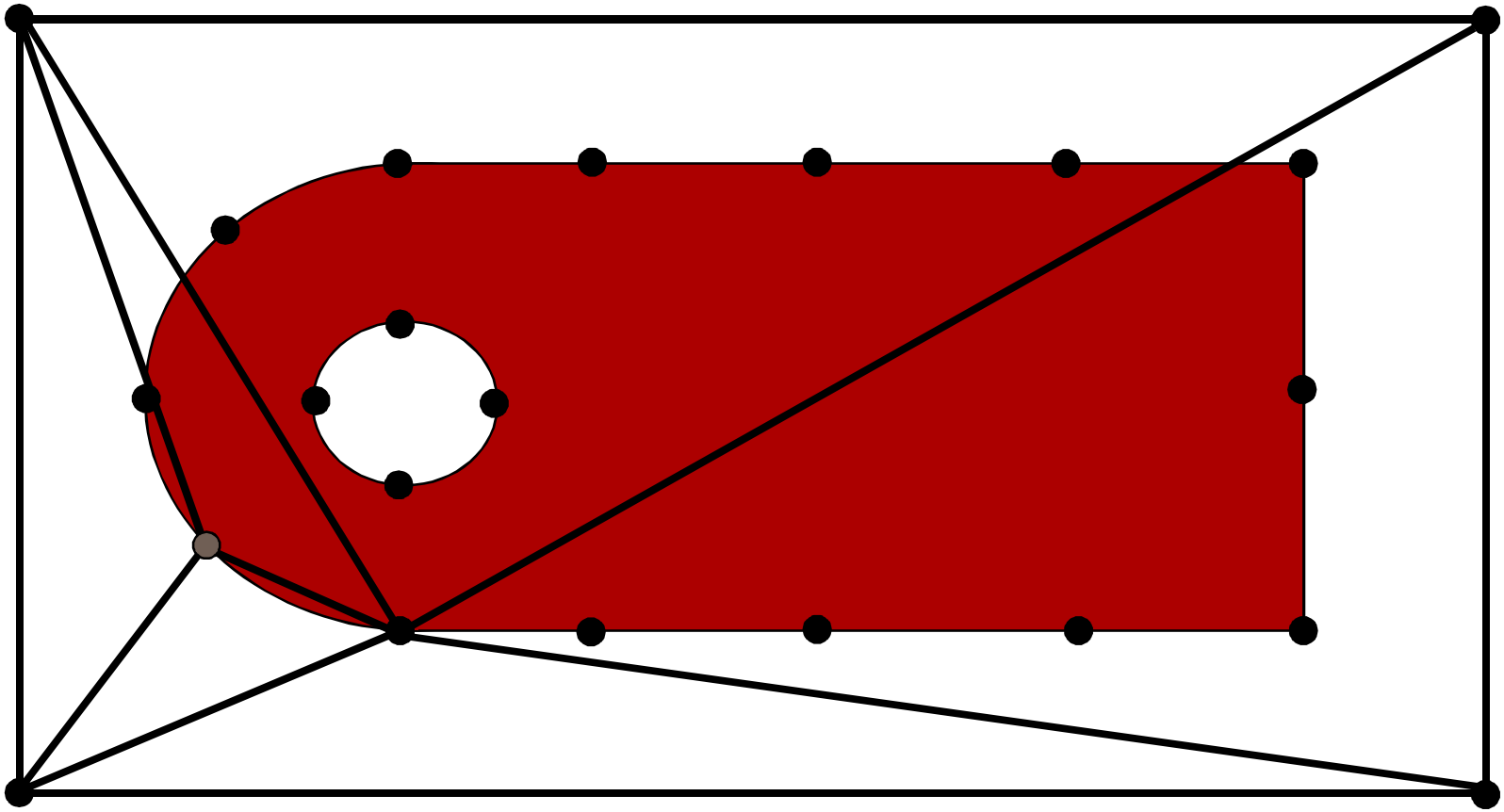
•Begin with Bounding Triangles (or Tetrahedra)

Delaunay



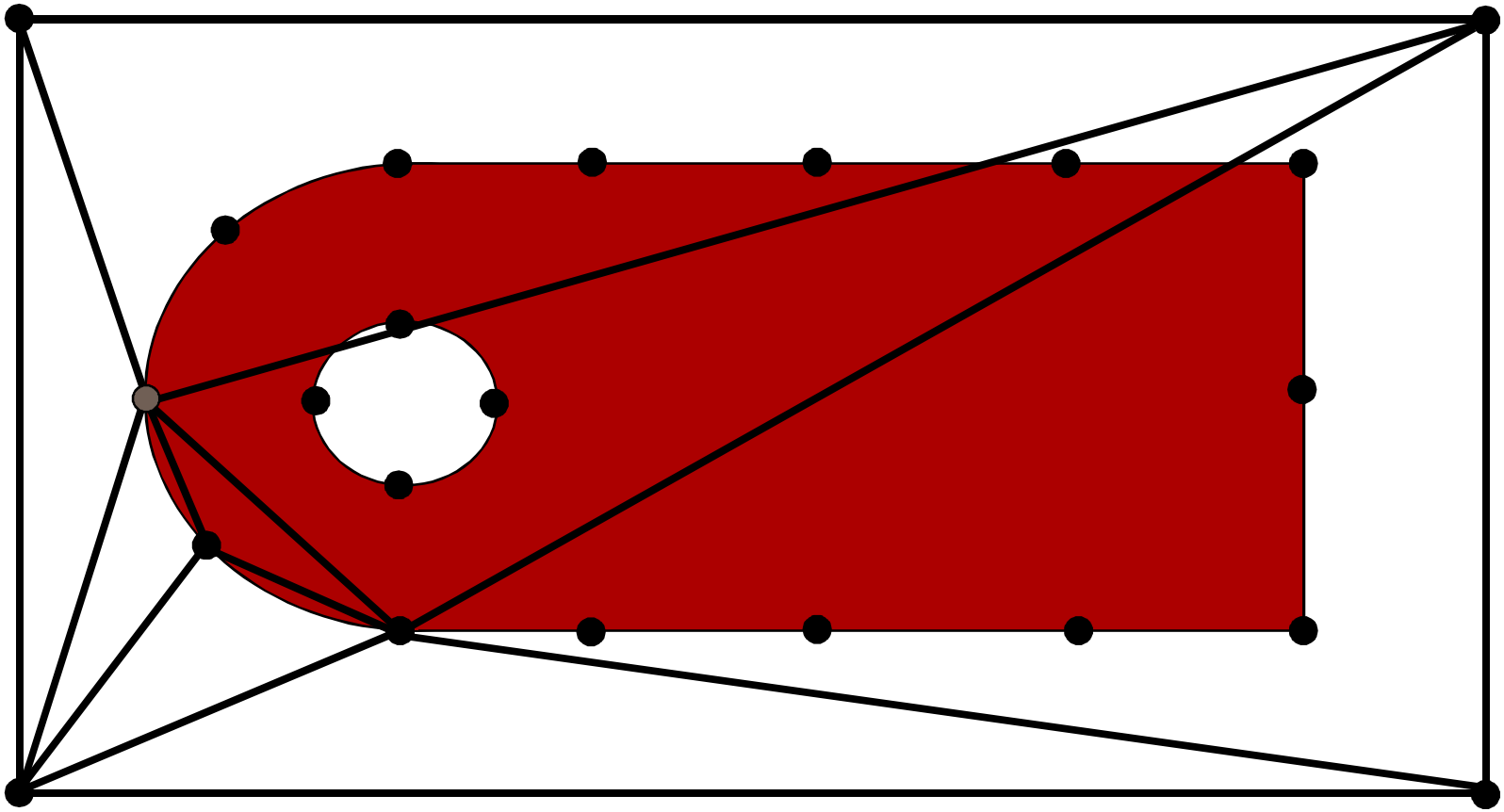
•Insert boundary nodes using Delaunay method (Lawson or Bowyer-Watson)

Delaunay



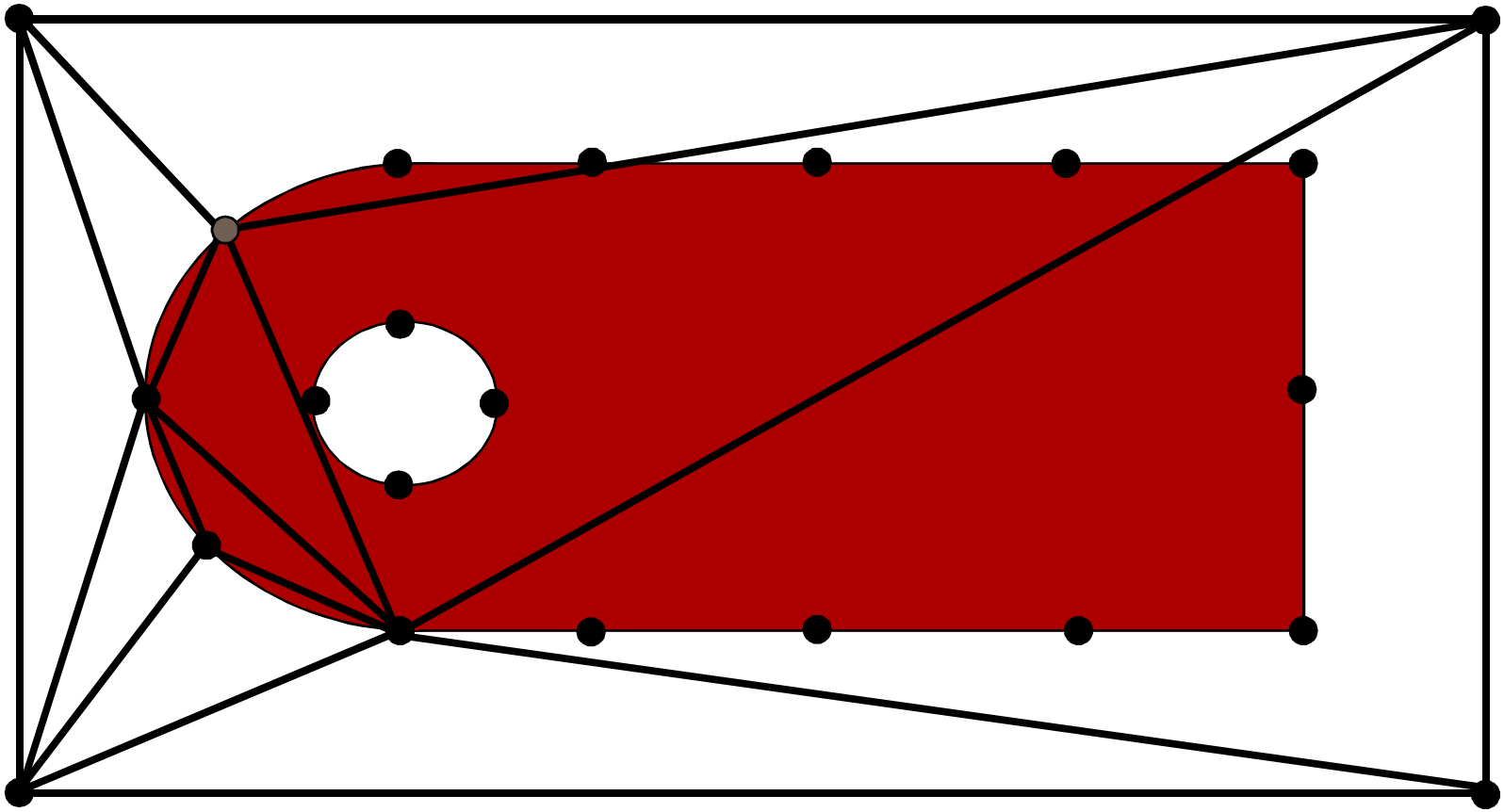
•Insert boundary nodes using Delaunay method (Lawson or Bowyer-Watson)

Delaunay



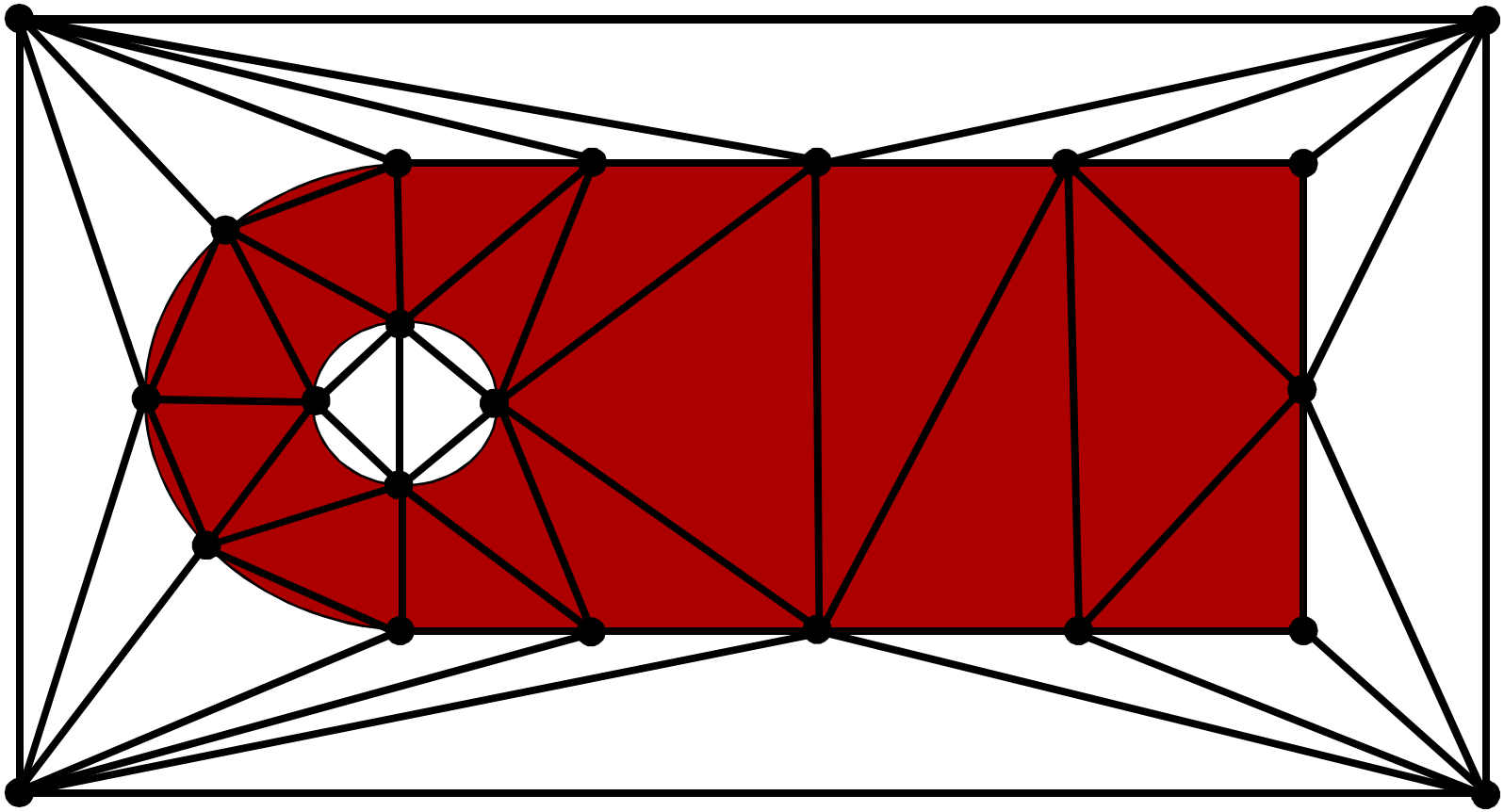
•Insert boundary nodes using Delaunay method (Lawson or Bowyer-Watson)

Delaunay



•Insert boundary nodes using Delaunay method (Lawson or Bowyer-Watson)

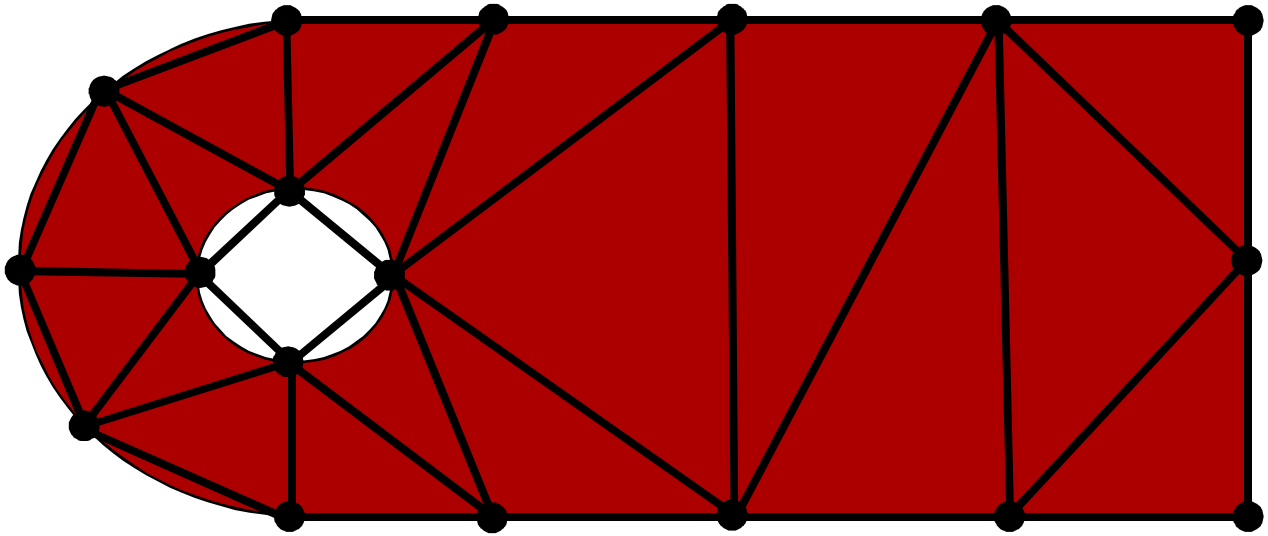
Delaunay



•Insert boundary nodes using Delaunay method (Lawson or Bowyer-Watson)

Delaunay

Sandia National Laboratories

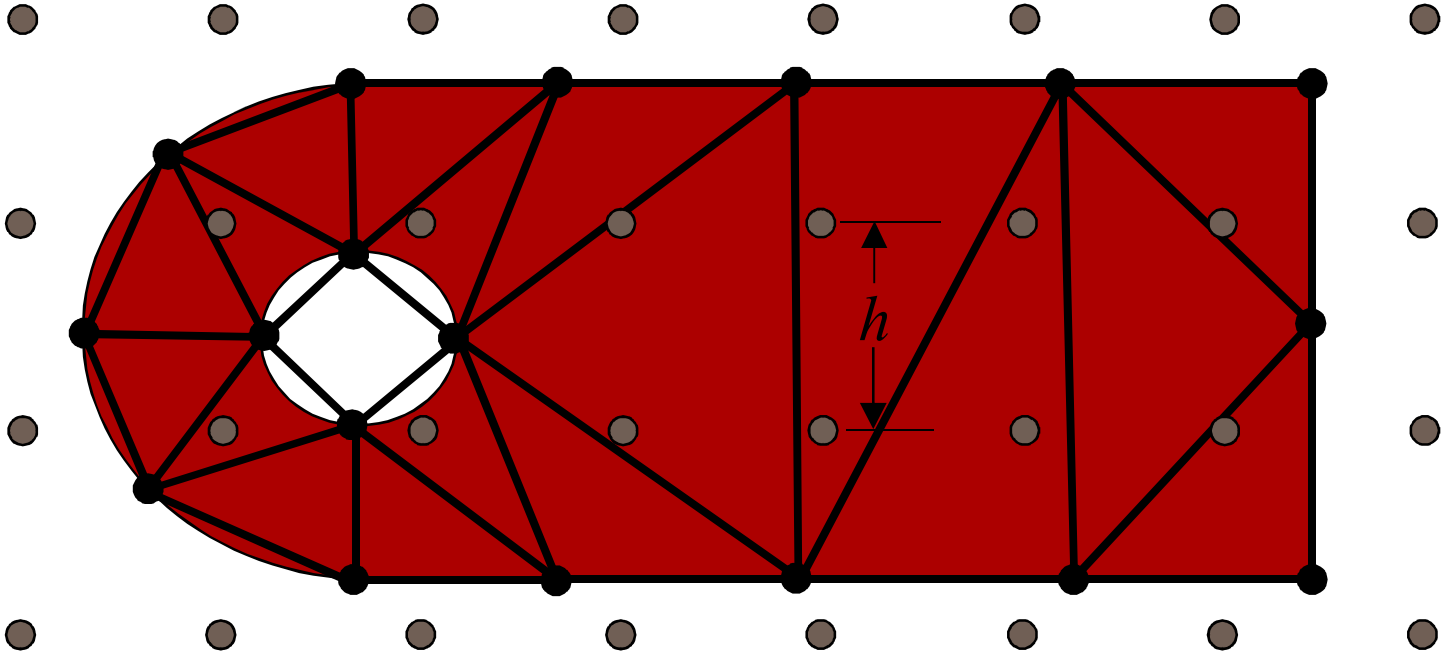


- Recover boundary
- Delete outside triangles
- Insert internal nodes

Delaunay

Sandia National Laboratories

Node Insertion Methods



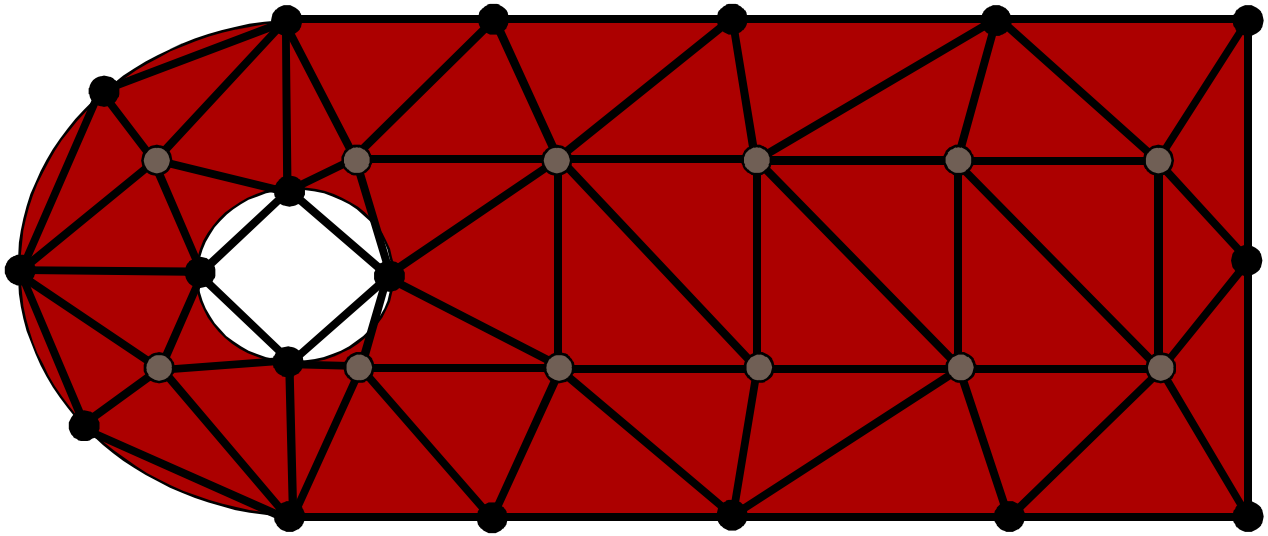
Grid Based

- Nodes introduced based on a regular lattice
- Lattice could be rectangular, triangular, quadtree, etc…
- Outside nodes ignored

Delaunay

Sandia National Laboratories

Node Insertion Methods



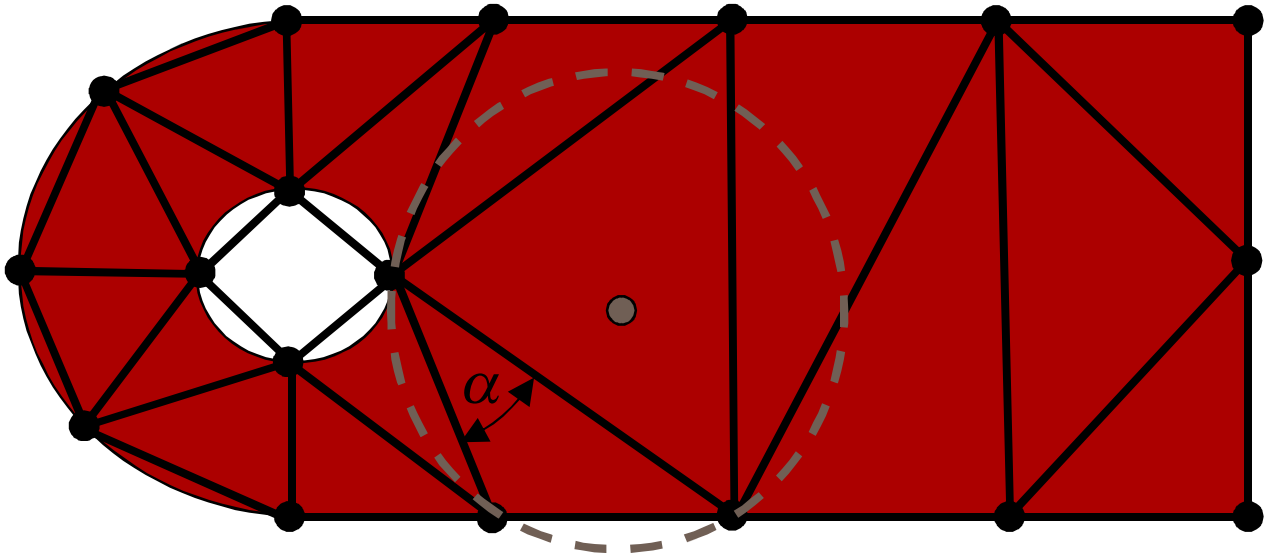
Grid Based

- Nodes introduced based on a regular lattice
- Lattice could be rectangular, triangular, quadtree, etc…
- Outside nodes ignored

Delaunay

Sandia National Laboratories

Node Insertion Methods



Circumcenter ("Guaranteed Quality")

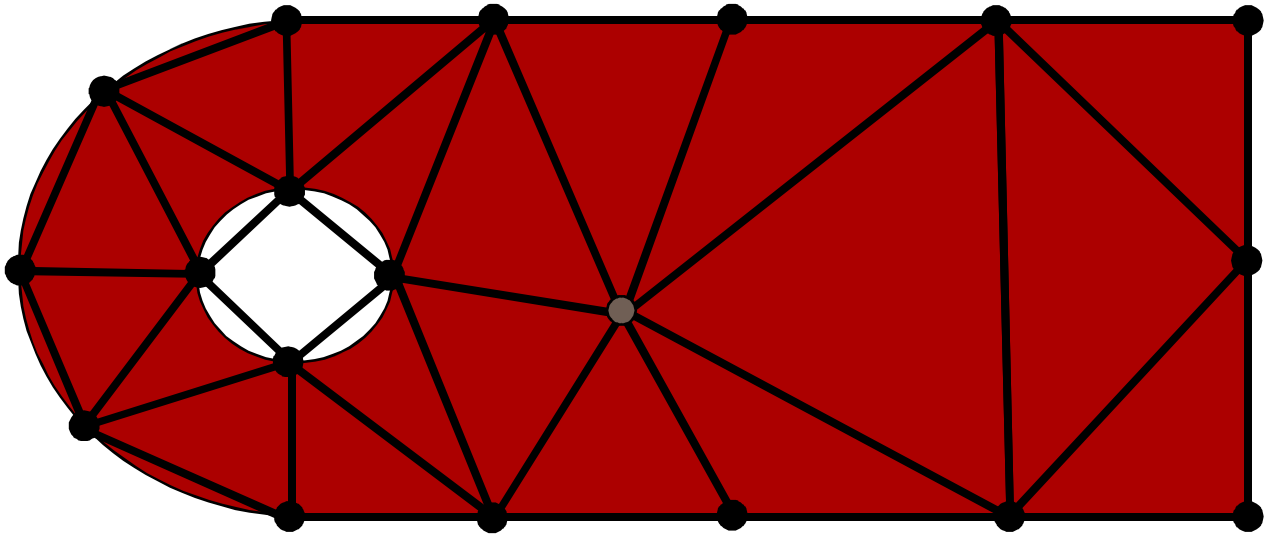
- Nodes introduced at triangle circumcenters
- Order of insertion based on minimum angle of any triangle
- Continues until minimum angle > predefined minimum $(\alpha \approx 30^\circ)$

(Chew,Ruppert,Shewchuk)

Delaunay

Sandia National Laboratories

Node Insertion Methods



Circumcenter ("Guaranteed Quality")

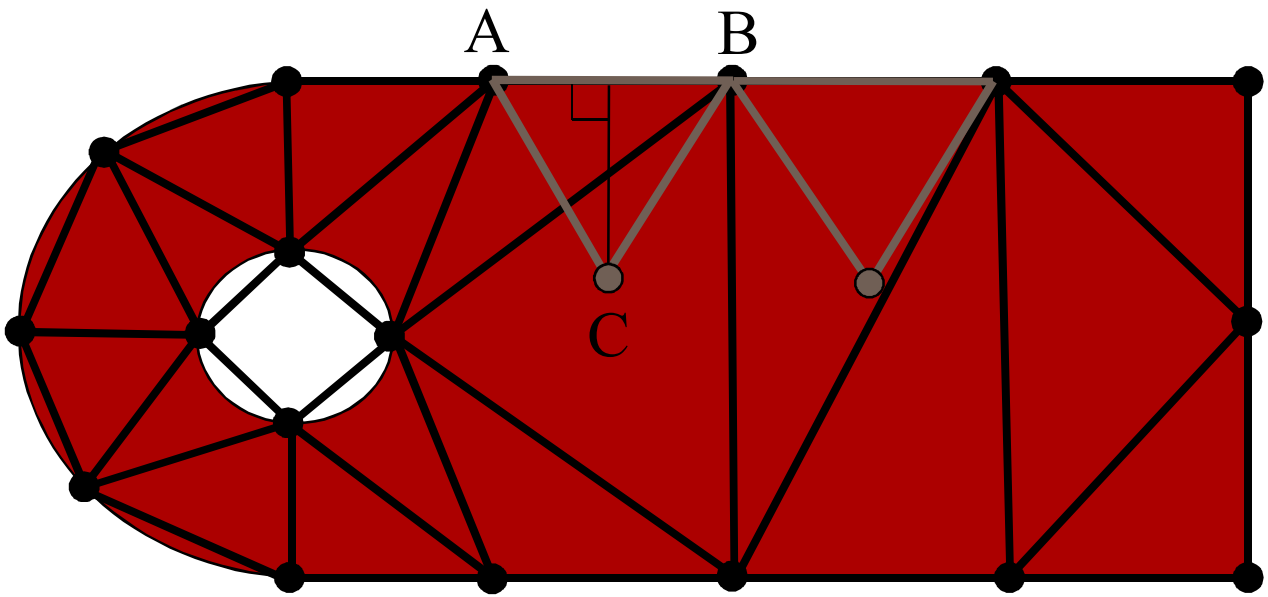
- Nodes introduced at triangle circumcenters
- Order of insertion based on minimum angle of any triangle
- Continues until minimum angle > predefined minimum $(\alpha \approx 30^\circ)$

(Chew,Ruppert,Shewchuk)

Delaunay

Sandia National Laboratories

Node Insertion Methods



Advancing Front

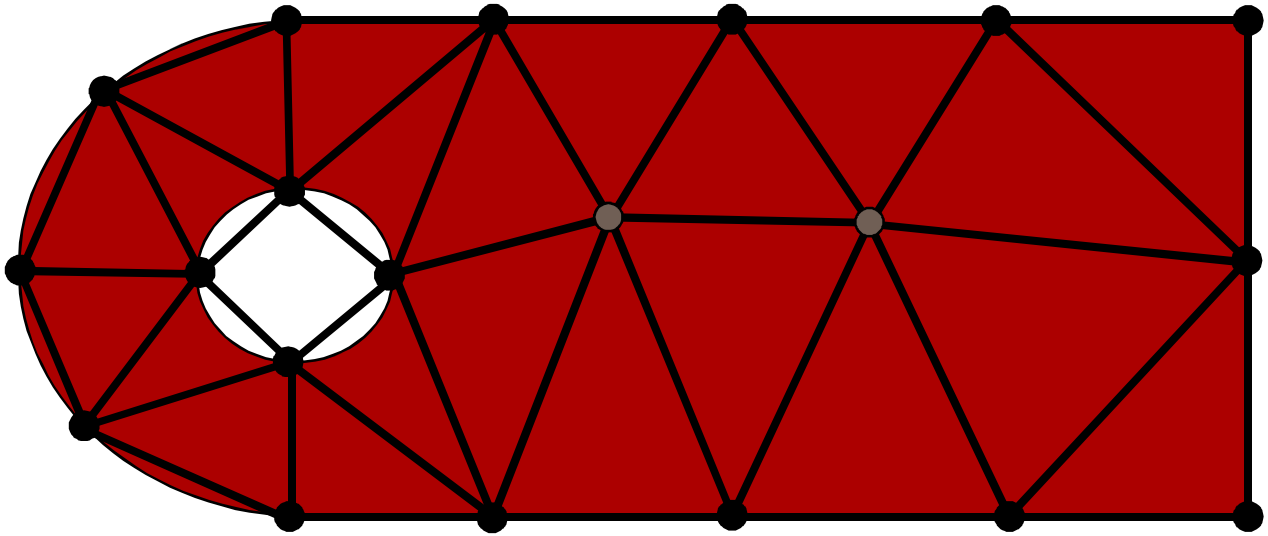
- "Front" structure maintained throughout
- Nodes introduced at ideal location from current front edge

(Marcum,95)

Delaunay

Sandia National Laboratories

Node Insertion Methods



Advancing Front

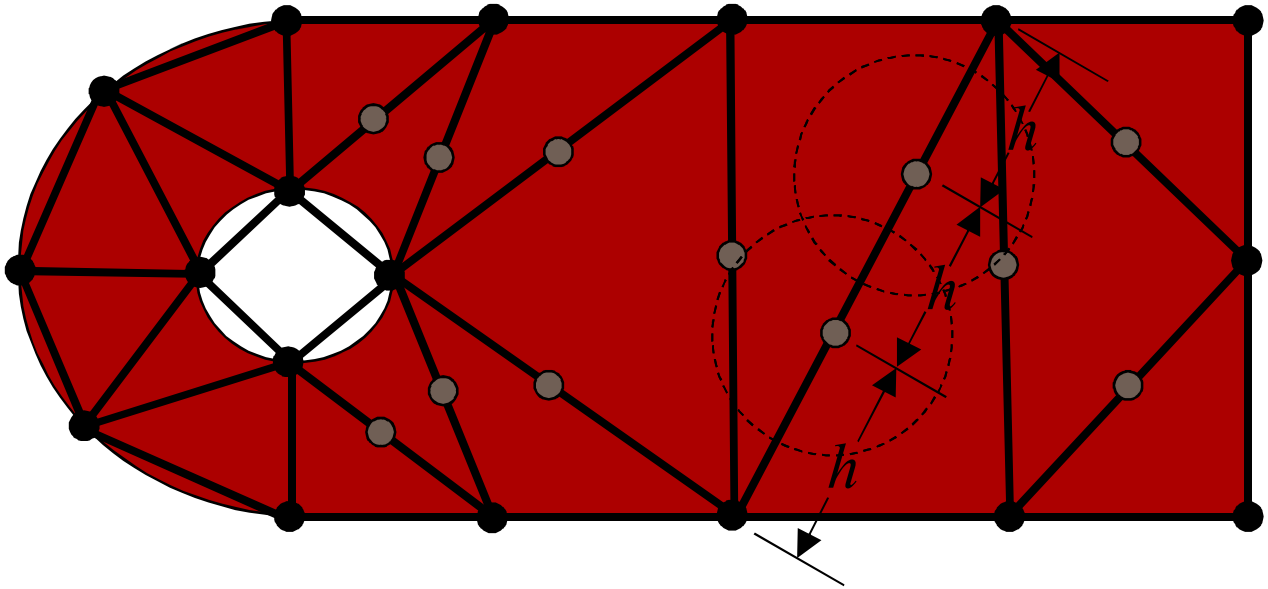
- "Front" structure maintained throughout
- Nodes introduced at ideal location from current front edge

(Marcum,95)

Delaunay

Sandia National Laboratories

Node Insertion Methods



Edges

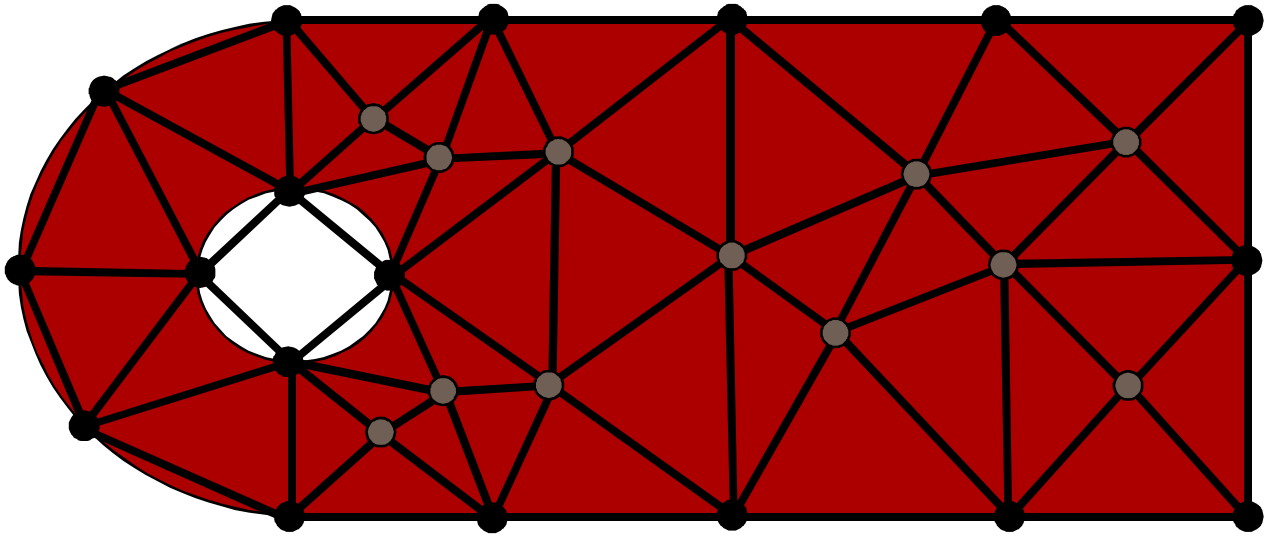
- Nodes introduced along existing edges at $l=h$
- Check to ensure nodes on nearby edges are not too close

(George,91)

Delaunay

Sandia National Laboratories

Node Insertion Methods



Edges

•Nodes introduced at along existing edges at $l=h$

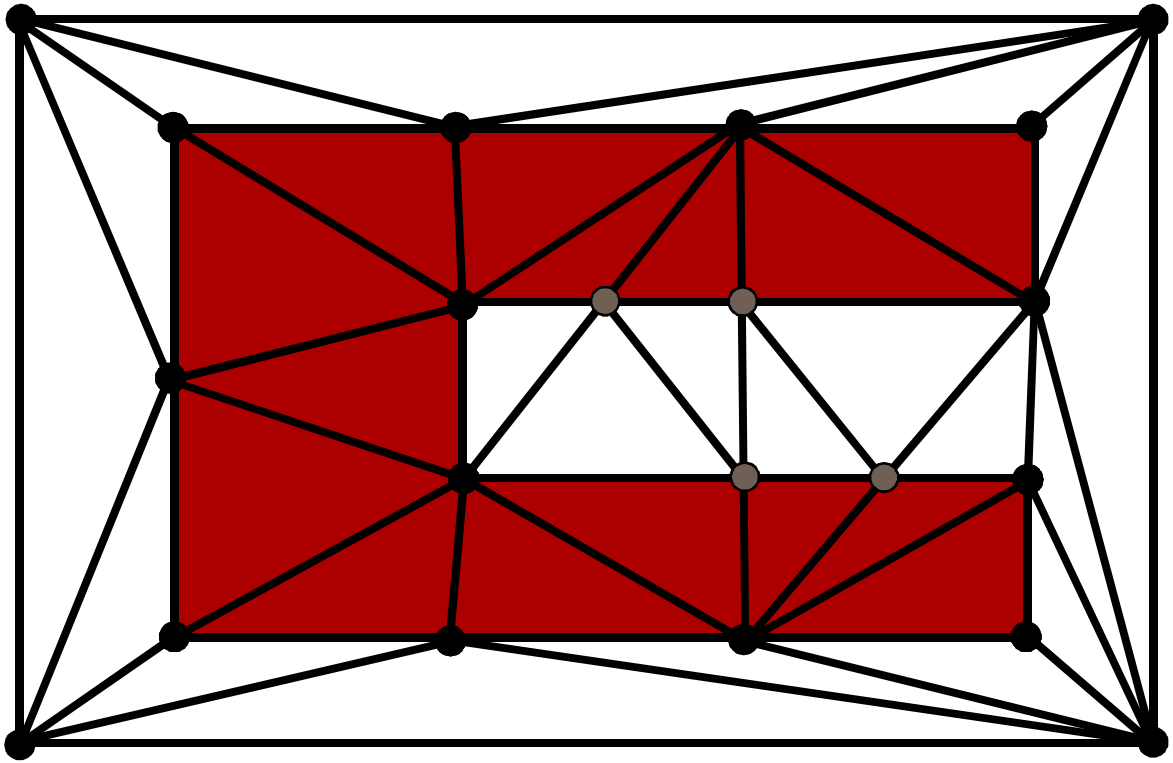
•Check to ensure nodes on nearby edges are not too close

(George,91)

Delaunay

Sandia National Laboratories

Boundary Constrained



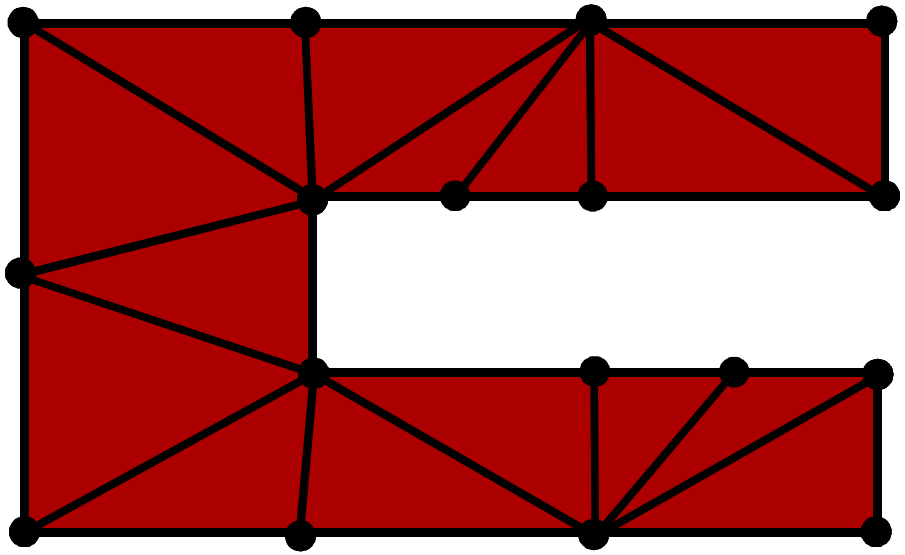
Boundary Intersection

•Nodes and edges introduced where Delaunay edges intersect boundary

Delaunay

Sandia National Laboratories

Boundary Constrained



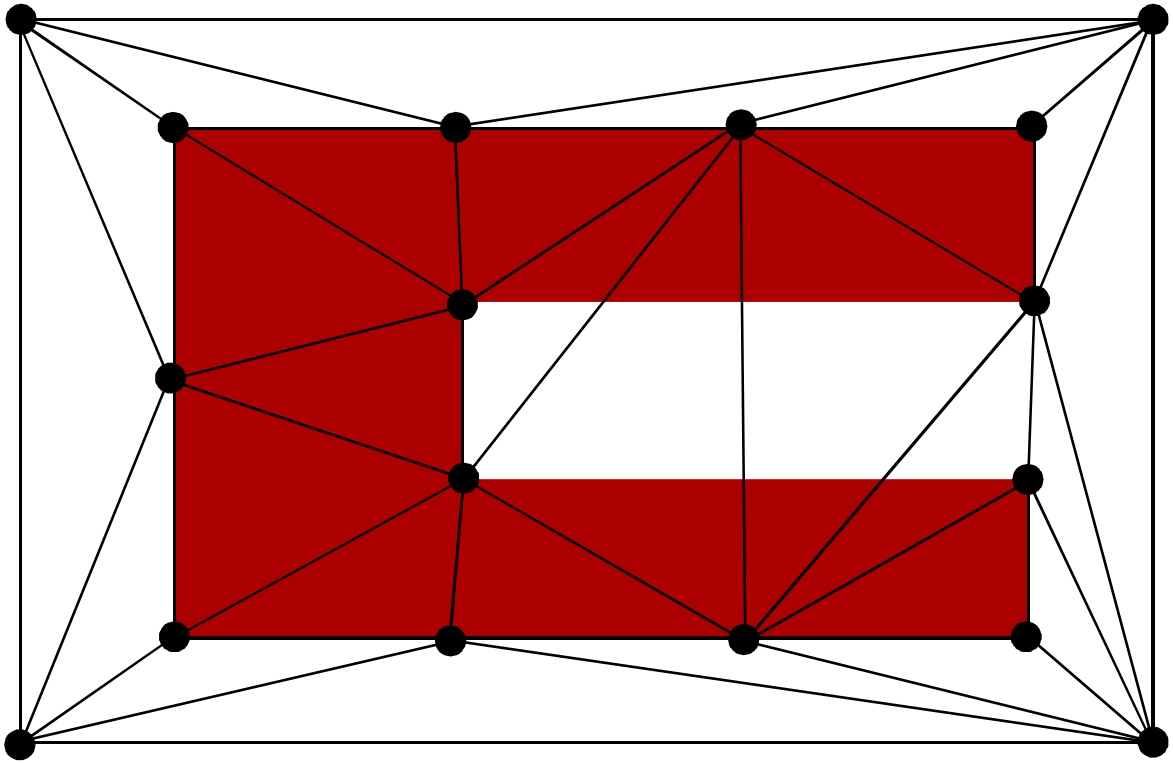
Boundary Intersection

•Nodes and edges introduced where Delaunay edges intersect boundary

Delaunay

Sandia National Laboratories

Boundary Constrained



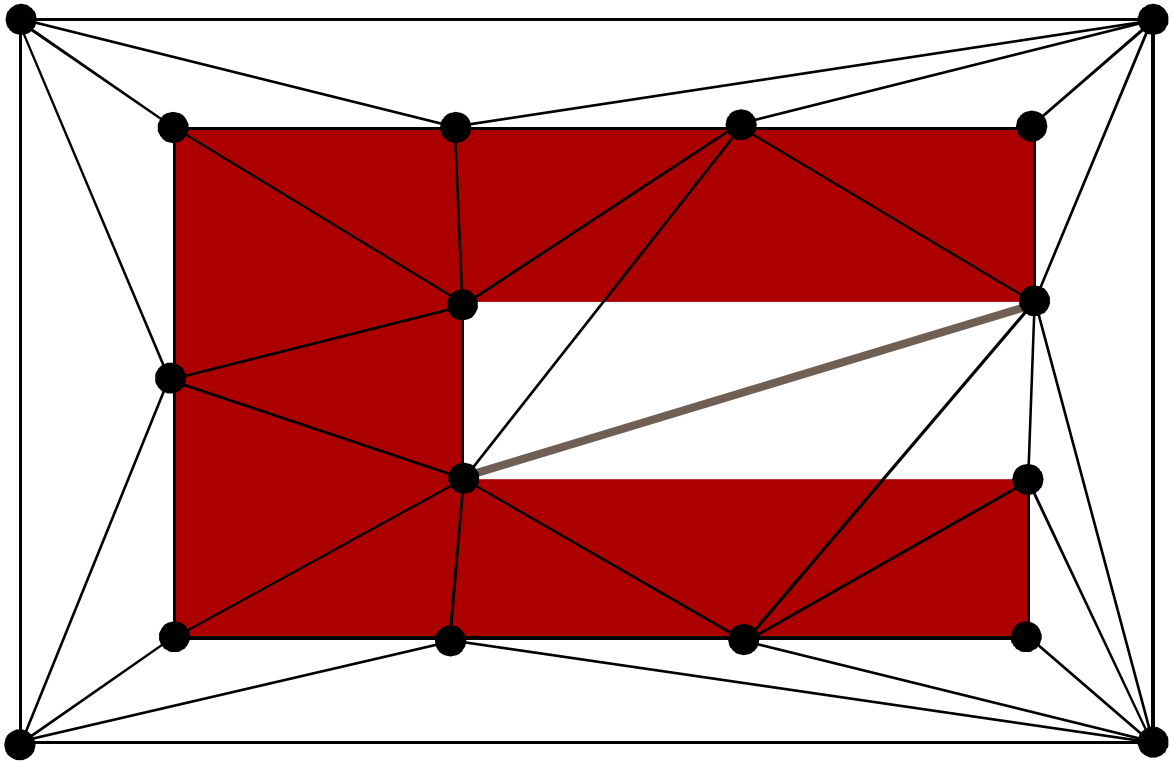
Local Swapping

•Edges swapped between adjacent pairs of triangles until boundary is maintained

Delaunay

Sandia National Laboratories

Boundary Constrained



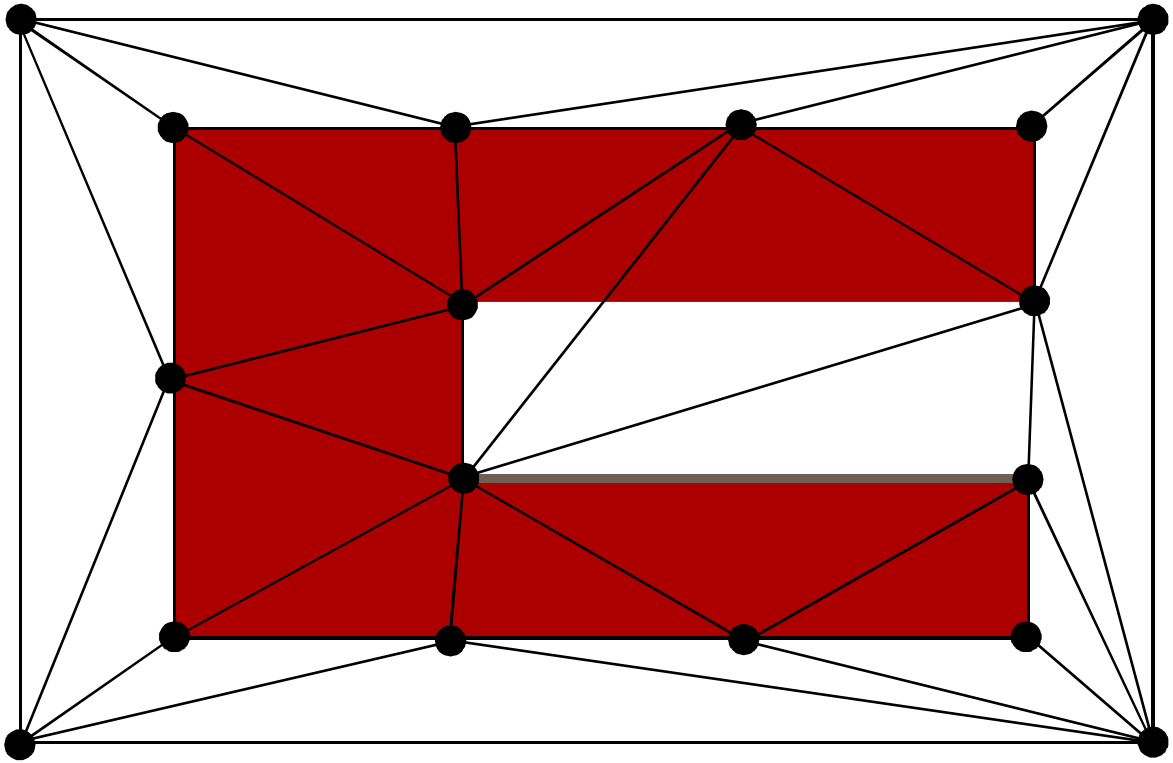
Local Swapping

•Edges swapped between adjacent pairs of triangles until boundary is maintained

Delaunay

Sandia National Laboratories

Boundary Constrained



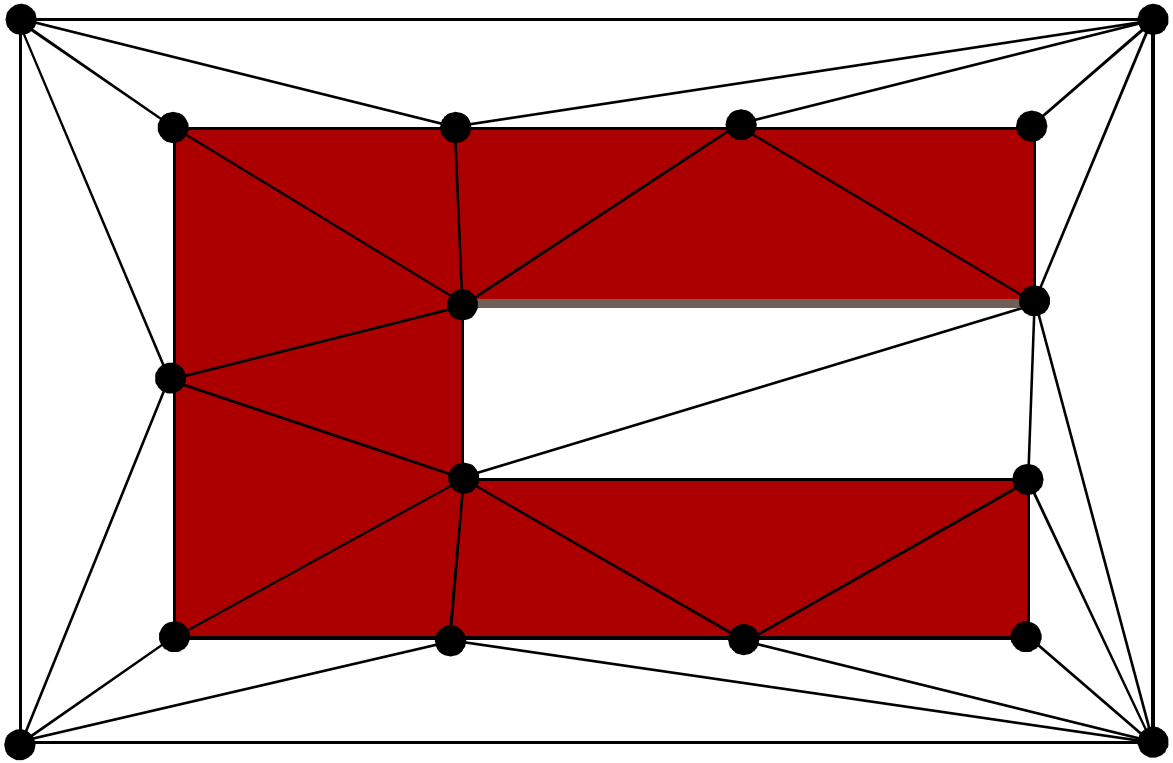
Local Swapping

•Edges swapped between adjacent pairs of triangles until boundary is maintained

Delaunay

Sandia National Laboratories

Boundary Constrained



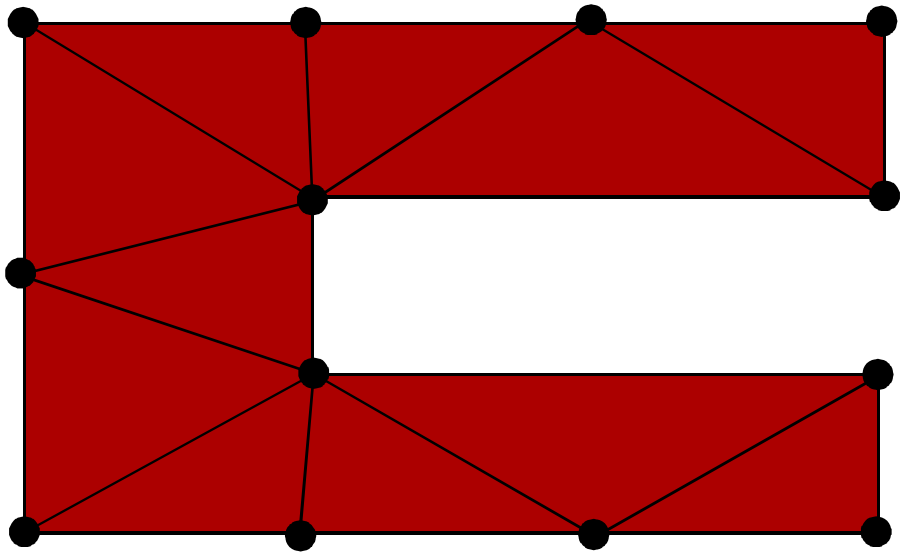
Local Swapping

•Edges swapped between adjacent pairs of triangles until boundary is maintained

Delaunay

Sandia National Laboratories

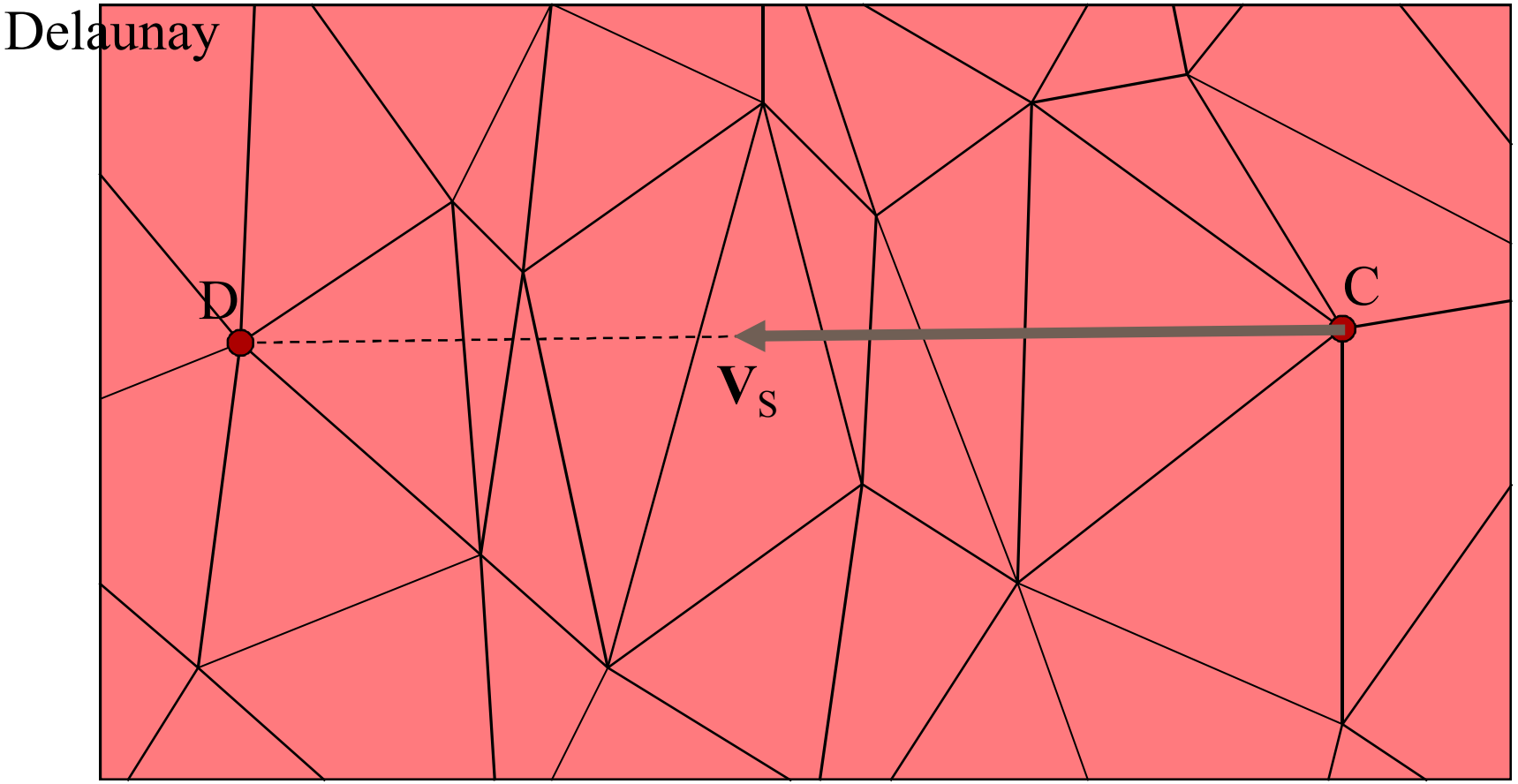
Boundary Constrained



Local Swapping

•Edges swapped between adjacent pairs of triangles until boundary is maintained

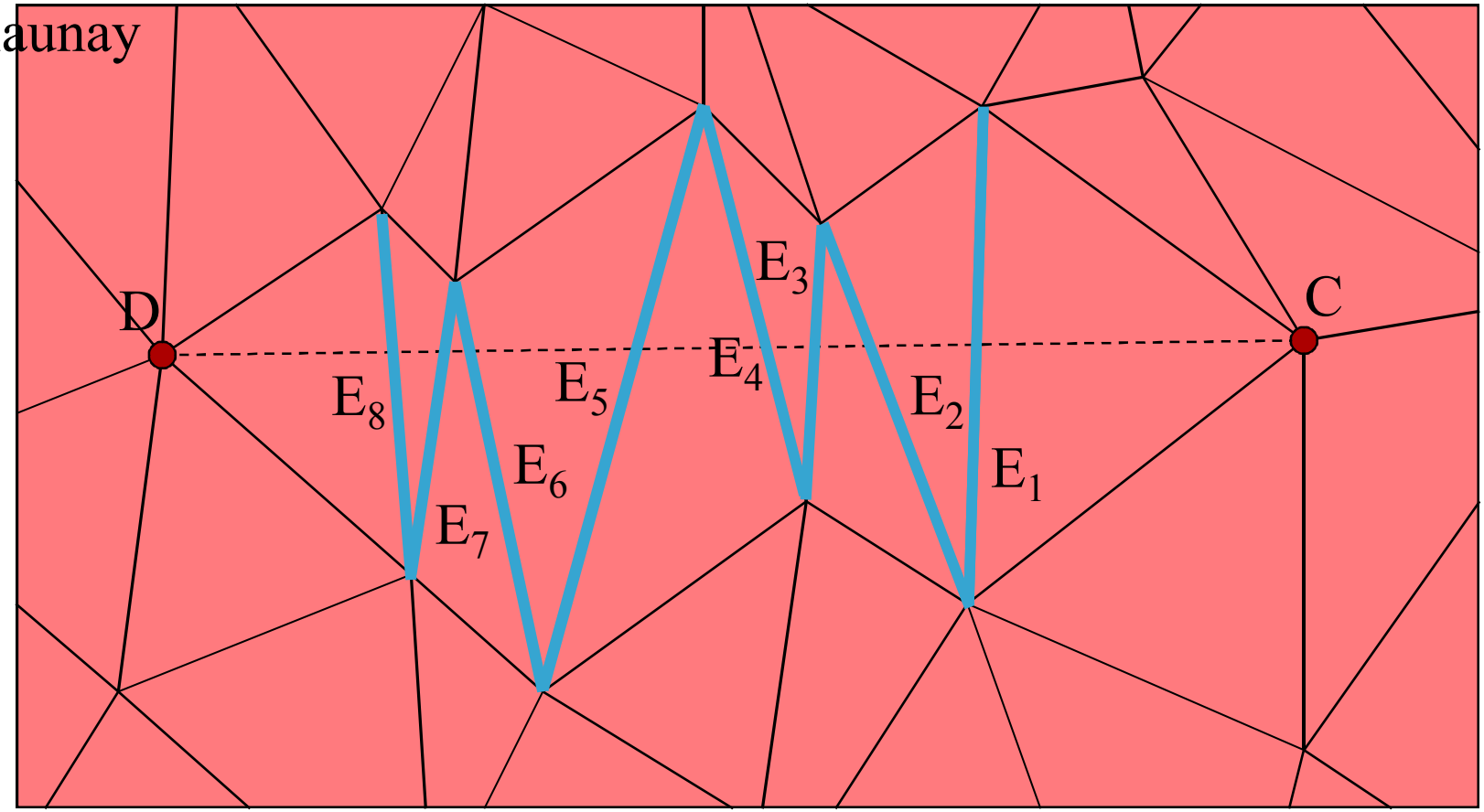
(George,91)(Owen,99)



Local Swapping Example

•Recover edge CD at vector $\mathbf{V}_s$

Boundary Constrained



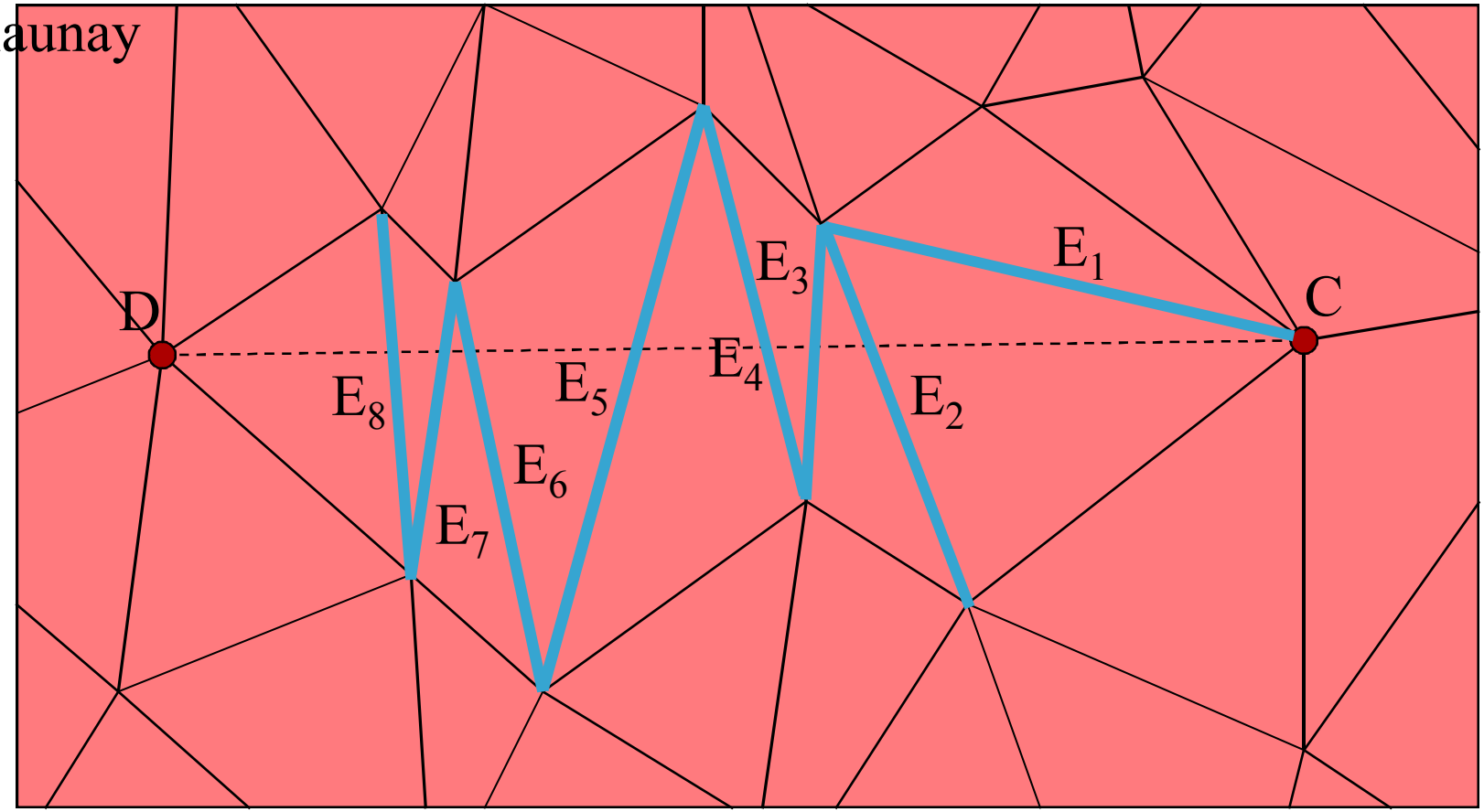
Local Swapping Example

•Make a list (queue) of all edges E_i, that intersect $\mathbf{V}_s$

Boundary Constrained

Sandia National Laboratories

Delaunay

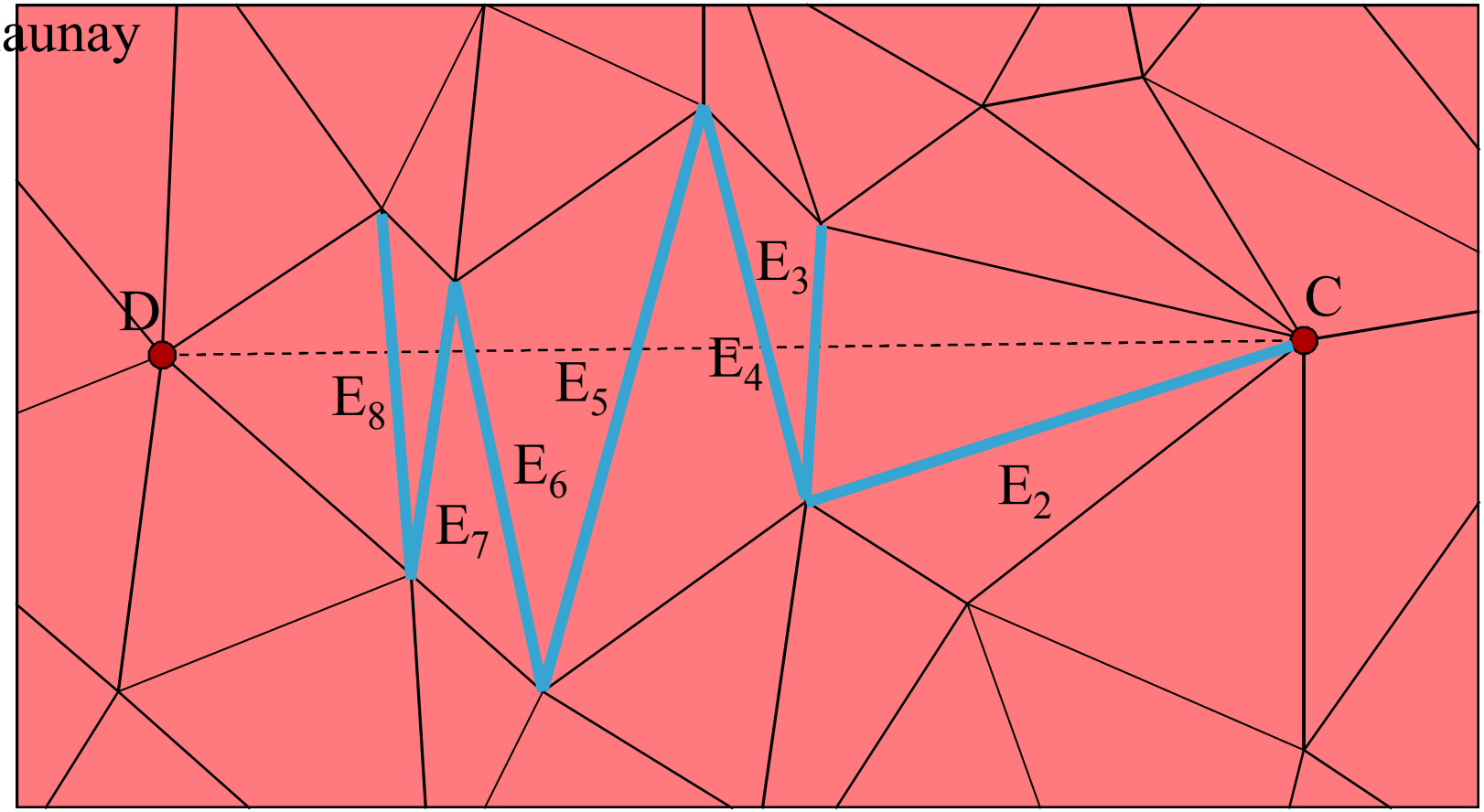


Local Swapping Example

•Swap the diagonal of adjacent triangle pairs for each edge in the list

Sandia National Laboratories

Delaunay

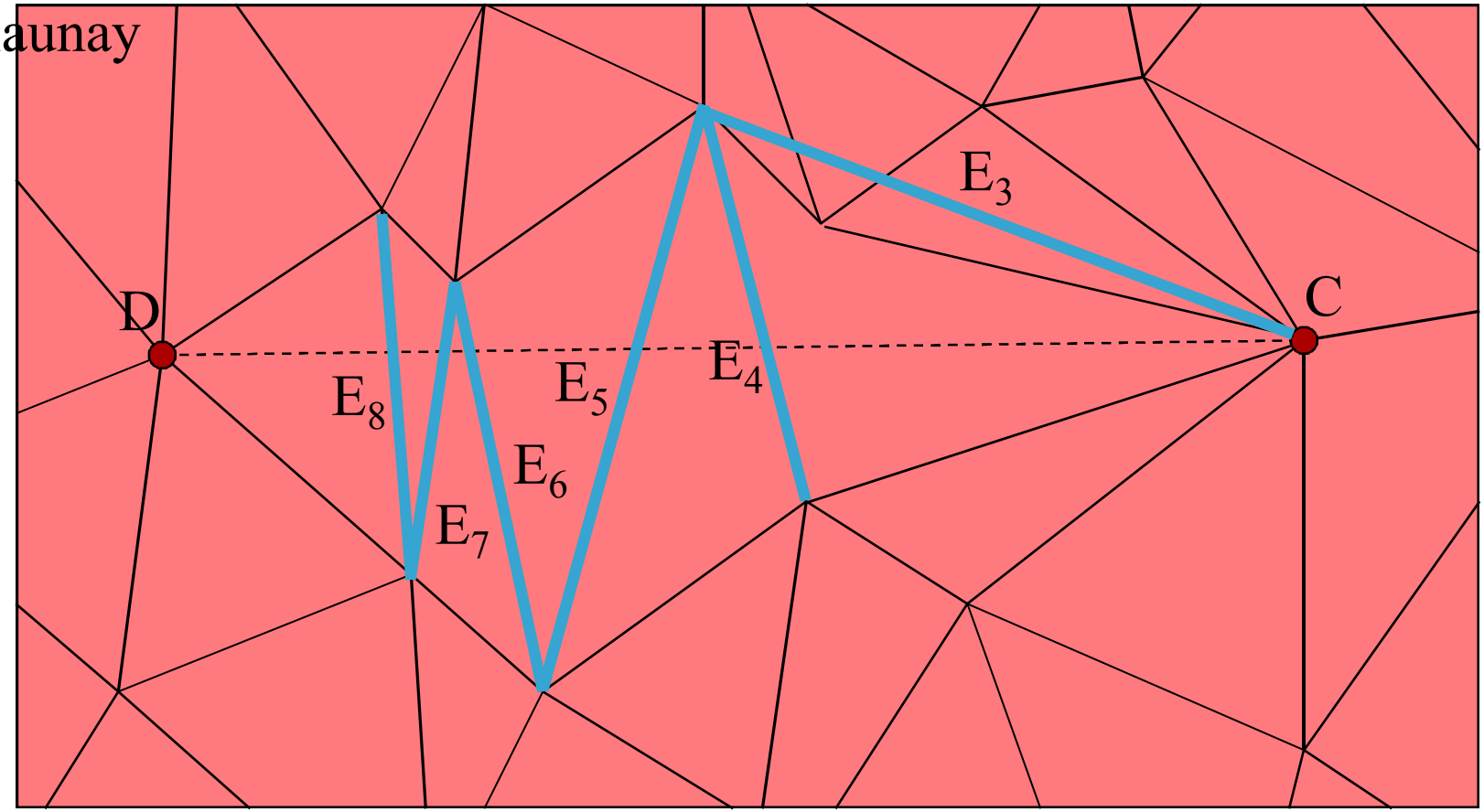


Local Swapping Example

•Check that resulting swaps do not cause overlapping triangles. If they do, then place edge at the back of the queue and try again later

Sandia National Laboratories

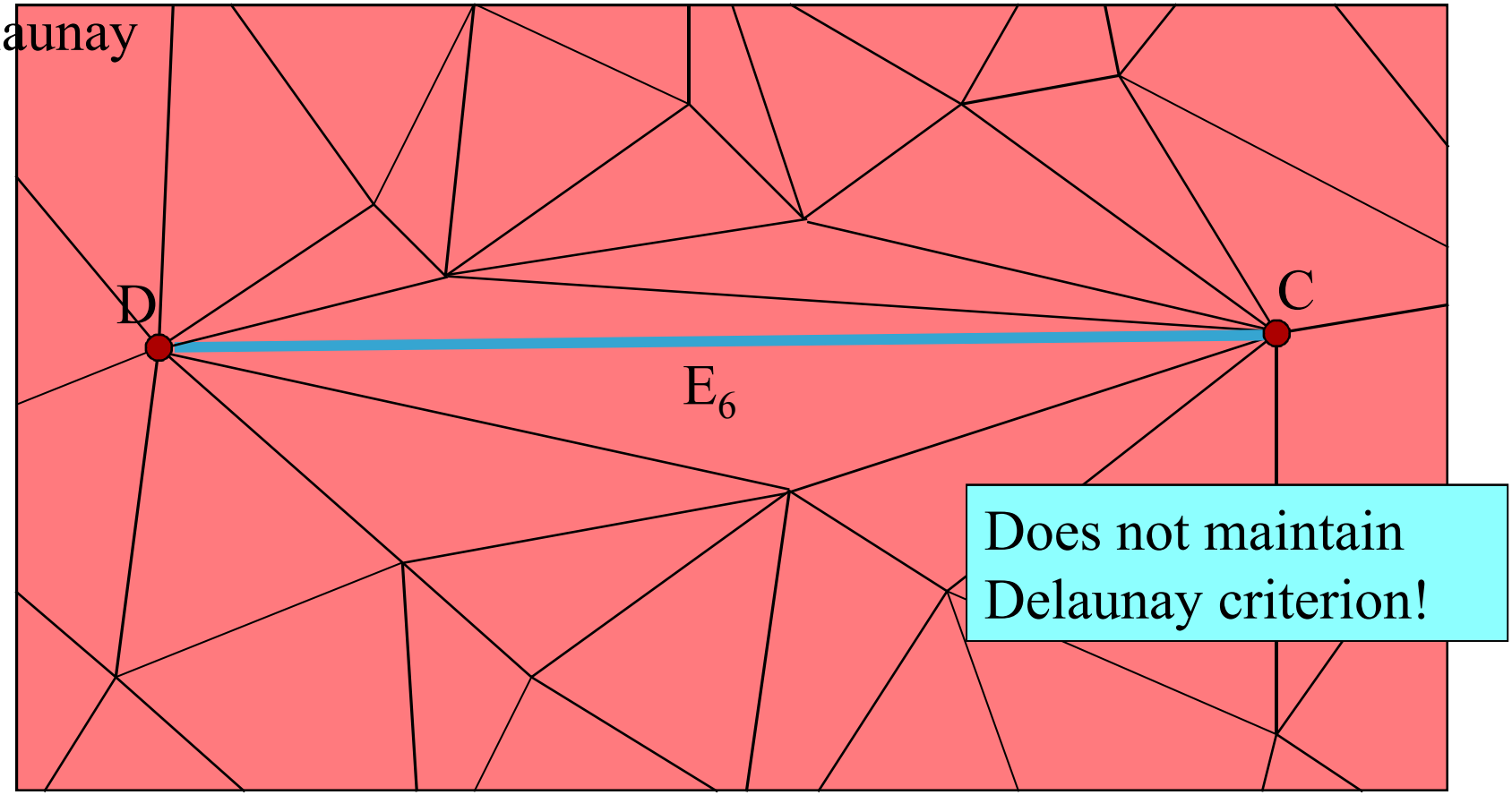
Delaunay



Local Swapping Example

•Check that resulting swaps do not cause overlapping triangles. If they do, then place edge at the back of the queue and try again later

Delaunay

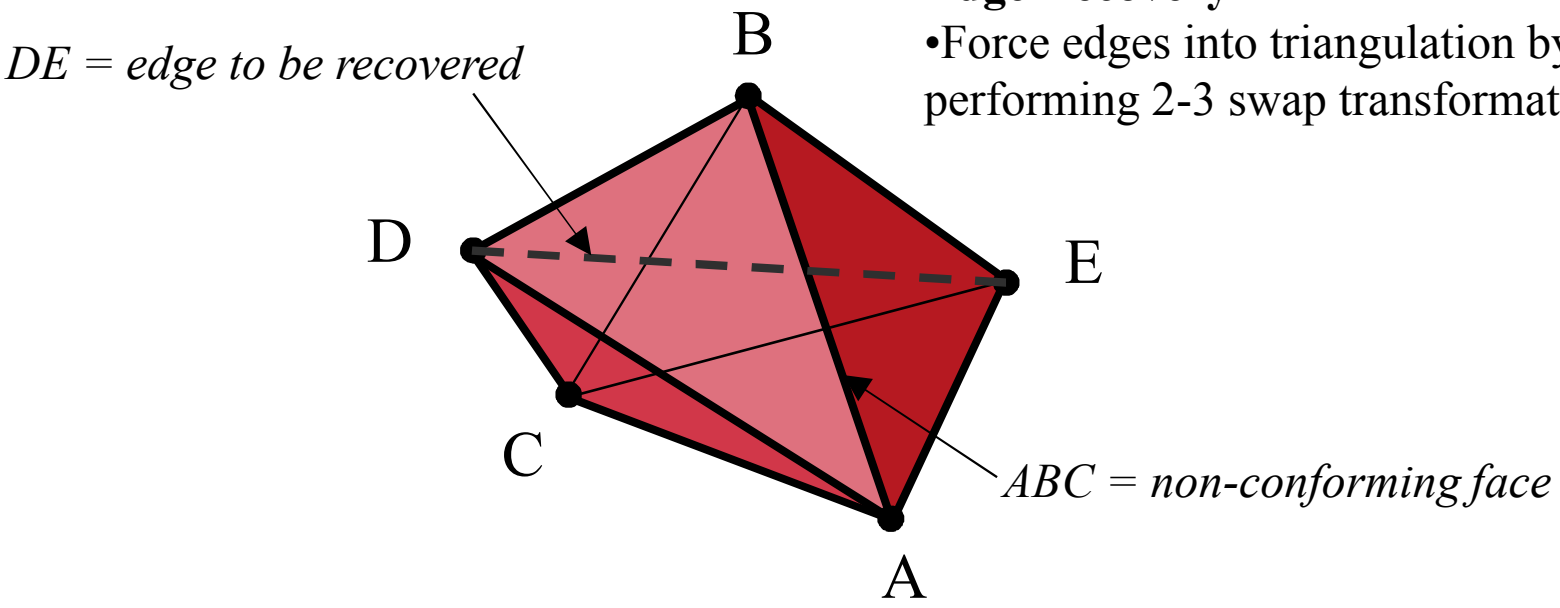


Local Swapping Example

- Final swap will recover the desired edge.
- Resulting triangle quality may be poor if multiple swaps were necessary

Delaunay

Boundary Constrained



Edge Recovery

•Force edges into triangulation by performing 2-3 swap transformation

3D Local Swapping

•Requires both boundary *edge* recovery and boundary *face* recovery

(George,91;99)(Owen,00)

Delaunay

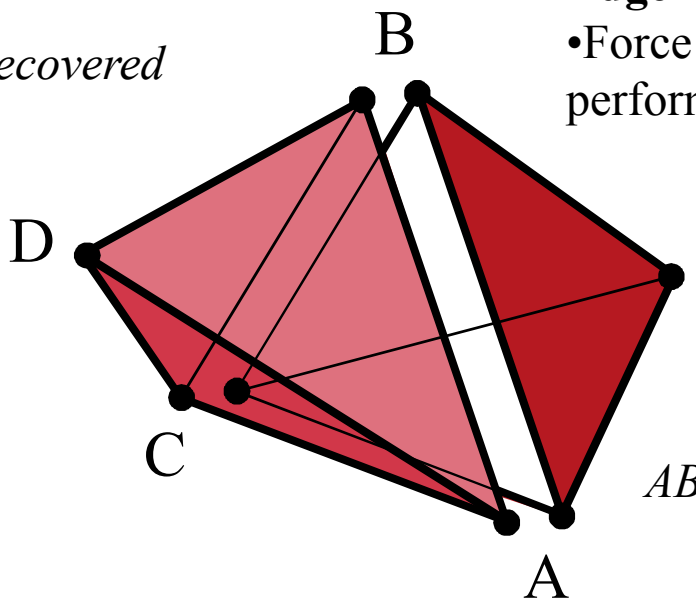
Sandia National Laboratories

Boundary Constrained

DE = edge to be recovered

2-3 Swap

ABCE
ACBD



ABC = non-conforming face

Edge Recovery
•Force edges into triangulation by performing 2-3 swap transformation

3D Local Swapping
•Requires both boundary *edge* recovery and boundary *face* recovery

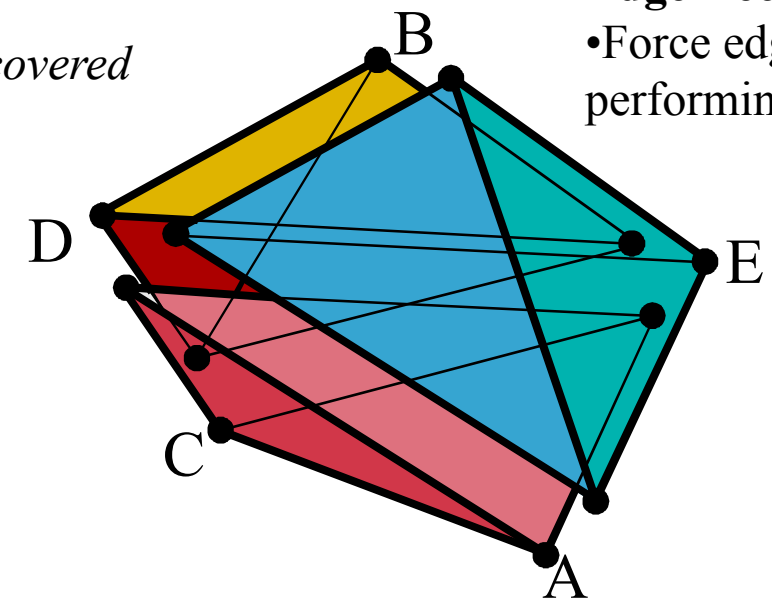
(George,91;99)(Owen,00)

Delaunay

Sandia National Laboratories

Boundary Constrained

DE = edge recovered



Edge Recovery

•Force edges into triangulation by performing 2-3 swap transformation

2-3 Swap

ABCE
ACBD
⇨
BAED
CBED
ACED

3D Local Swapping

•Requires both boundary *edge* recovery and boundary *face* recovery

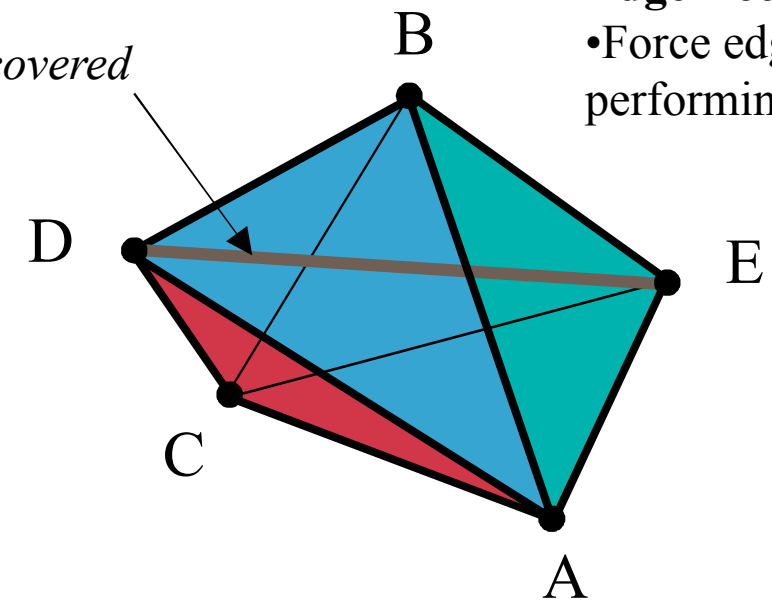
(George,91;99)(Owen,00)

Delaunay

Sandia National Laboratories

Boundary Constrained

DE = edge recovered



Edge Recovery

•Force edges into triangulation by performing 2-3 swap transformation

2-3 Swap

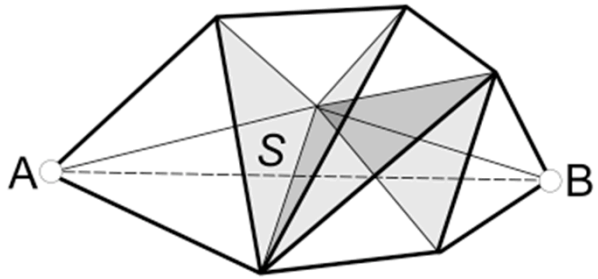
ABCE
ACBD
⇨
BAED
CBED
ECED

3D Local Swapping

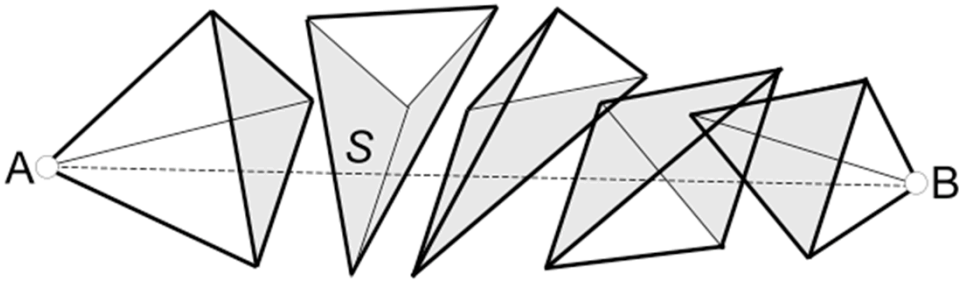
•Requires both boundary *edge* recovery and boundary *face* recovery

(George,91;99)(Owen,00)

Delaunay



Edge AB to be recovered

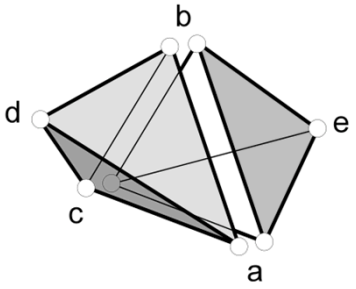
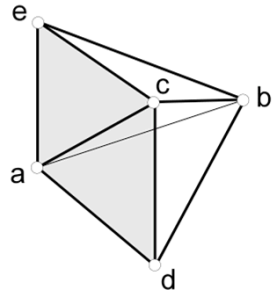
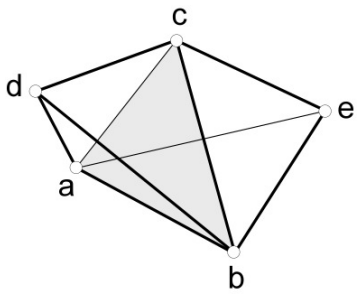
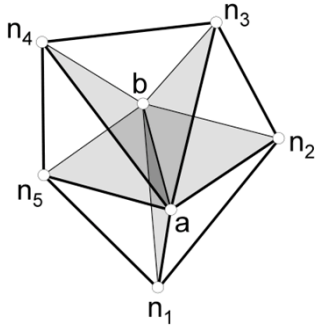
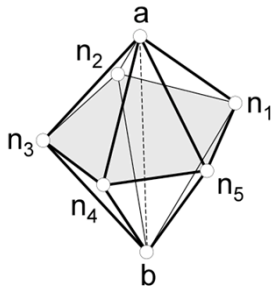
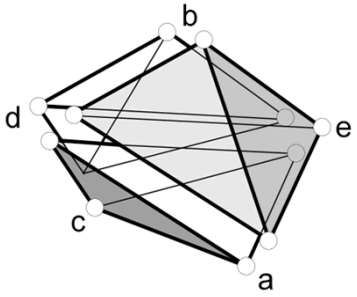
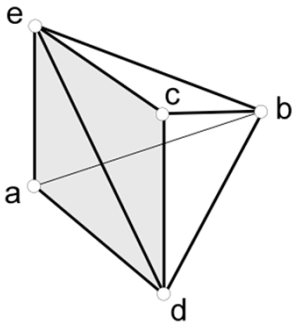
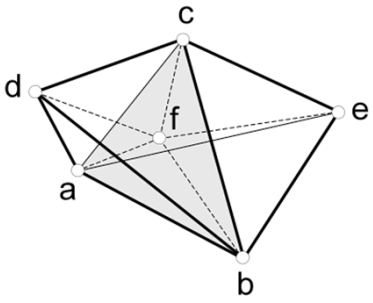
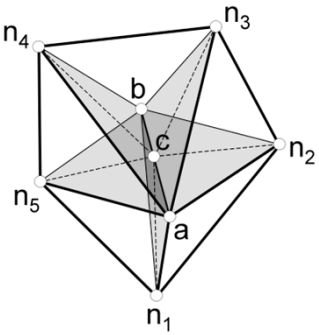
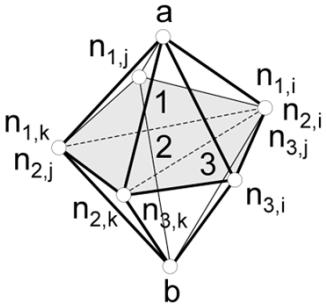


Exploded view of tets intersected by AB

3D Edge Recovery

- Form queue of faces through which edge AB will pass
- Perform 2-3 swap transformations on all faces in the list
- If overlapping tets result, place back on queue and try again later
- If still cannot recover edge, then insert "steiner" point

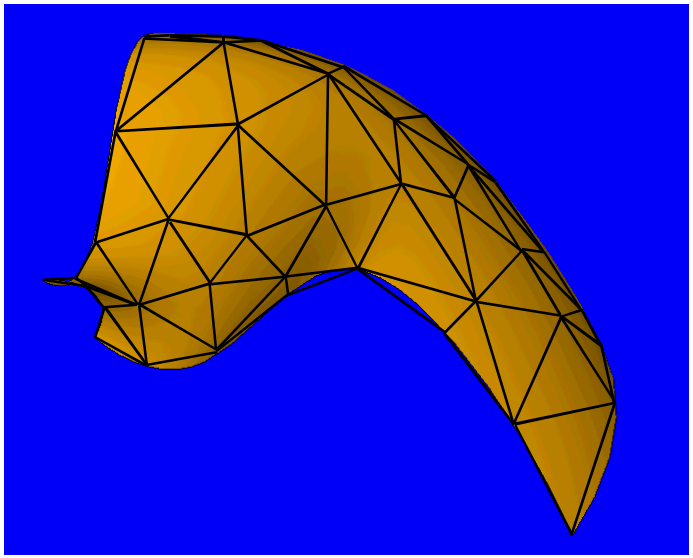
Delaunay

2-3 Swap	*2-2 Swap*	*Face Split*	*Edge Split*	*Edge Suppress*
				
abce, acbd	aceb, adcb	abce, acbd	abn_in_{i+1} {i=1…N}	abn_in_{i+1} {i=1…N}
				
abde, bcde, cade	adeb, edcb	abfe, bcfe, cafe bafd, cbfd, acfd	abn_in_{i+1}, cbn_in_{i+1} {i=1…N} N = no. adj. tets at edge ab	$n_{m,k}n_{m.j}n_{m.i}a$, $n_{m,k}n_{m.j}n_{m.i}b$ {m=1…M} M = no. unique trias in polygon $P=\{n_1,n_2,n_3\}$

Surface Meshing

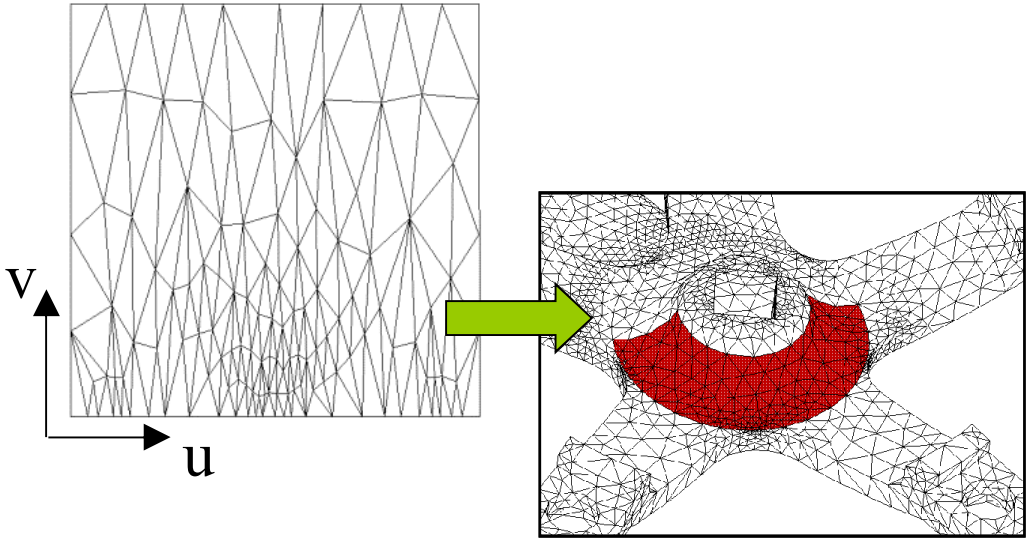
Sandia National Laboratories

Direct 3D Meshing



•Elements formed in 3D using actual x-y-z representation of surface

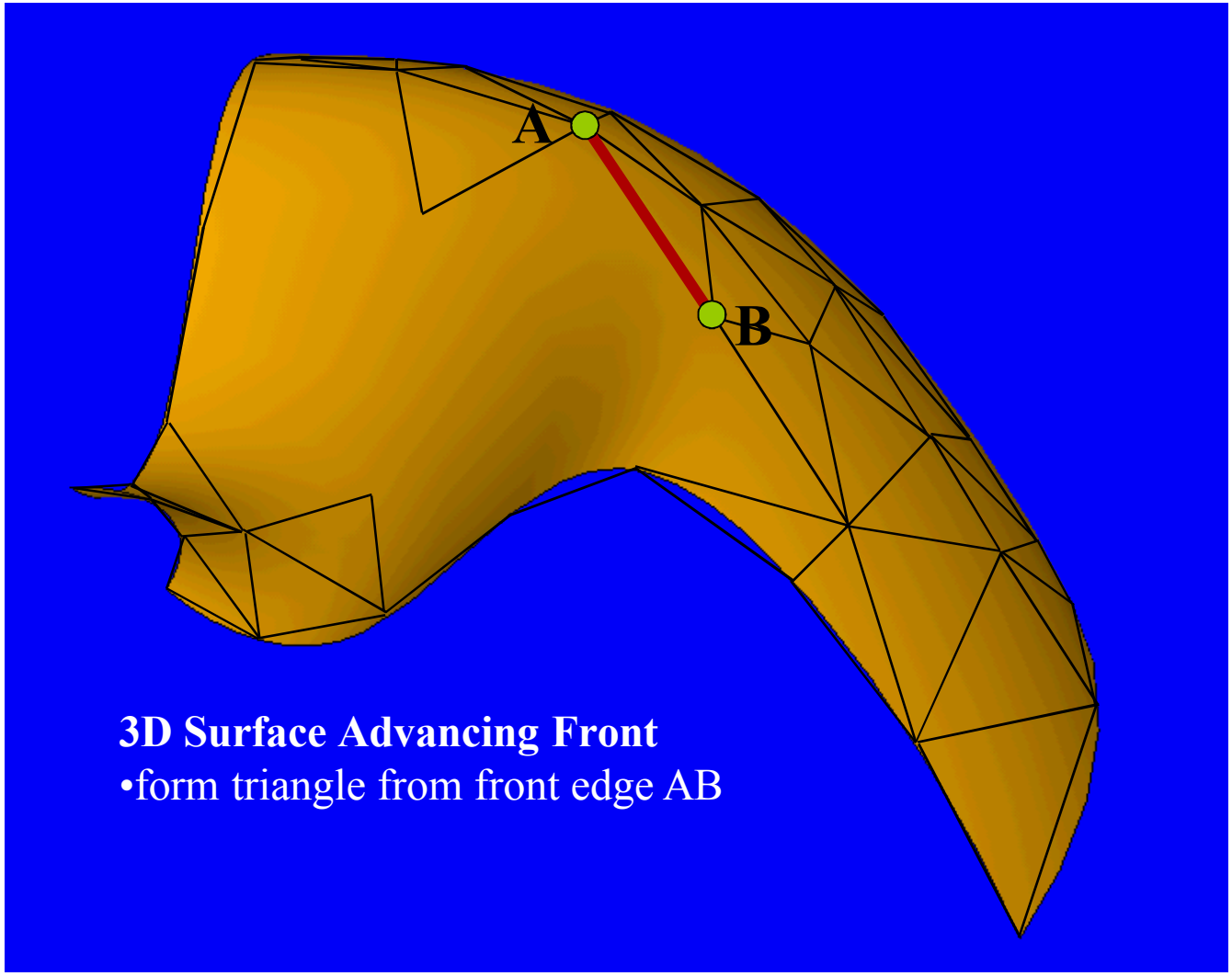
Parametric Space Meshing



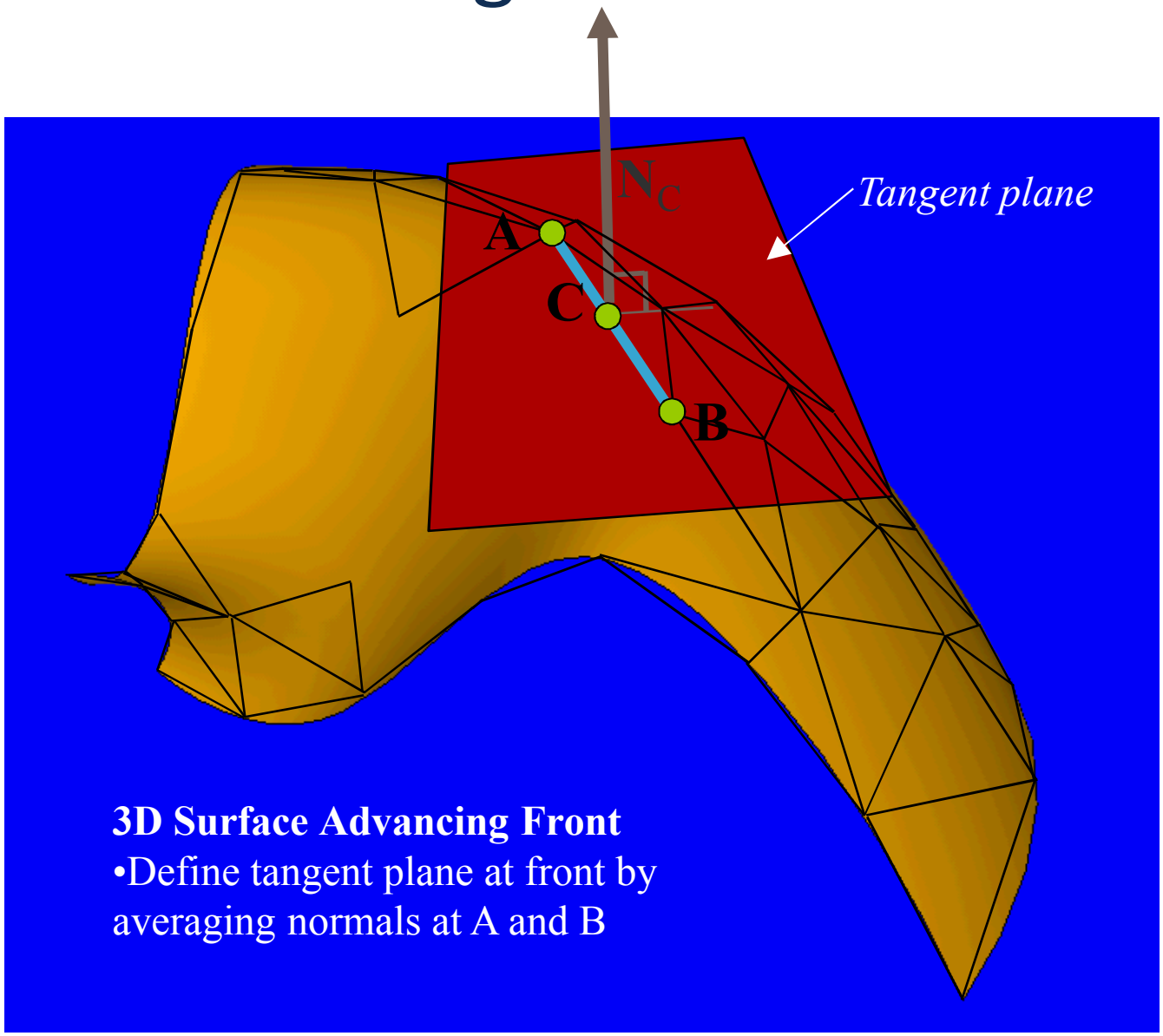
•Elements formed in 2D using parametric representation of surface
•Node locations later mapped to 3D

Surface Meshing

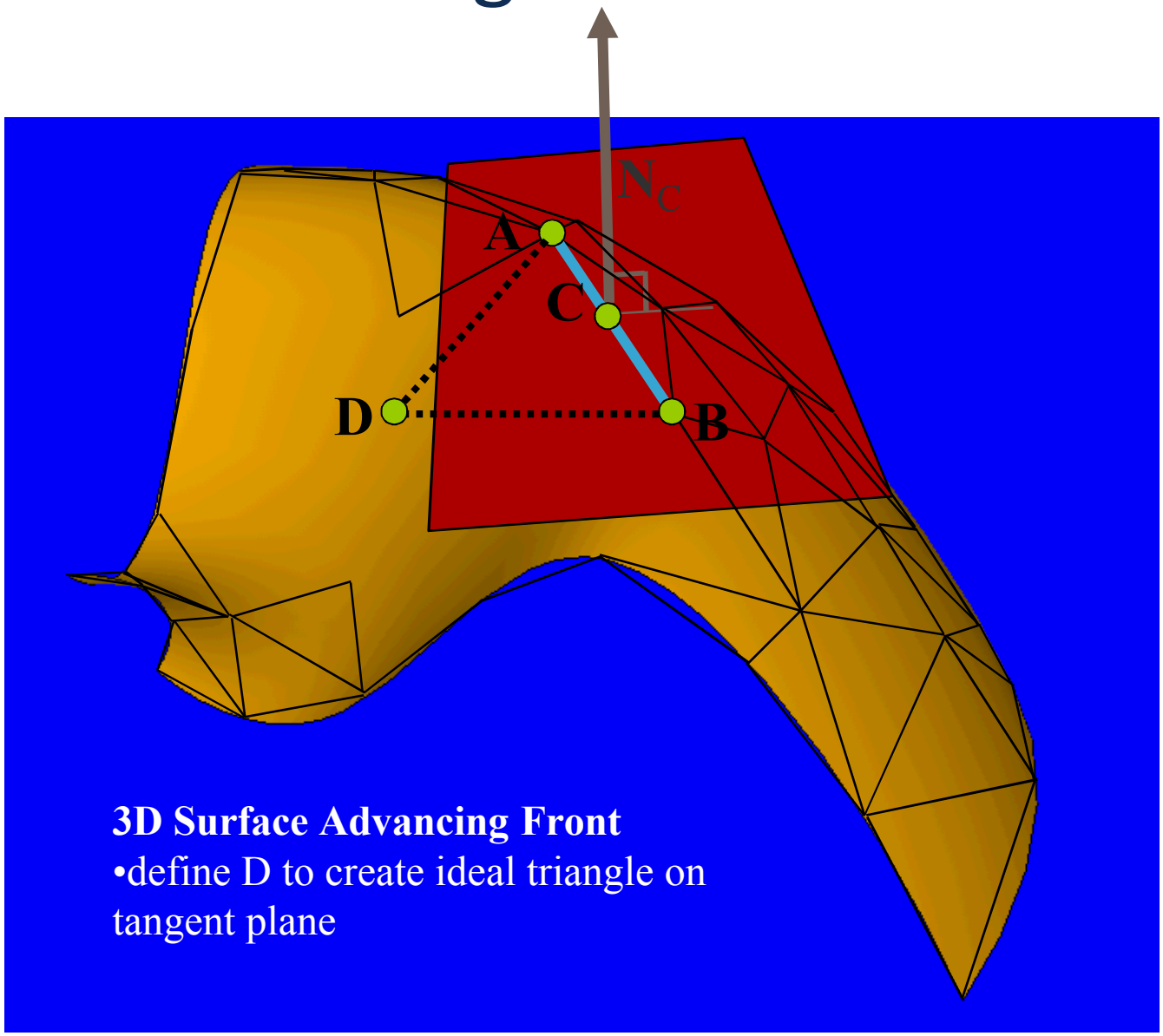
Sandia National Laboratories



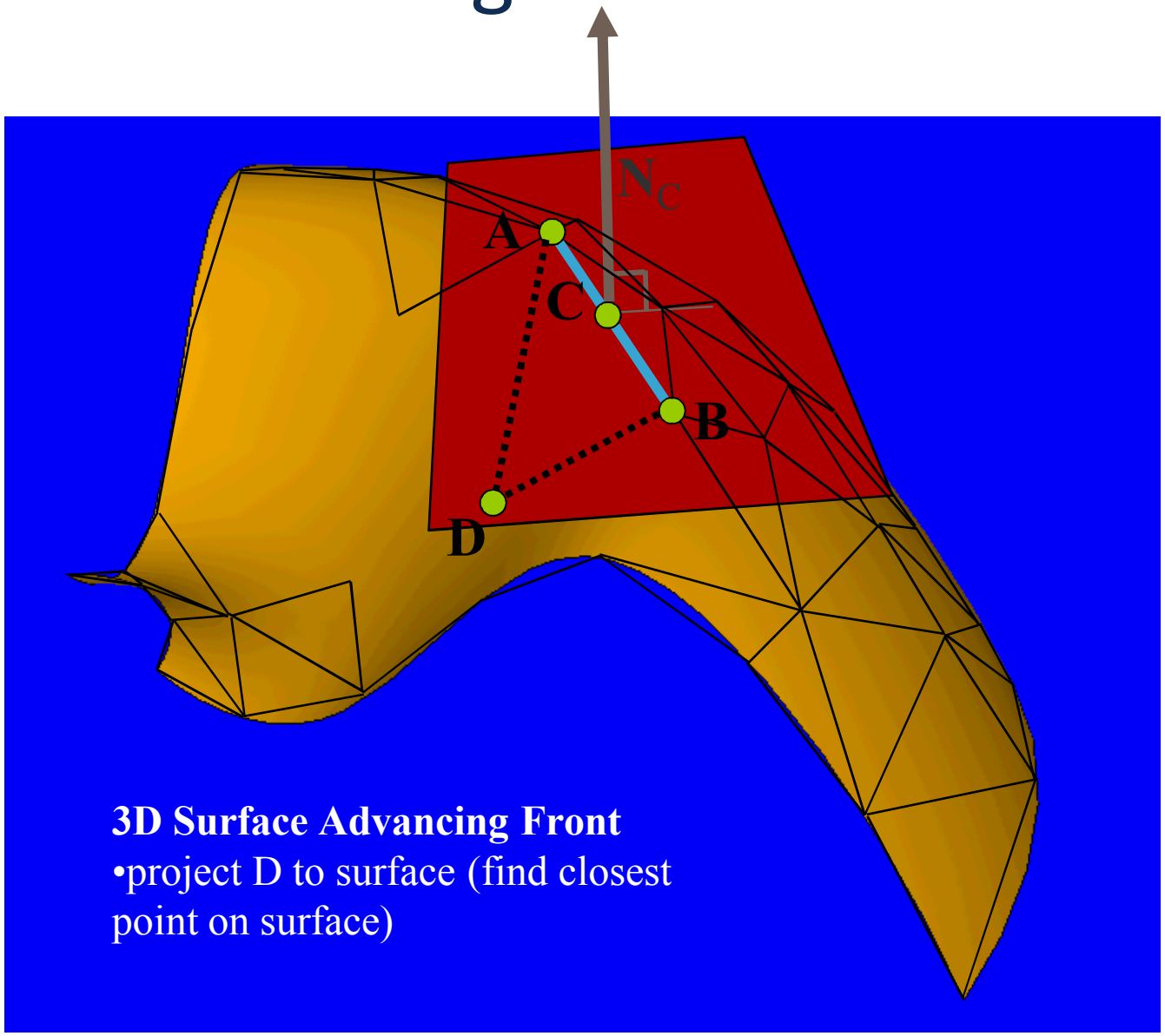
Surface Meshing



Surface Meshing



Surface Meshing

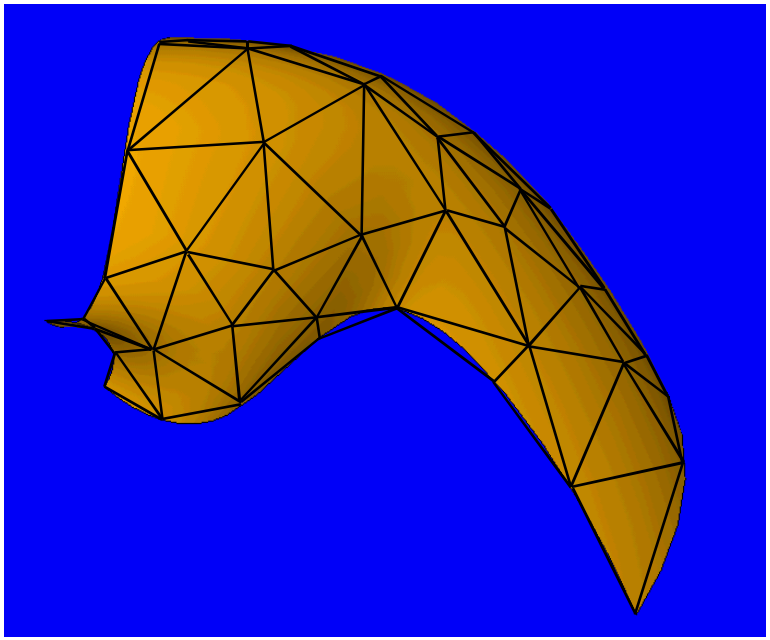


Surface Meshing

Sandia National Laboratories

3D Surface Advancing Front

- Must determine overlapping or intersecting triangles in 3D. (Floating point robustness issues)
- Extensive use of geometry evaluators (for normals and projections)
- Typically slower than parametric implementations
- Generally higher quality elements
- Avoids problems with poor parametric representations (typical in many CAD environments)
- (Lo,96;97); (Cass,96)

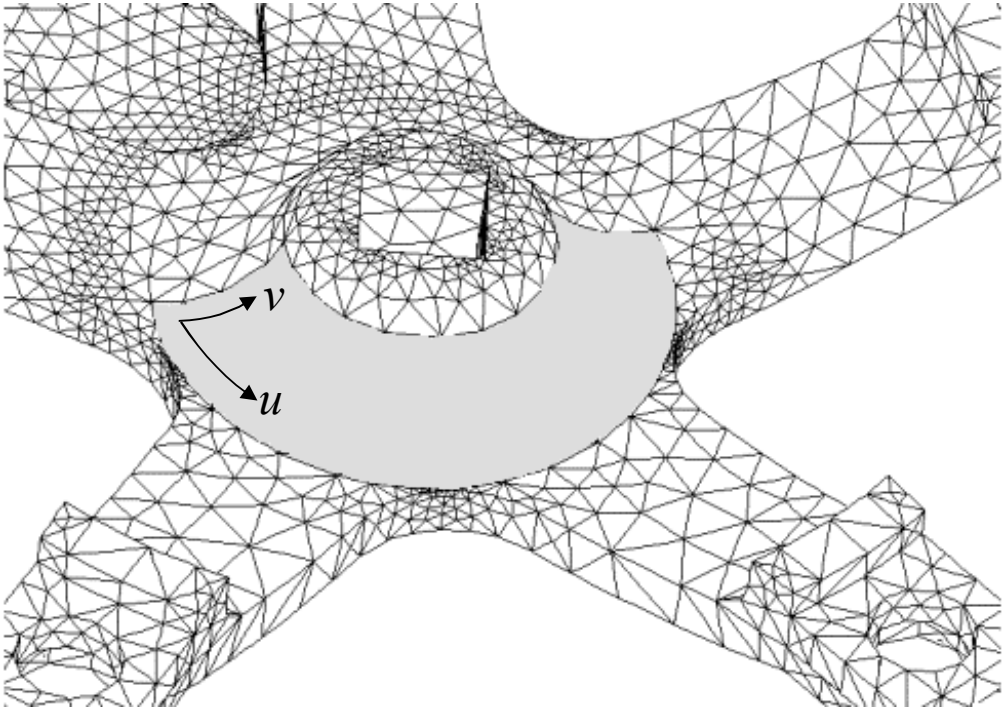


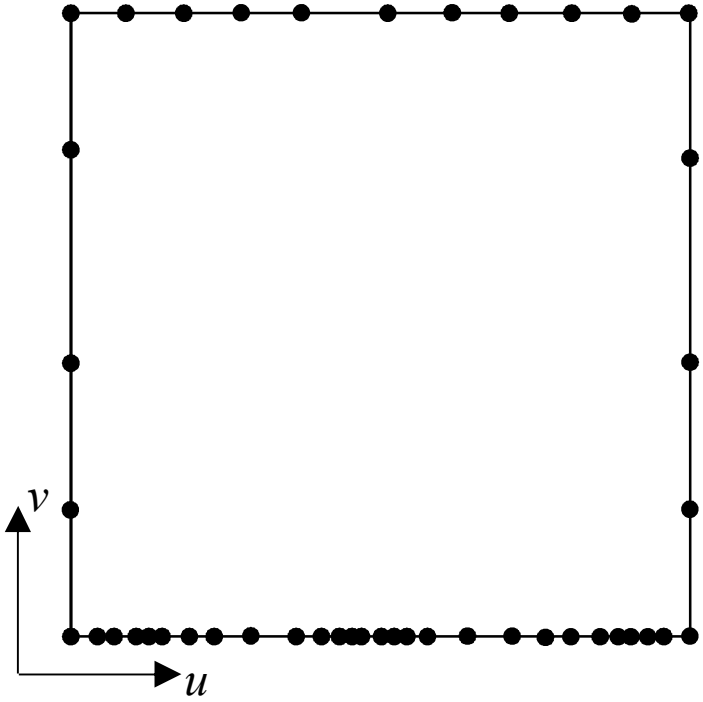
Surface Meshing

Sandia National Laboratories

Parametric Space Mesh Generation

- Parameterization of the NURBS provided by the CAD model can be used to reduce the mesh generation to 2D

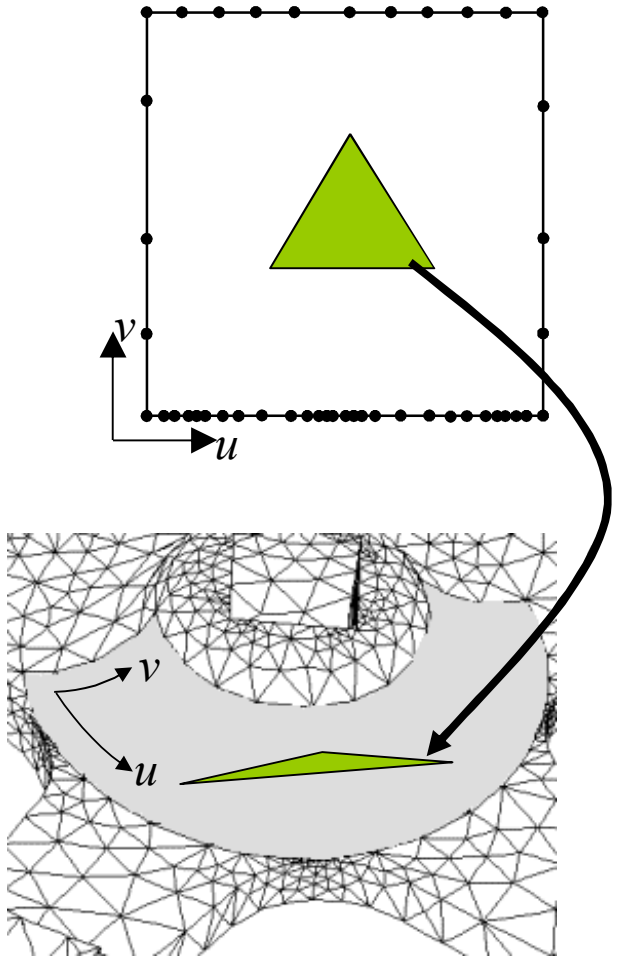




Surface Meshing

Parametric Space Mesh Generation

- Isotropic: Target element shapes are equilateral triangles
  - Equilateral elements in parametric space may be distorted when mapped to 3D space.
  - If parametric space resembles 3D space without too much distortion from *u*-*v* space to *x*-*y*-*z* space, then isotropic methods can be used.

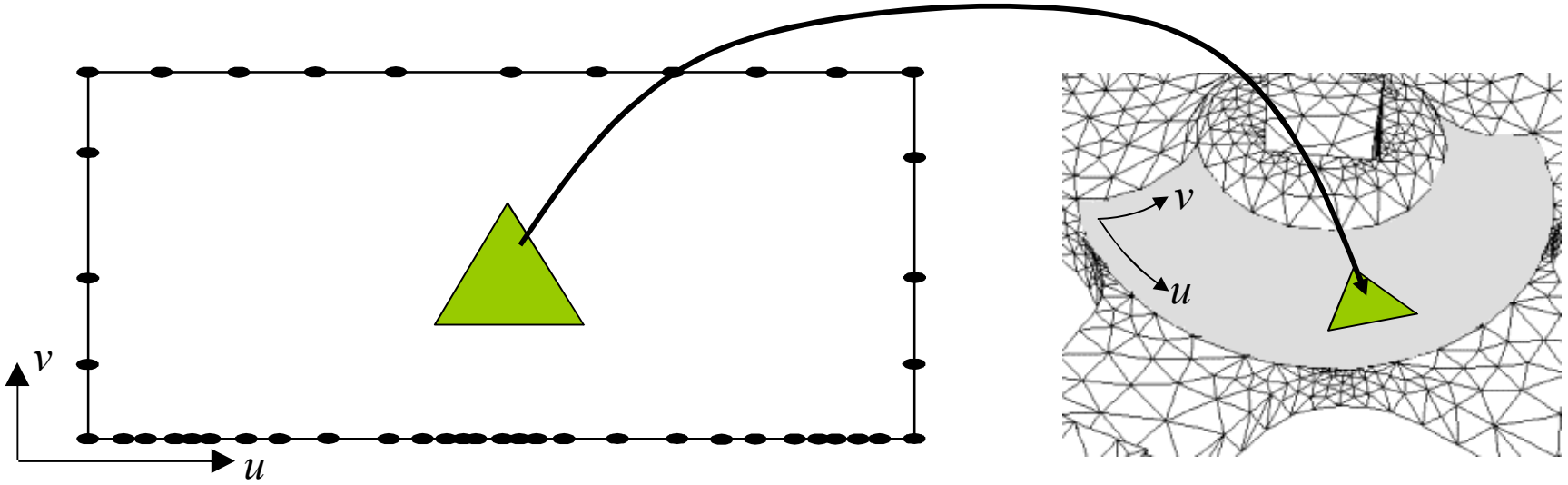


Surface Meshing

Sandia National Laboratories

Parametric Space Mesh Generation

- Parametric space can be "customized" or *warped* so that isotropic methods can be used.
- Works well for many cases.
- In general, isotropic mesh generation does not work well for parametric meshing



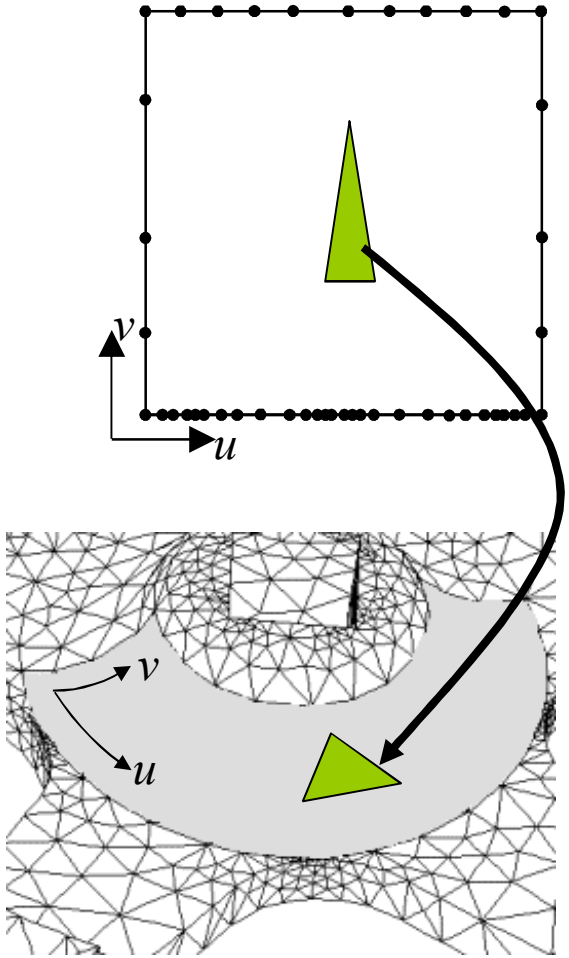
Warped parametric space

Surface Meshing

Sandia National Laboratories

Parametric Space Mesh Generation

- Anisotropic: Triangles are stretched based on a specified vector field
  - Triangles appear stretched in 2d (parametric space), but are near equilateral in 3D



Surface Meshing

Sandia National Laboratories

Parametric Space Mesh Generation

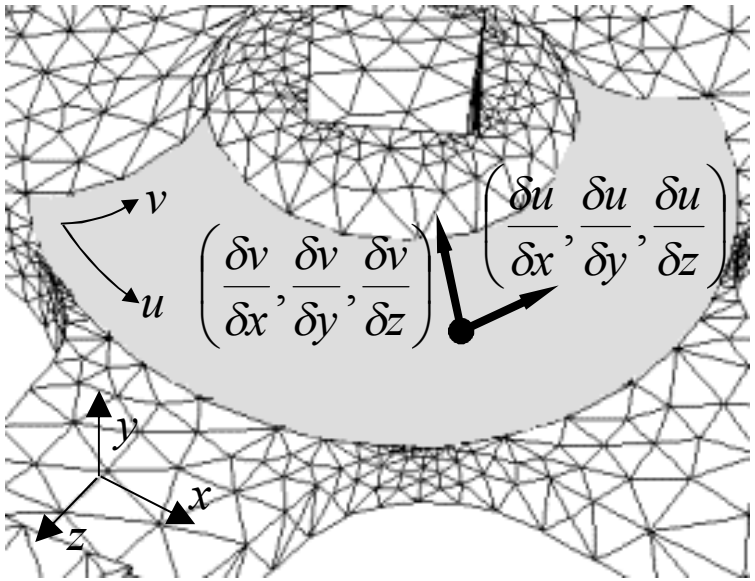
•Stretching is based on field of surface derivatives

$$\Delta\mathbf{u} = \left(\frac{\delta u}{\delta x}, \frac{\delta u}{\delta y}, \frac{\delta u}{\delta z}\right) \qquad \Delta\mathbf{v} = \left(\frac{\delta v}{\delta x}, \frac{\delta v}{\delta y}, \frac{\delta v}{\delta z}\right)$$

•Metric, **M** can be defined at every location on surface. Metric at location **X** is:

$$\mathbf{M}(\mathbf{X}) = \begin{bmatrix} E & F \\ F & G \end{bmatrix}$$

$$E = \Delta\mathbf{u} \cdot \Delta\mathbf{u} \qquad F = \Delta\mathbf{u} \cdot \Delta\mathbf{v} \qquad G = \Delta\mathbf{v} \cdot \Delta\mathbf{v}$$

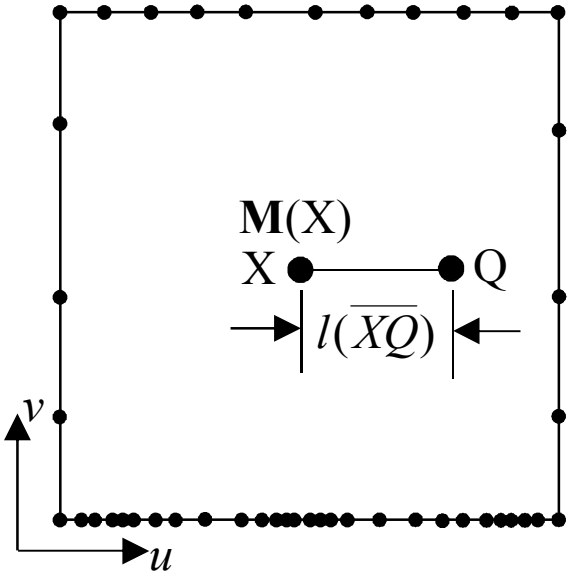


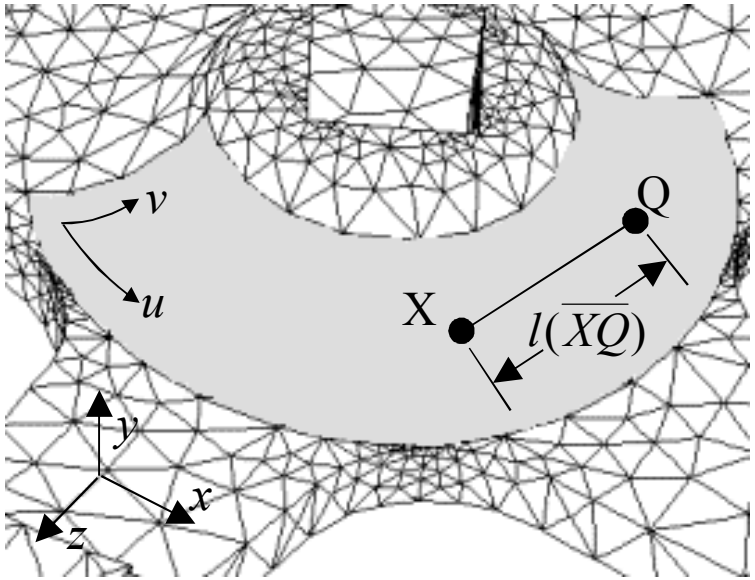
Surface Meshing

Parametric Space Mesh Generation

•Distances in parametric space can now be measured as a function of direction and location on the surface. Distance from point X to Q is defined as:

$$l(\overline{XQ}) \approx \sqrt{\overline{XQ}^T \mathbf{M}(\mathbf{X}) \overline{XQ}}$$



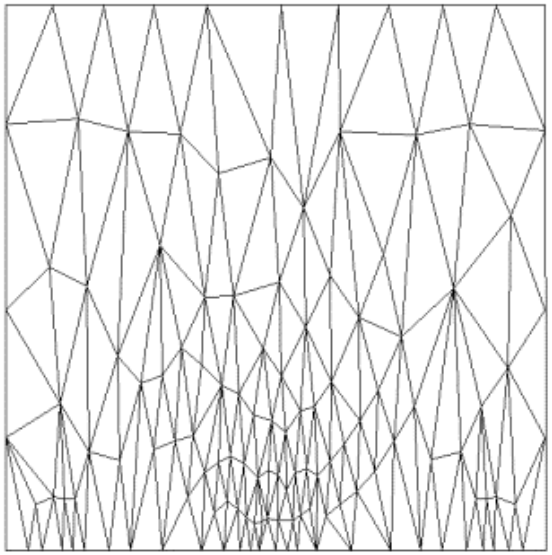


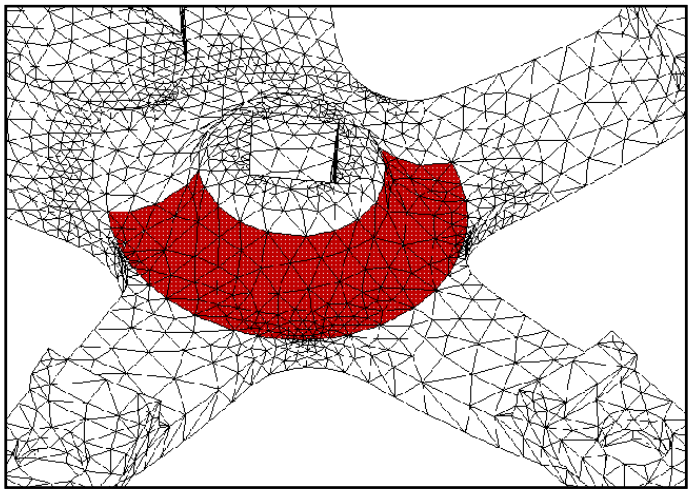
Surface Meshing

Sandia National Laboratories

Parametric Space Mesh Generation

•Use essentially the same isotropic methods for 2D mesh generation, except distances and angles are now measured with respect to the local metric tensor **M**(X).
•Can use Delaunay (George, 99) or Advancing Front Methods (Tristano,98)



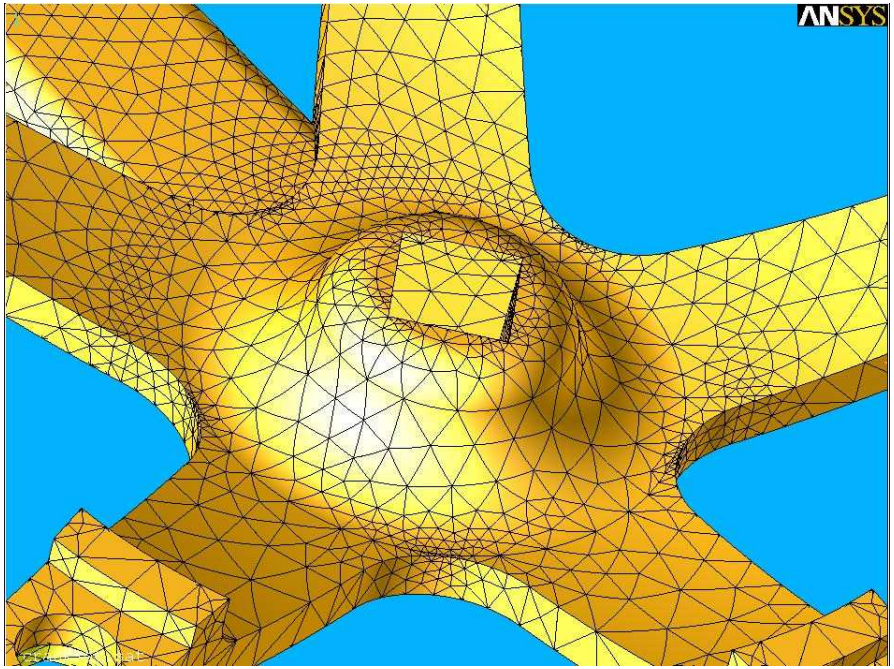


Surface Meshing

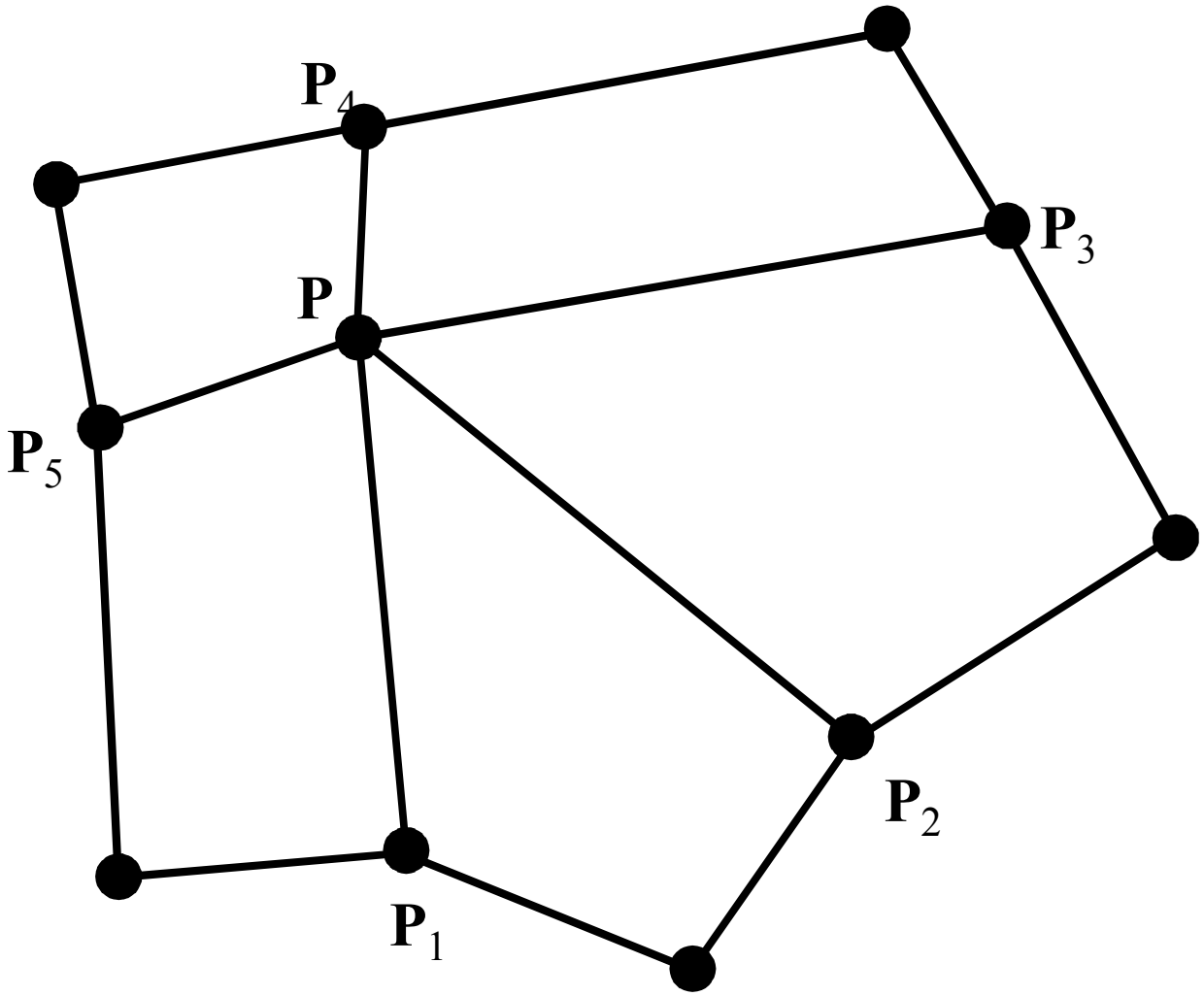
Sandia National Laboratories

Parametric Space Mesh Generation

- Is generally faster than 3D methods
- Is generally more robust (No 3D intersection calculations)
- Poor parameterization can cause problems
- Not possible if no parameterization is provided
    - Can generate your own parametric space (Flatten 3D surface into 2D) (Marcum, 99) (Sheffer,00)



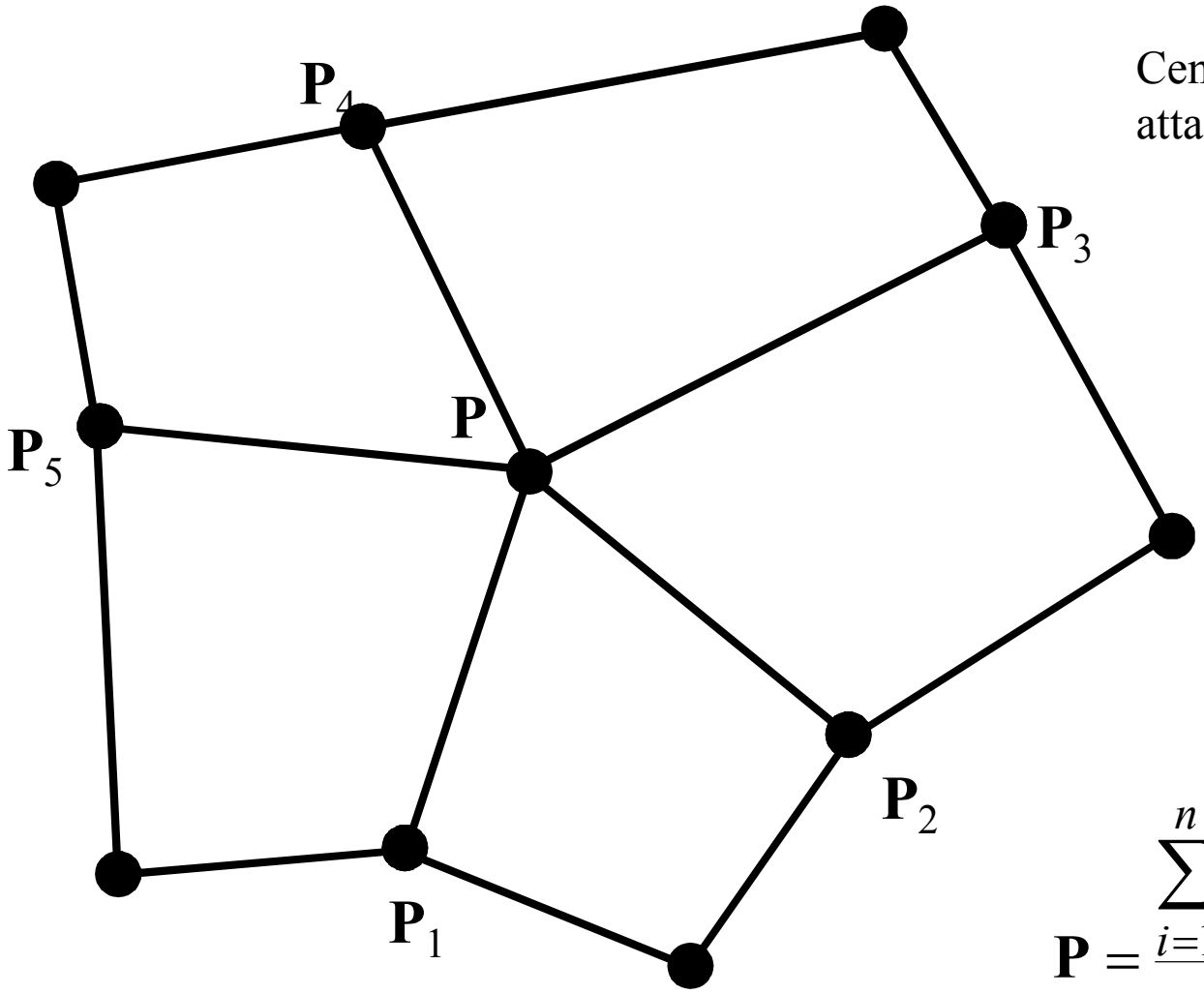
Smoothing



(Field, 1988)

Laplacian

Smoothing



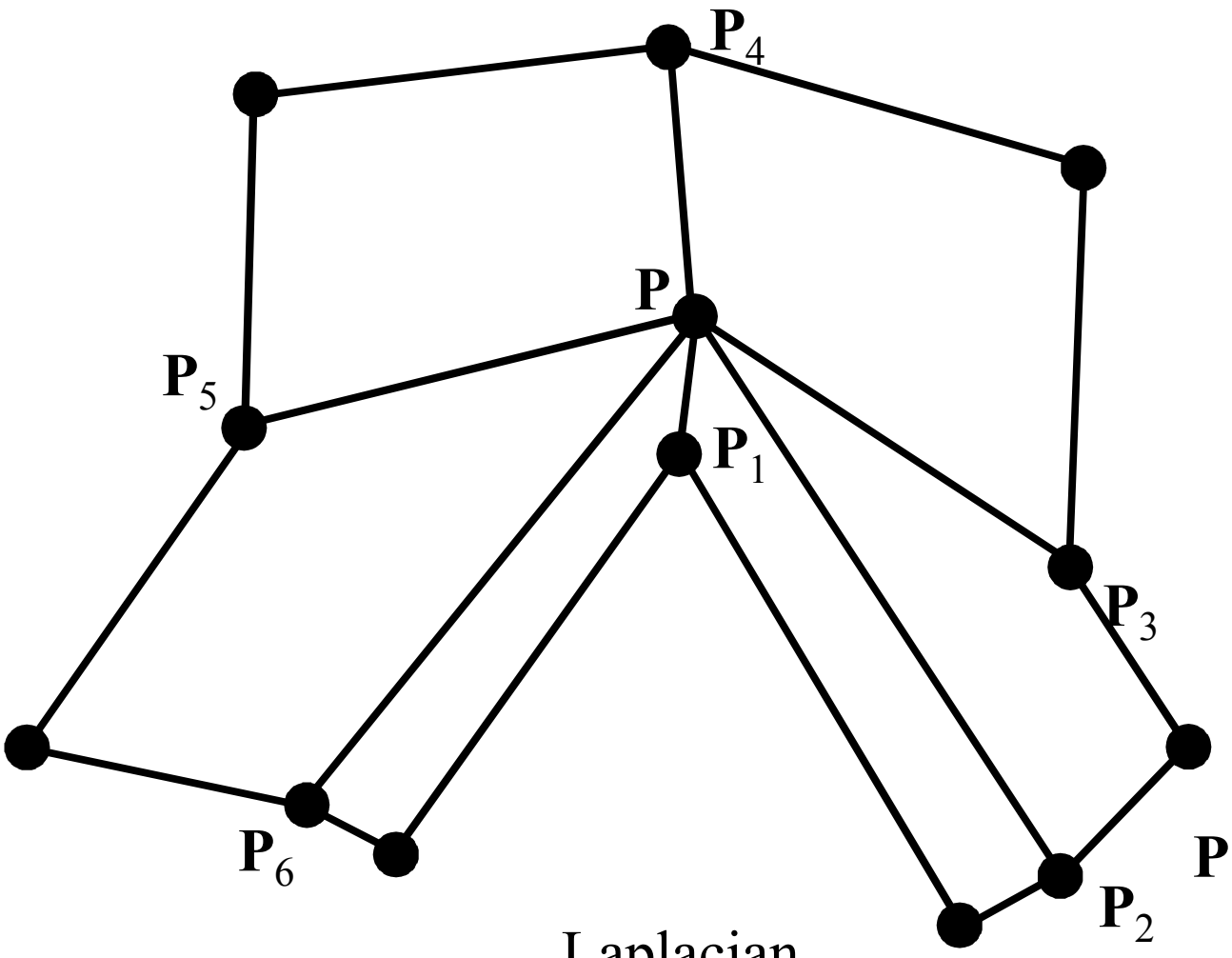
Centroid of attached nodes

$$\mathbf{P} = \frac{\sum_{i=1}^{n} \mathbf{P}_i}{n}$$

(Field, 1988)

Laplacian

Smoothing



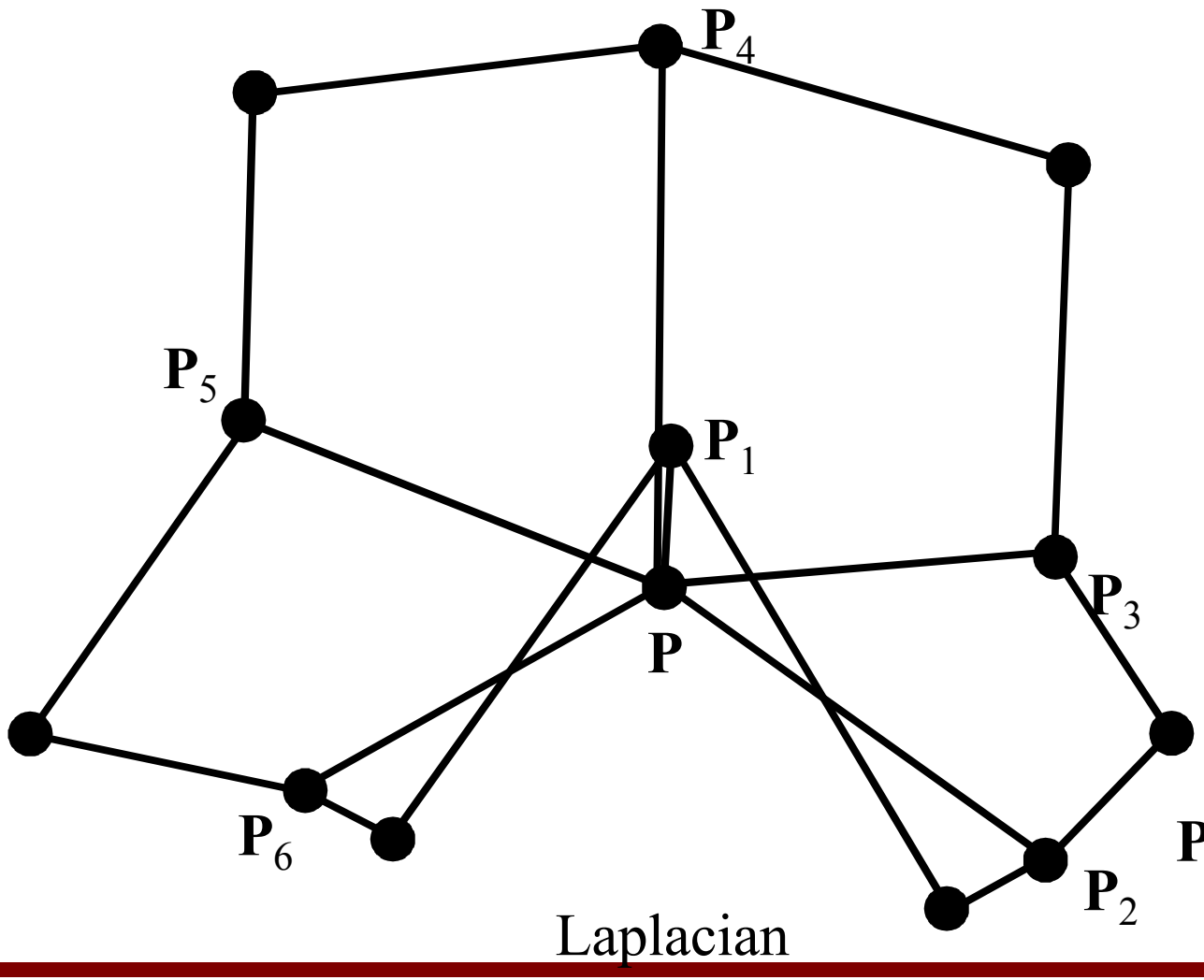
Centroid of attached nodes

Can create inverted elements

$$\mathbf{P} = \frac{\sum_{i=1}^{n} \mathbf{P}_i}{n}$$

Laplacian

Smoothing



Centroid of attached nodes

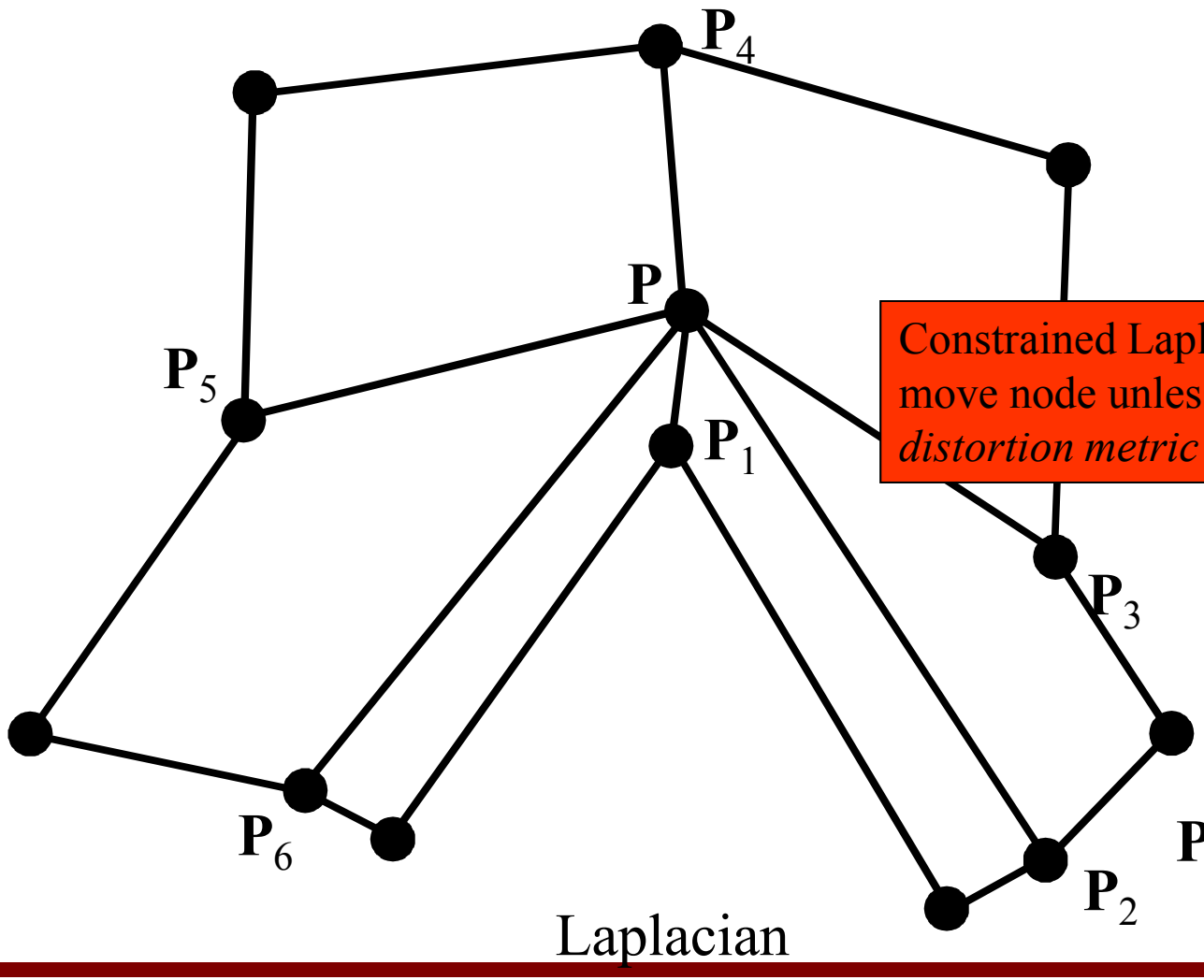
Can create inverted elements

$$\mathbf{P} = \frac{\sum_{i=1}^{n} \mathbf{P}_i}{n}$$

Laplacian

Smoothing

Sandia National Laboratories



Centroid of attached nodes

Can create inverted elements

Constrained Laplacian: Do not move node unless minimum *distortion metric* is improved

$$\mathbf{P} = \frac{\sum_{i=1}^{n} \mathbf{P}_i}{n}$$

Laplacian

Smoothing

Sandia National Laboratories

Smoothing Procedure

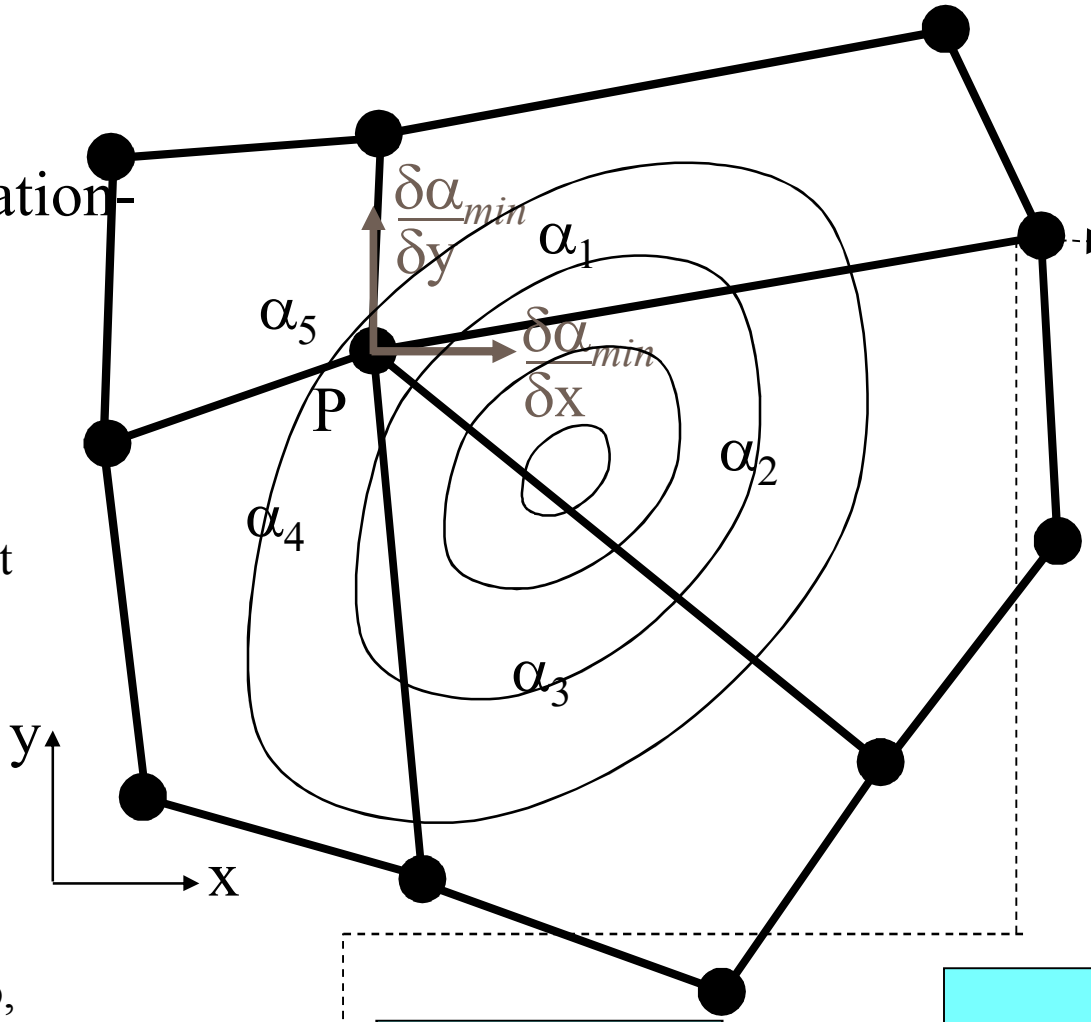
```
LET iter=0
SET all NODE v to active
REPEAT (main smoothing loop)
   FOR EACH NODE v DO:
      1.  If v has been deactivated from smooth, then CONINUE.
      2.  distance moved = move_node(v)
      3.  If distance moved < move tolerance, then
             deactivate v
          Else,
             Allow move.
             If neighbor node to v is inactive, then activate it.
    END DO
    iter++;
UNTIL all nodes are deactivated OR iter > MAX_ITER (end of main smoothing loop)
```

Smoothing

Optimization-Based Method

Steepest descent optimization

(Canann, Tristano, Staten, 1998)



Compute shape metric α_i for each quad at P

Choose α_{min} to optimize

Compute gradient vector $\mathbf{g}=\mathbf{g}_{min}$ by perturbing small δ in x and y

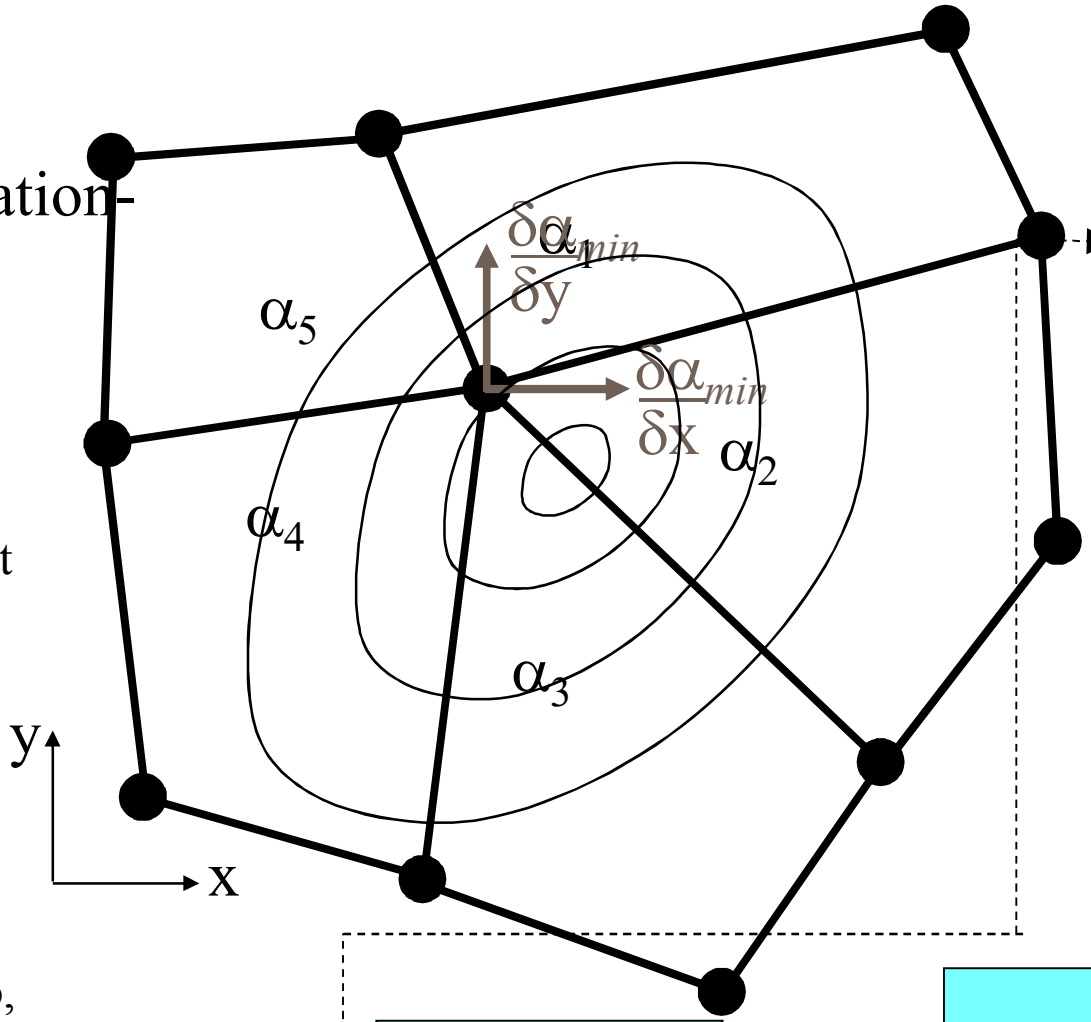
$$\gamma = \min\left\{\frac{\alpha_i - \alpha_{\min}}{\mathbf{g}\cdot\mathbf{g} - \mathbf{g}\cdot\mathbf{g}_i}\right\}$$

$$\mathbf{P}_{new} = \mathbf{P} + \gamma\mathbf{g}$$

Smoothing

Optimization-Based Method

Steepest descent optimization



Compute shape metric α_i for each quad at P

Choose α_{min} to optimize

Compute gradient vector $\mathbf{g}=\mathbf{g}_{min}$ by perturbing small δ in x and y

$$\gamma = \min\left\{\frac{\alpha_i - \alpha_{\min}}{\mathbf{g}\cdot\mathbf{g} - \mathbf{g}\cdot\mathbf{g}_i}\right\}$$

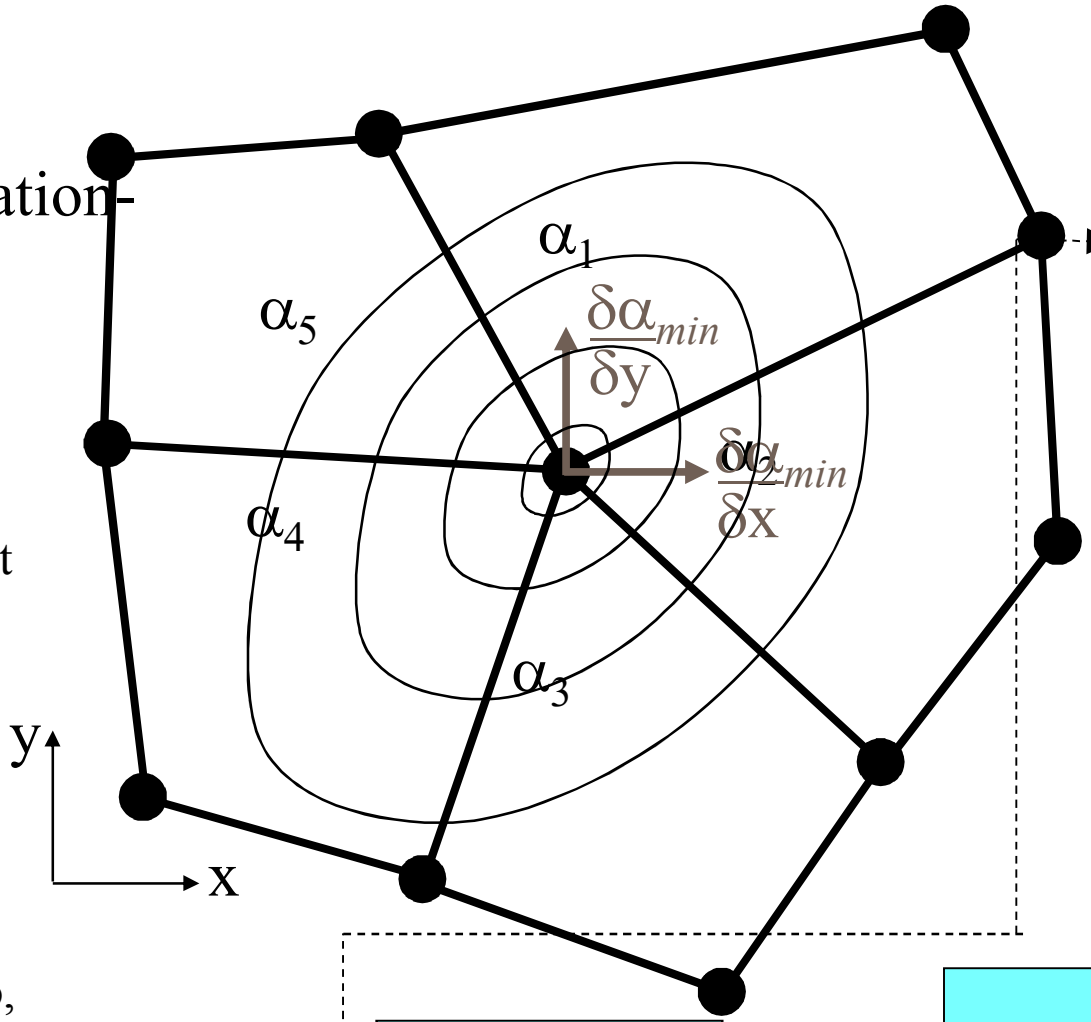
$$\mathbf{P}_{new} = \mathbf{P} + \gamma\mathbf{g}$$

(Canann, Tristano, Staten, 1998)

Smoothing

Optimization-Based Method

Steepest descent optimization



Compute shape metric α_i for each quad at P

Choose α_{min} to optimize

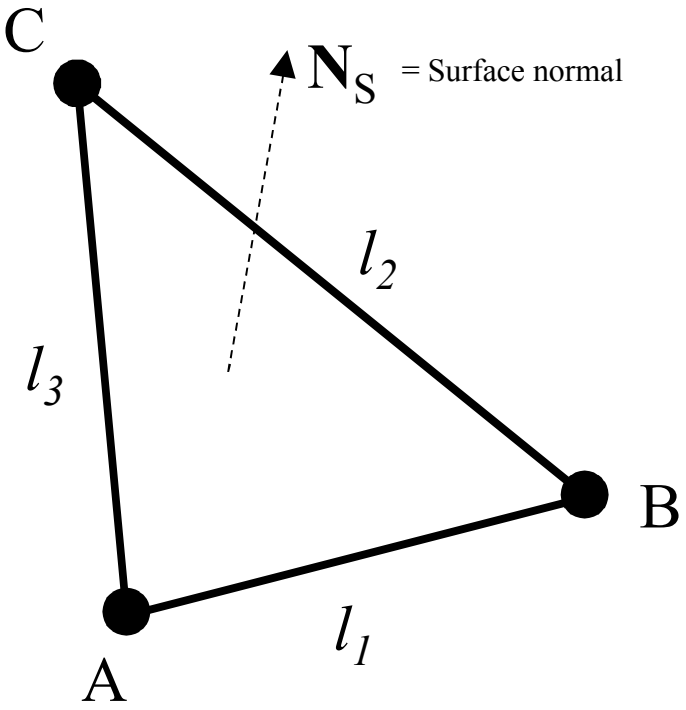
Compute gradient vector $\mathbf{g}=\mathbf{g}_{min}$ by perturbing small δ in x and y

$$\gamma = \min\left\{\frac{\alpha_i - \alpha_{\min}}{\mathbf{g}\cdot\mathbf{g} - \mathbf{g}\cdot\mathbf{g}_i}\right\}$$

$$\mathbf{P}_{new} = \mathbf{P} + \gamma\mathbf{g}$$

(Canann, Tristano, Staten, 1998)

Smoothing

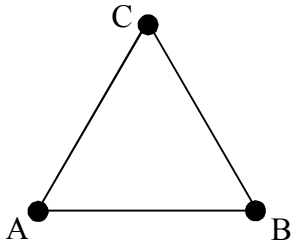


(Canann, Tristano, Staten, 1998)

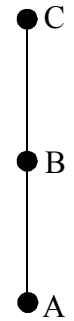
Triangle Distortion Metric

$$\alpha(ABC) = (I)\sqrt{3}\,\frac{area(ABC)}{l_1^2 + l_2^2 + l_3^2}$$

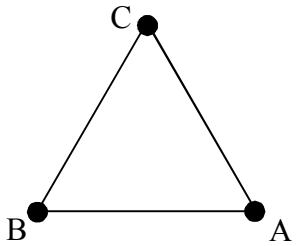
$$I = \begin{cases} 1 & for\ \left(\overline{CA} \times \overline{CB}\right) \cdot \mathbf{N}_S > 0 \\ -1 & for\ \left(\overline{CA} \times \overline{CB}\right) \cdot \mathbf{N}_S < 0 \end{cases}$$



α=1

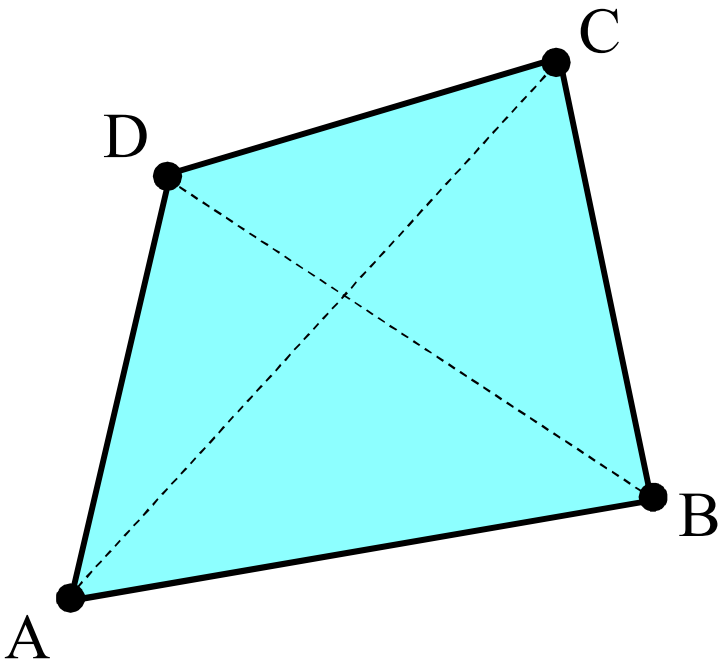


α=0



α=-1

Smoothing



Quad Distortion Metric

Compute four triangle metrics

$\alpha(\Delta ABC)$ $\alpha(\Delta CDA)$

$\alpha(\Delta BCD)$ $\alpha(\Delta DAB)$

Order metrics in descending order

$\alpha_1 \geq \alpha_2 \geq \alpha_3 \geq \alpha_4$

$$\beta = \frac{\alpha_3 \alpha_4}{\alpha_1 \alpha_2}$$

(Lee, Lo, 1994)

Smoothing

Sandia National Laboratories

Combined Laplacian and Optimization Based Smoothing

```
Compute initial distortion metrics for all elements
LET iter=0
SET all NODE v to active
REPEAT (main smoothing loop)
   FOR EACH NODE v DO:
      1.  If v has been deactivated from smooth, then CONINUE.
      2.  If v wasn't moved by optimization-based smoothing in last iteration, then
             distance_moved = move_node_laplacian(v)
      3.  If (iter >= 2) then
             distance_moved = move_node_optimize(v)
      4.  If distance_moved < move_tolerance AND α_min > min_allowable_metric, then
             deactivate v
          Else,
             Allow move, and update adjacent distortion metrics to v.
             If neighbor node to v is inactive, then activate it.
    END DO
    iter++;
UNTIL all nodes are deactivated OR iter > MAX_ITER (end of main smoothing loop)
```

(Canann, Tristano, Staten, 1998)
(Freitag, 1997)