An Introduction to Mesh Generation Algorithms
Part 1

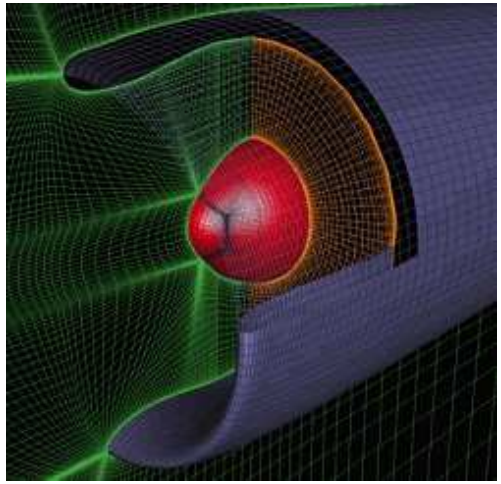
Fathi El-Yafi

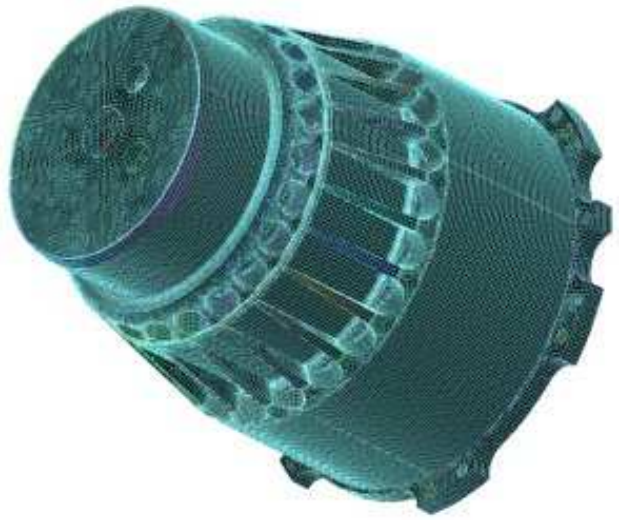
Project and Software Development Manager

Engineering Simulation

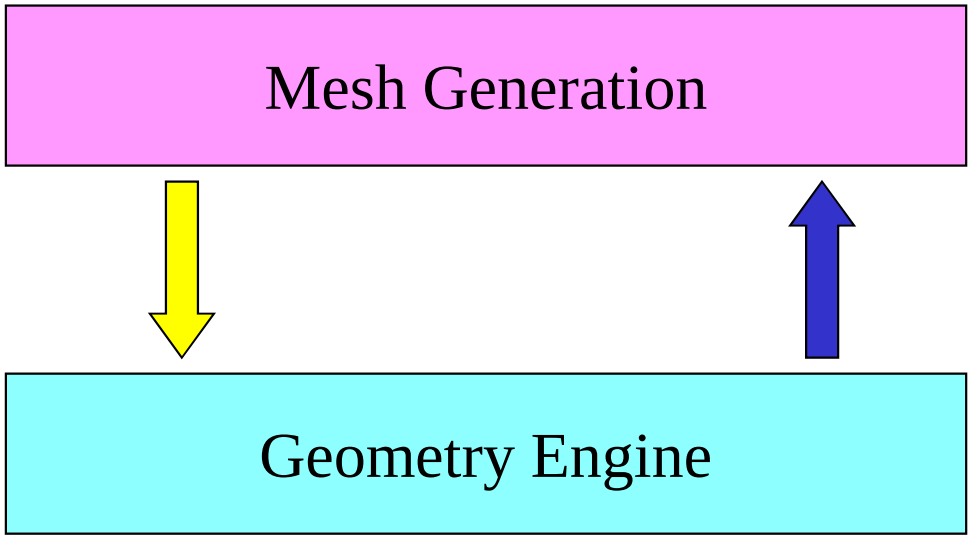
Overview

- **The Mesh Generation Process**
- **Meshing Algorithms**
  - Tri / Tet Methods
  - Quad / Hex Methods
  - Hybrid Methods
  - Surface Meshing
- **Algorithm Characteristics**

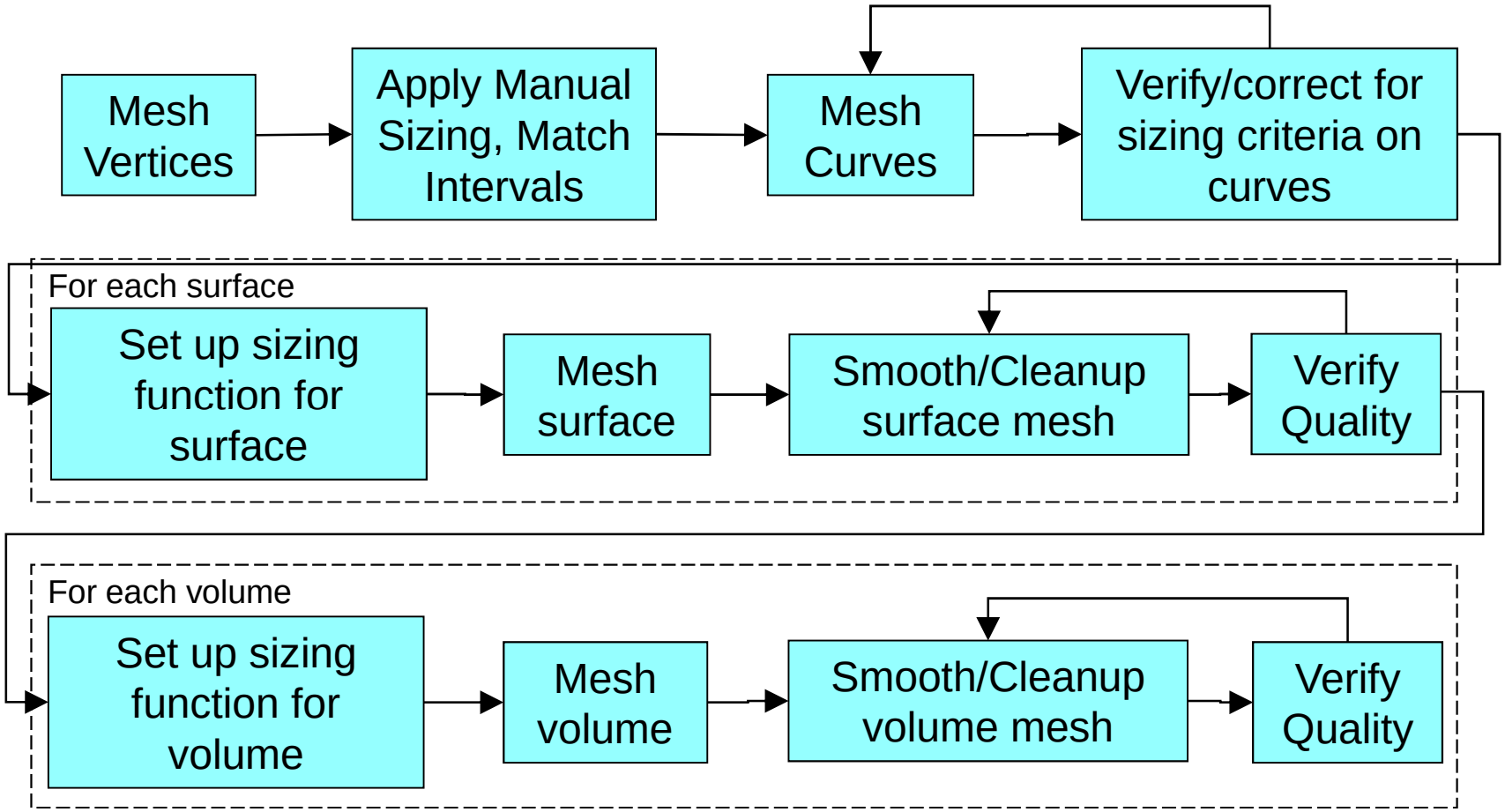




Geometry / Mesh

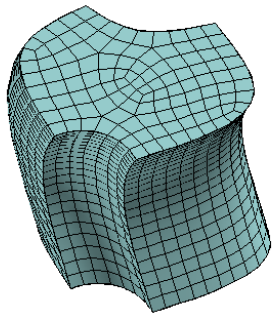


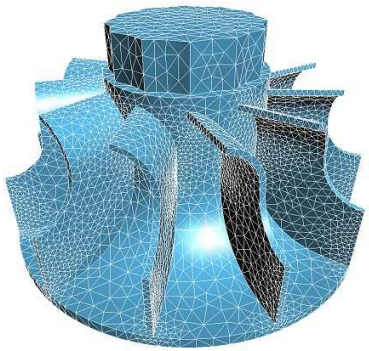
Mesh Generation Process

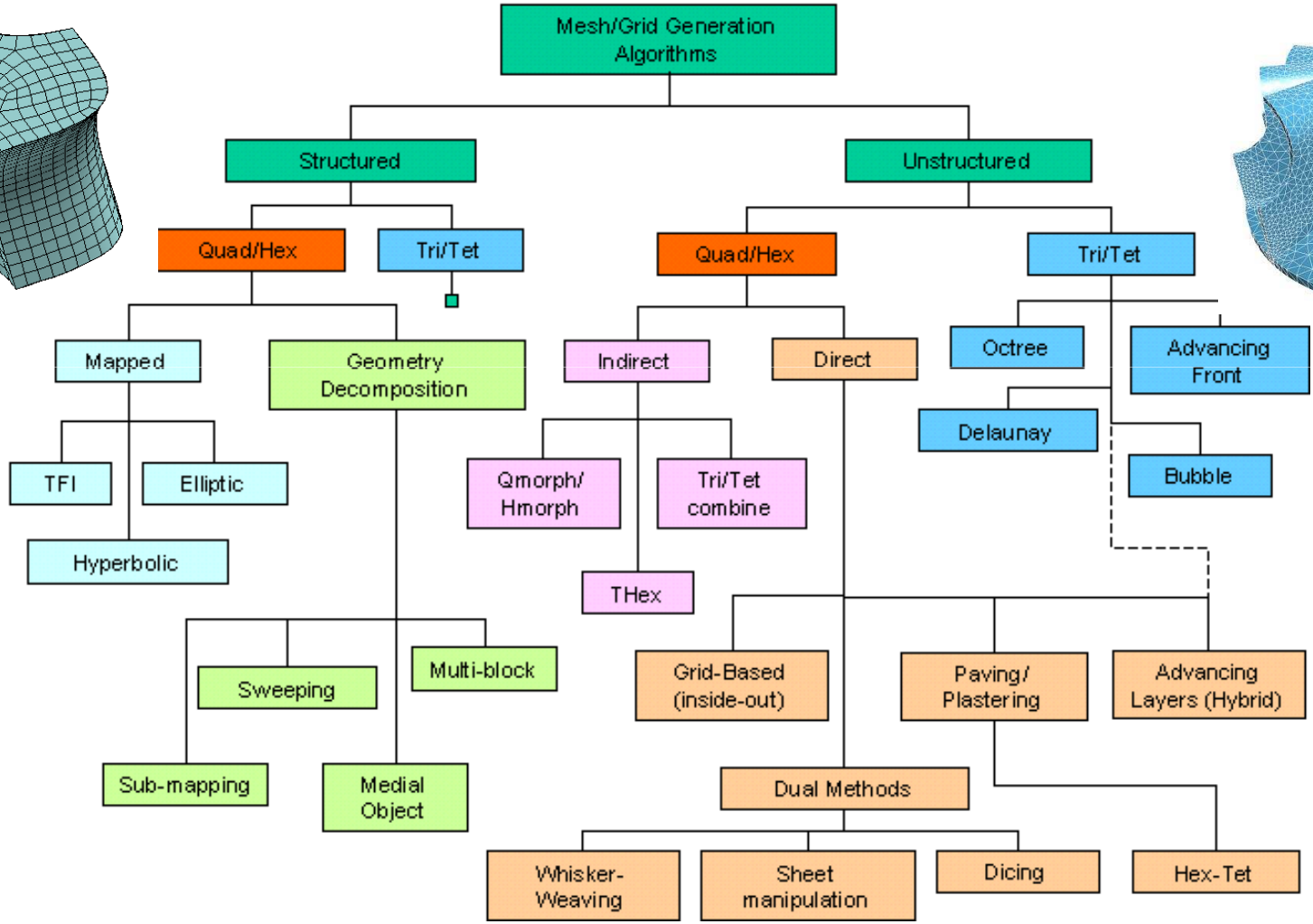


The Mesh Generation Process

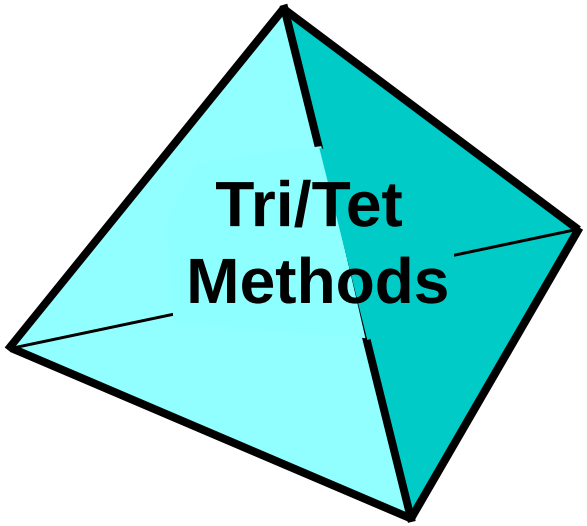
Meshing Algorithms



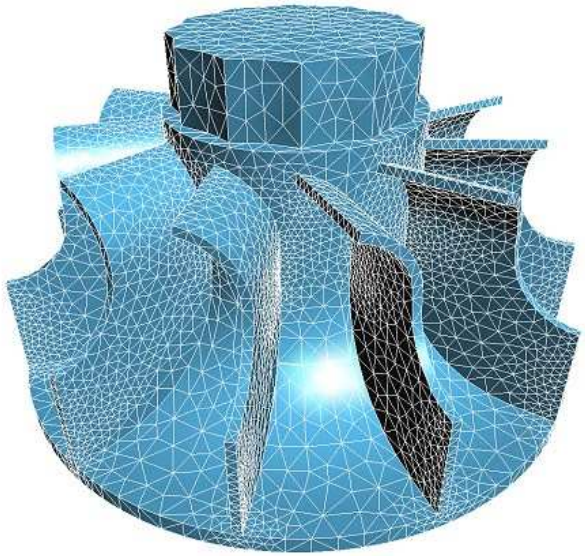


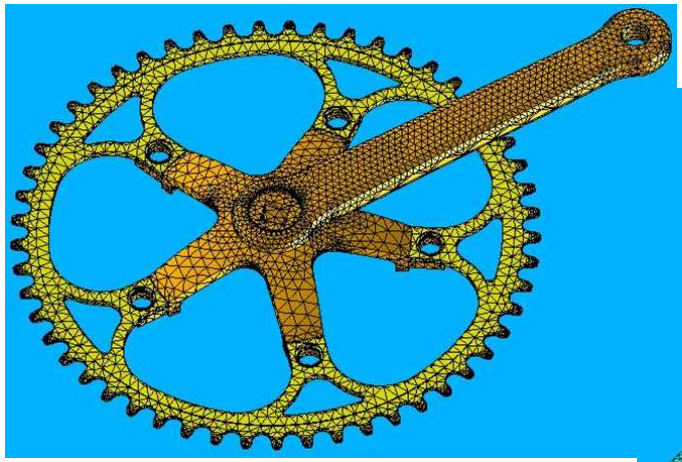


Meshing Algorithms

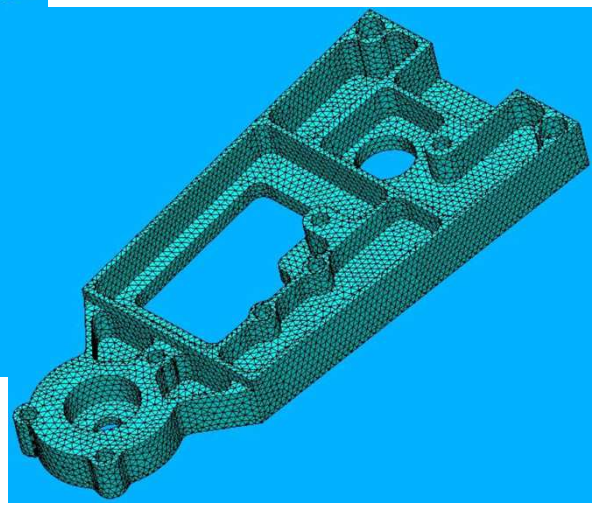


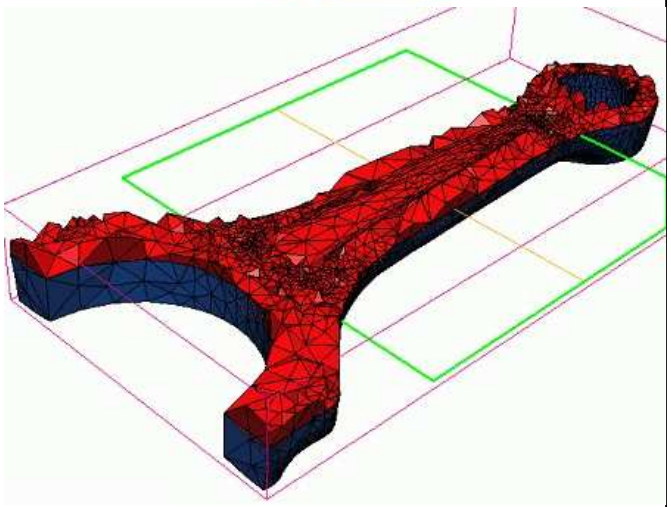
- Octree
- Advancing Front
- Delaunay



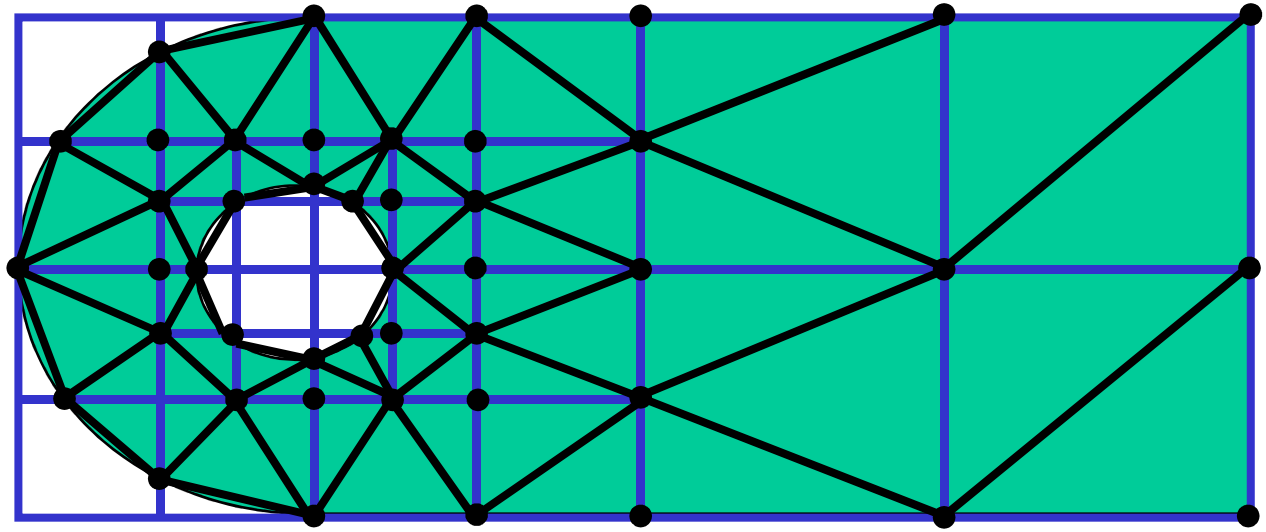


http://www.ansys.com



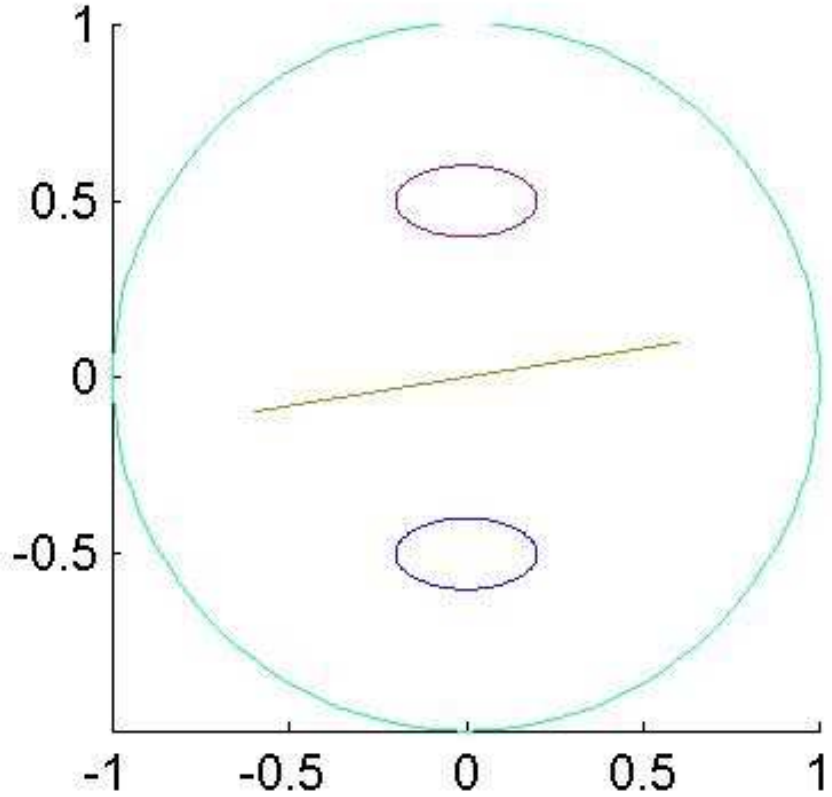


Octree / Quadtree



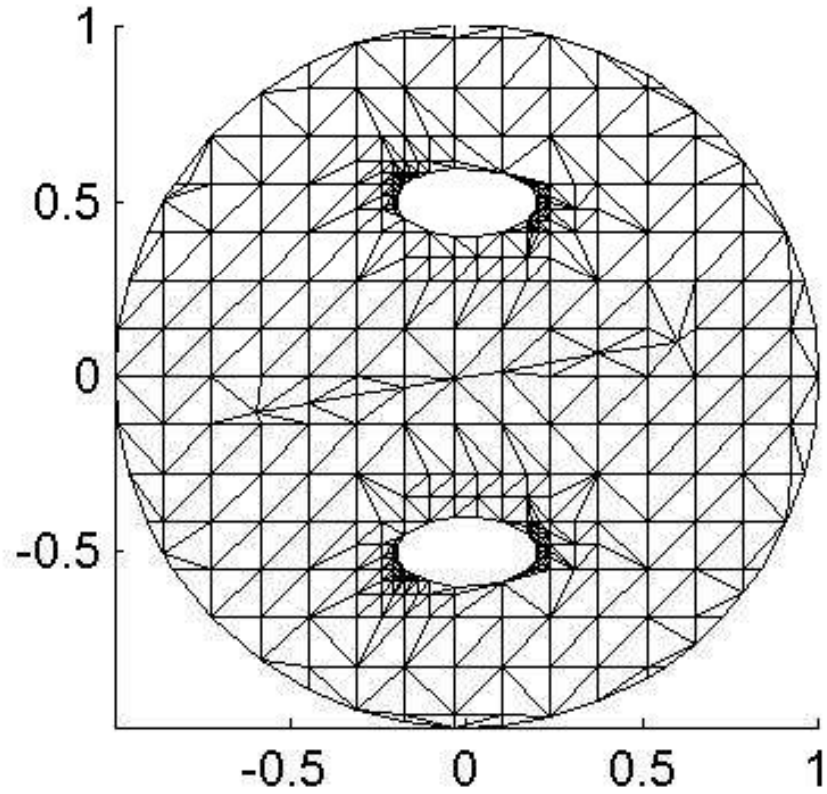
- Define initial bounding box (*root* of Quadtree)
- Recursively break into 4 *leaves* per *root* to resolve geometry
- Find intersections of leaves with geometry boundary
- Mesh each *leaf* using corners, side nodes and intersections with geometry
- Delete Outside
- (Yerry and Shephard, 84), (Shepherd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

Octree / Quadtree



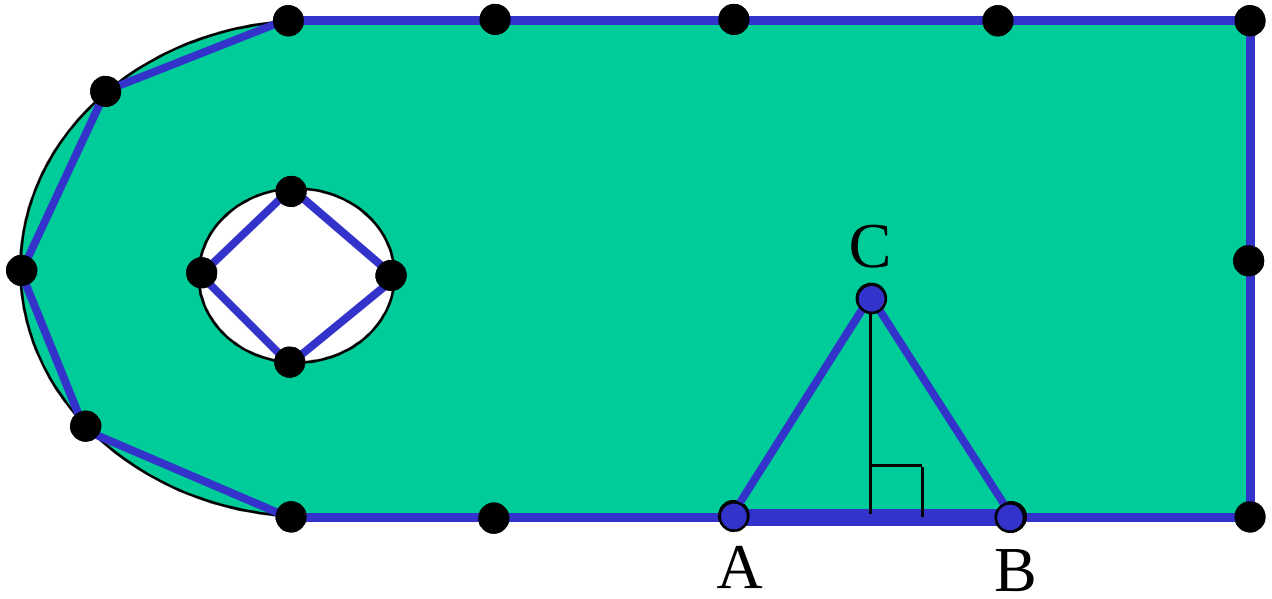
QMG,
Cornell University

Octree / Quadtree



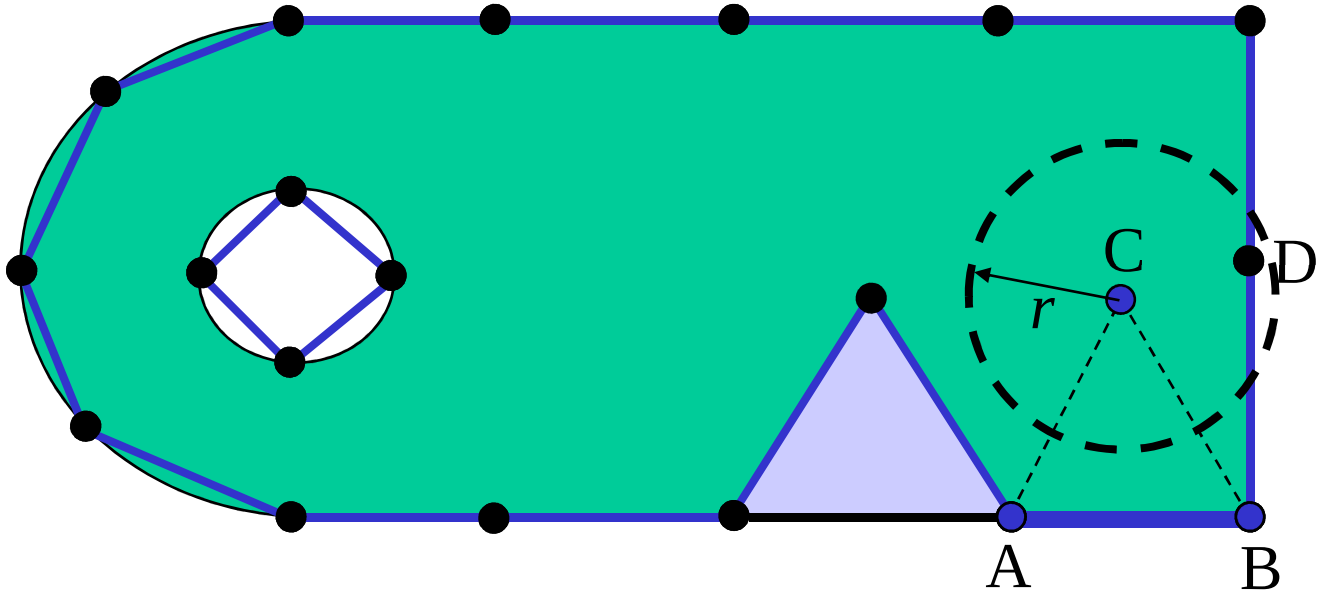
QMG,
Cornell University

Advancing Front



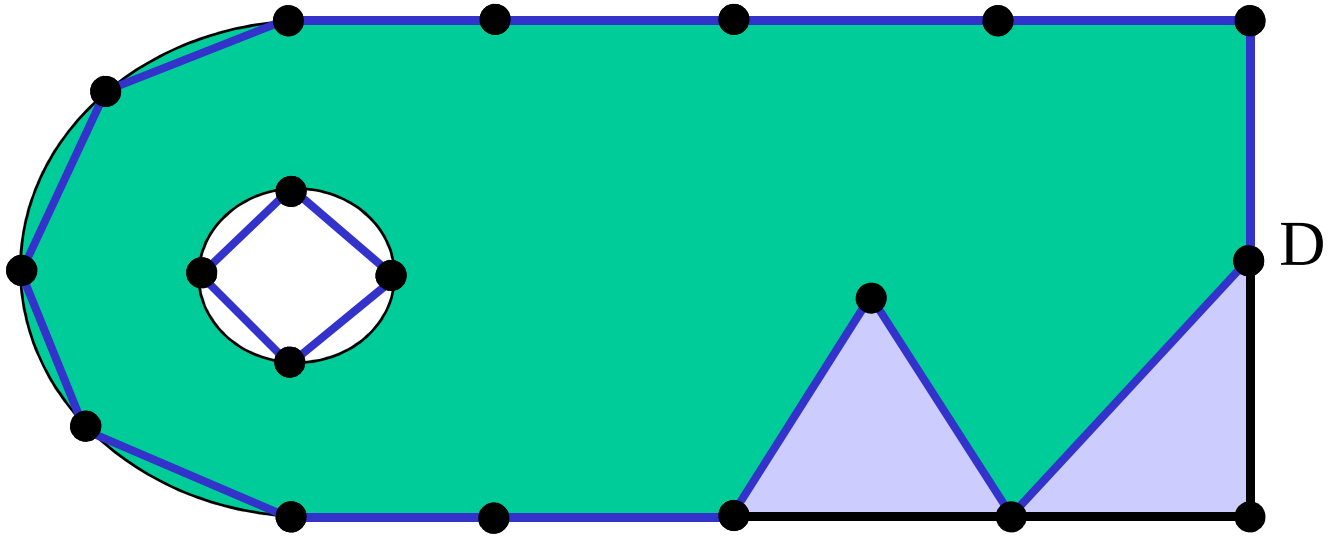
- Begin with boundary mesh - define as initial *front*
- For each edge (face) on front, locate ideal node C based on front AB

Advancing Front



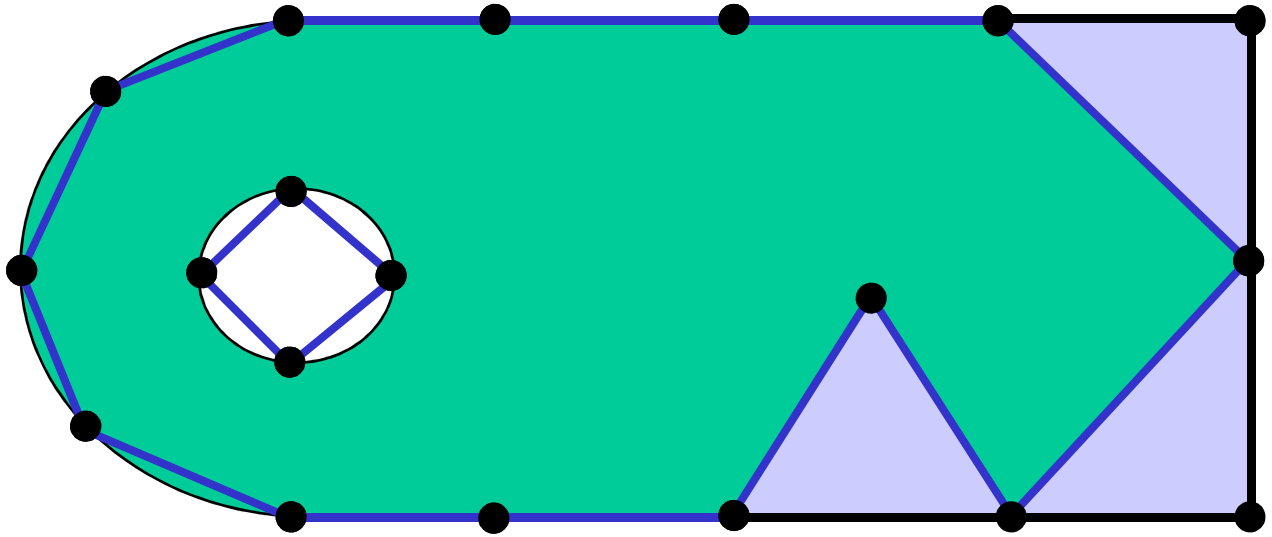
➢ Determine if any other nodes on current front are within search radius *r* of ideal location C (Choose D instead of C)

Advancing Front



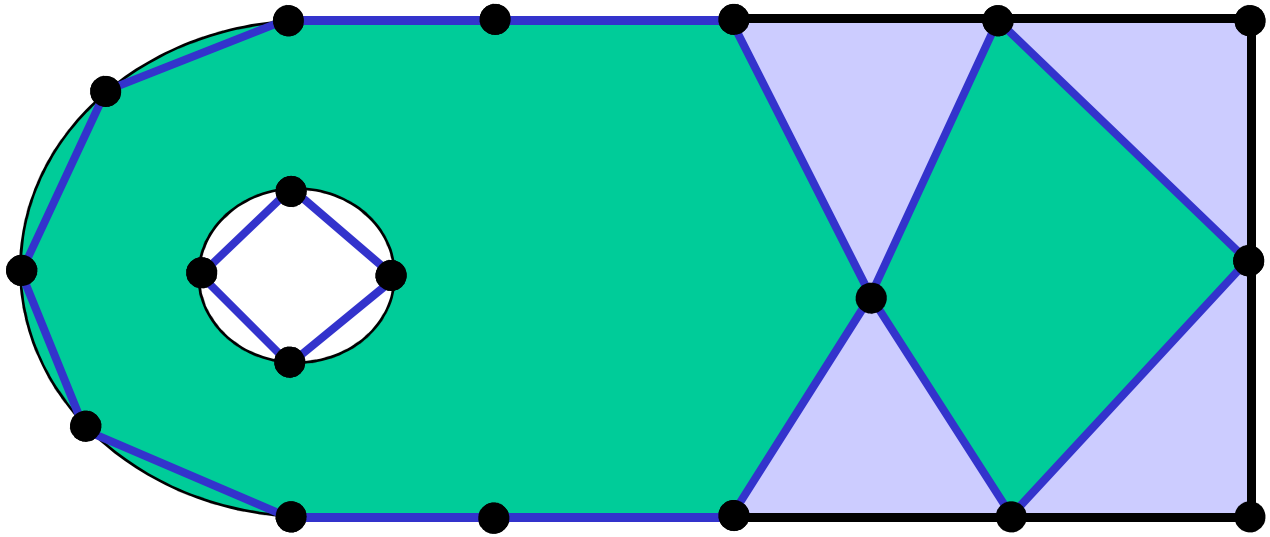
- Book-Keeping: New *front edges* added and deleted from *front* as triangles are formed
- Continue until no *front edges* remain on *front*

Advancing Front



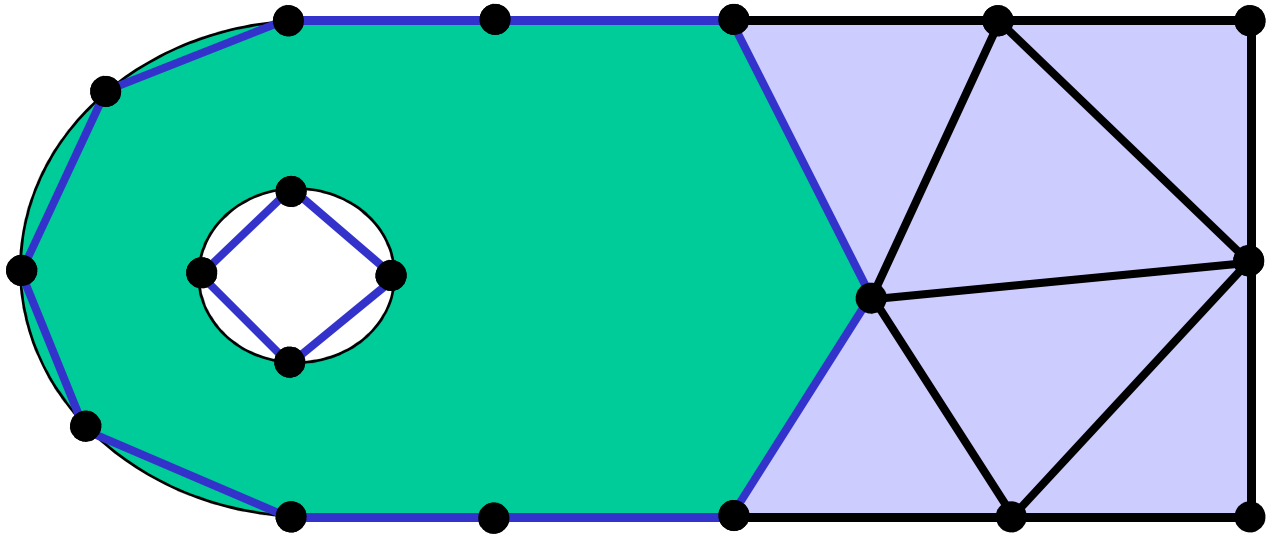
- Book-Keeping: New *front edges* added and deleted from *front* as triangles are formed
- Continue until no *front edges* remain on *front*

Advancing Front



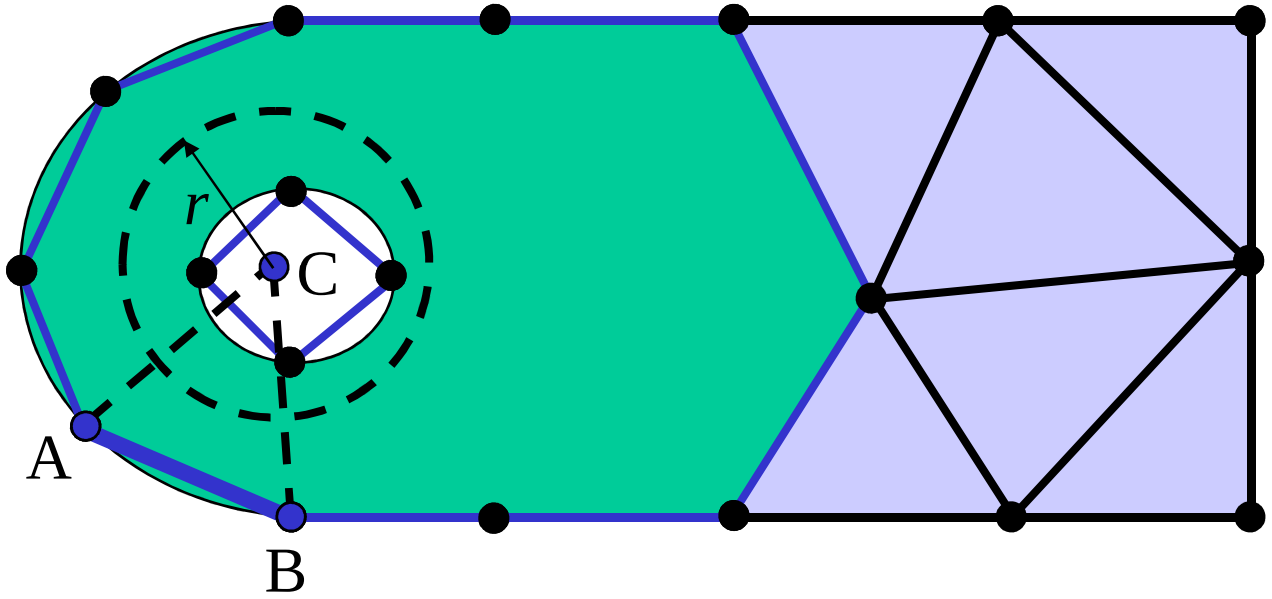
- Book-Keeping: New *front edges* added and deleted from *front* as triangles are formed
- Continue until no *front edges* remain on *front*

Advancing Front



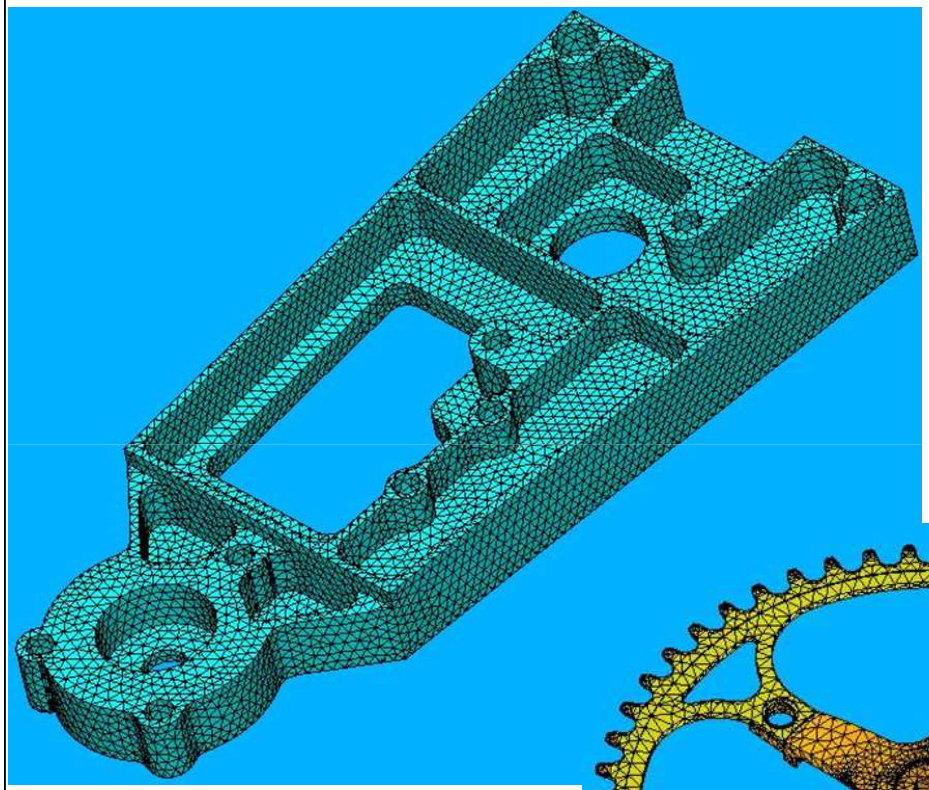
- Book-Keeping: New *front edges* added and deleted from *front* as triangles are formed
- Continue until no *front edges* remain on *front*

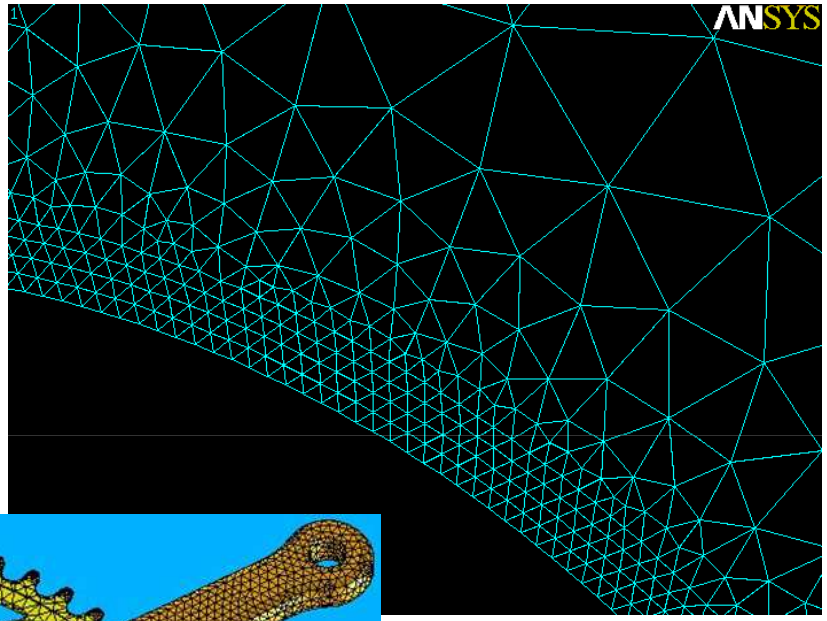
Advancing Front

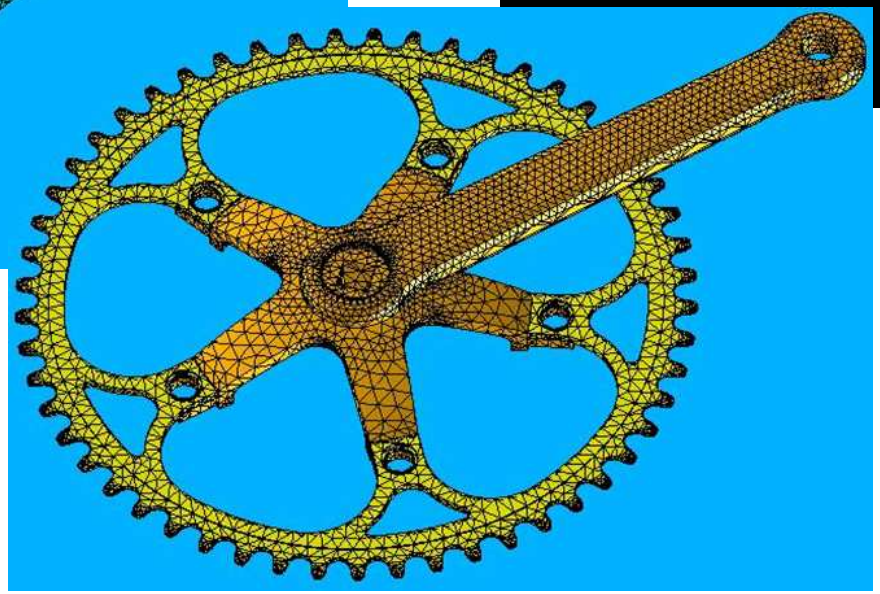


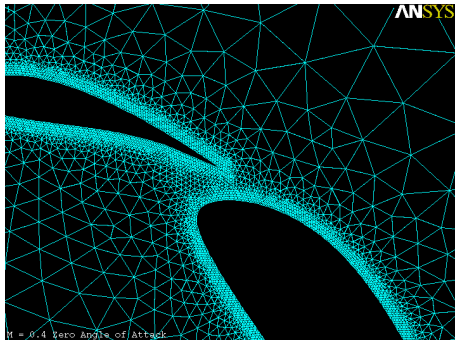
- Where multiple choices are available, use best quality (closest shape to equilateral)
- Reject any that would intersect existing front
- Reject any inverted triangles (|AB X AC| > 0)
- (Lohner,88;96)(Lo,91)

Advancing Front



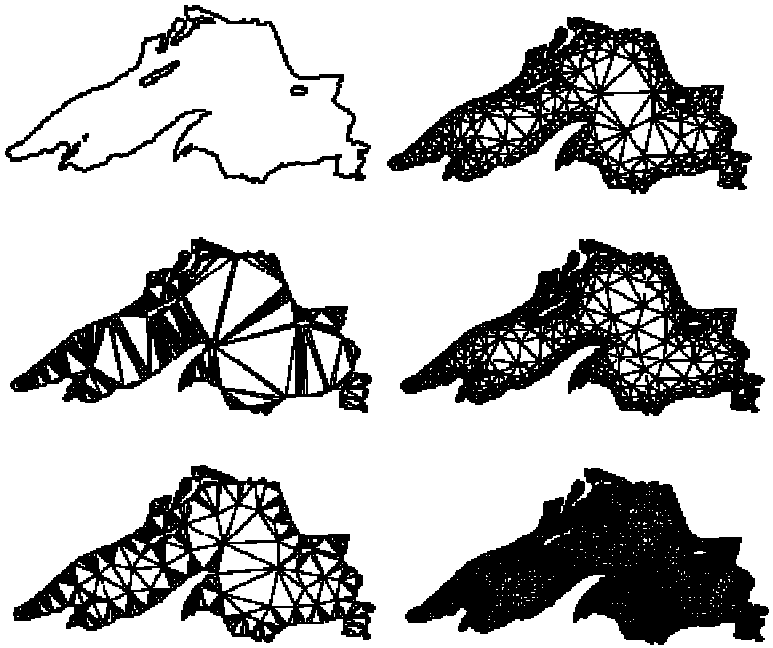




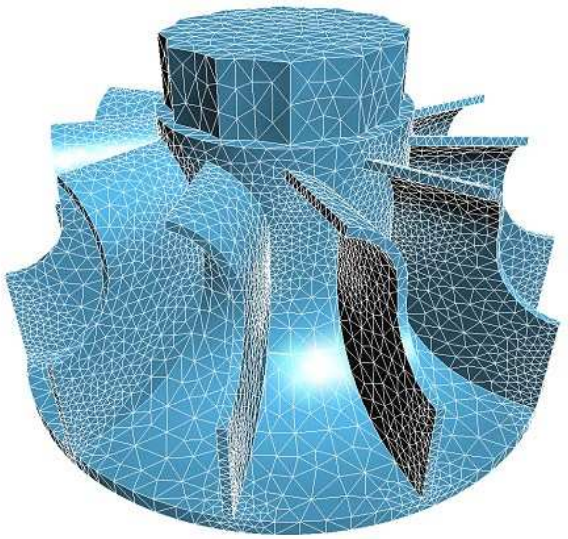


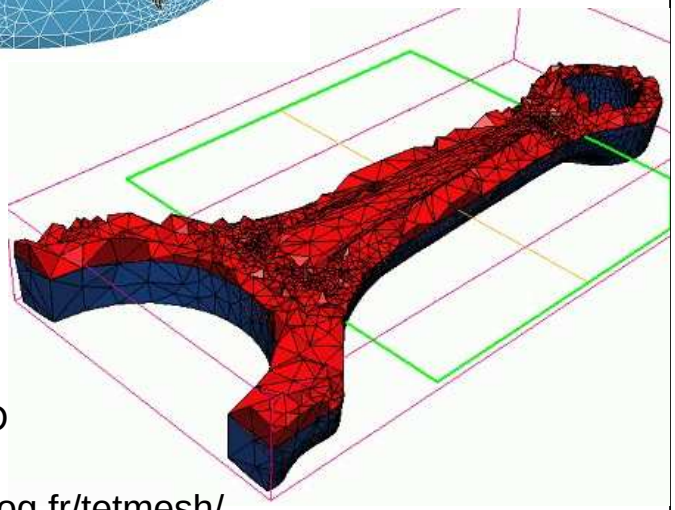
Ansys, Inc.
www.ansys.com

Delaunay



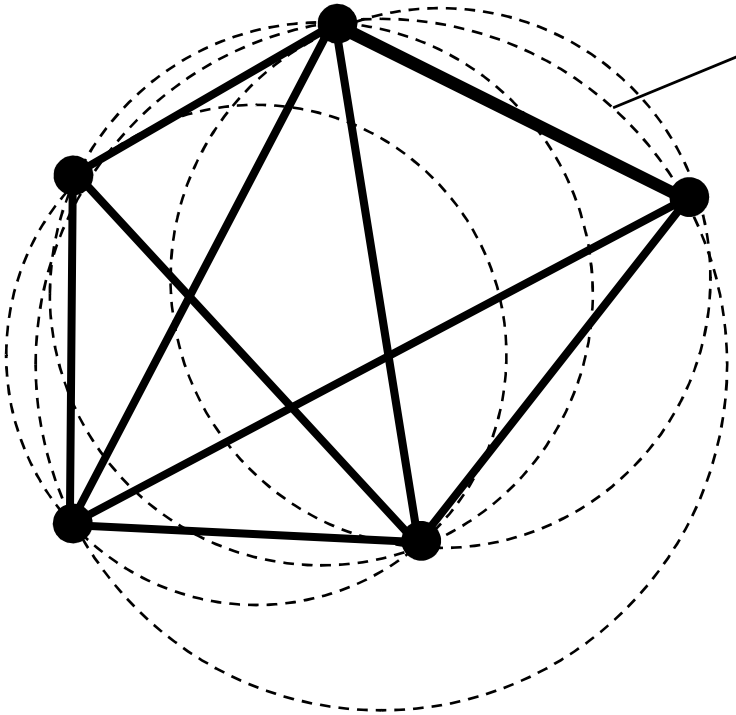
Triangle
Jonathon Shewchuk
http://www-2.cs.cmu.edu/~quake/triangle.html





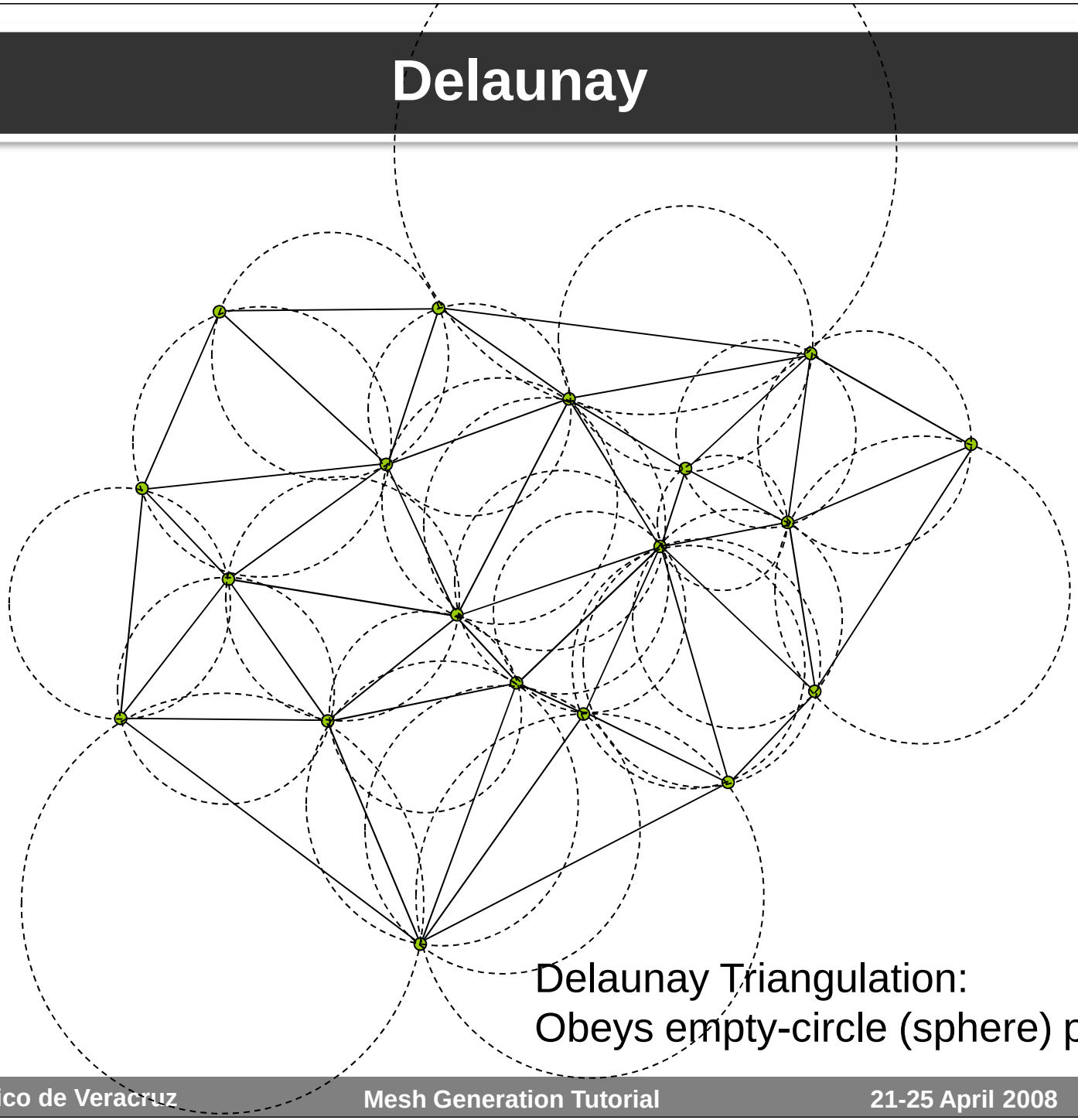
Tetmesh-GHS3D
INRIA, France
http://www.simulog.fr/tetmesh/

Delaunay



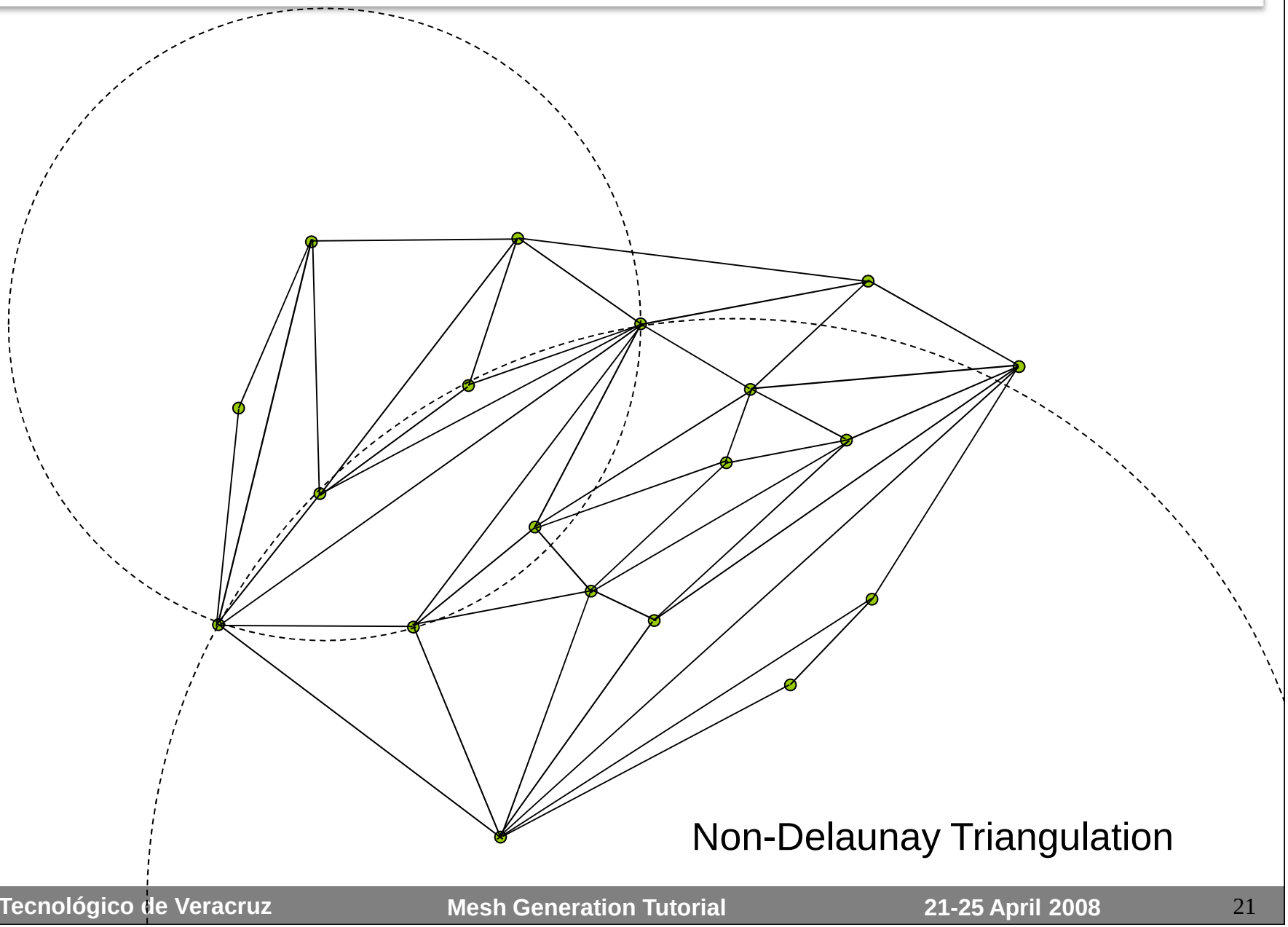
Empty Circle (Sphere) Property:
No other vertex is contained within the circumcircle (circumsphere) of any triangle (tetrahedron)

Delaunay



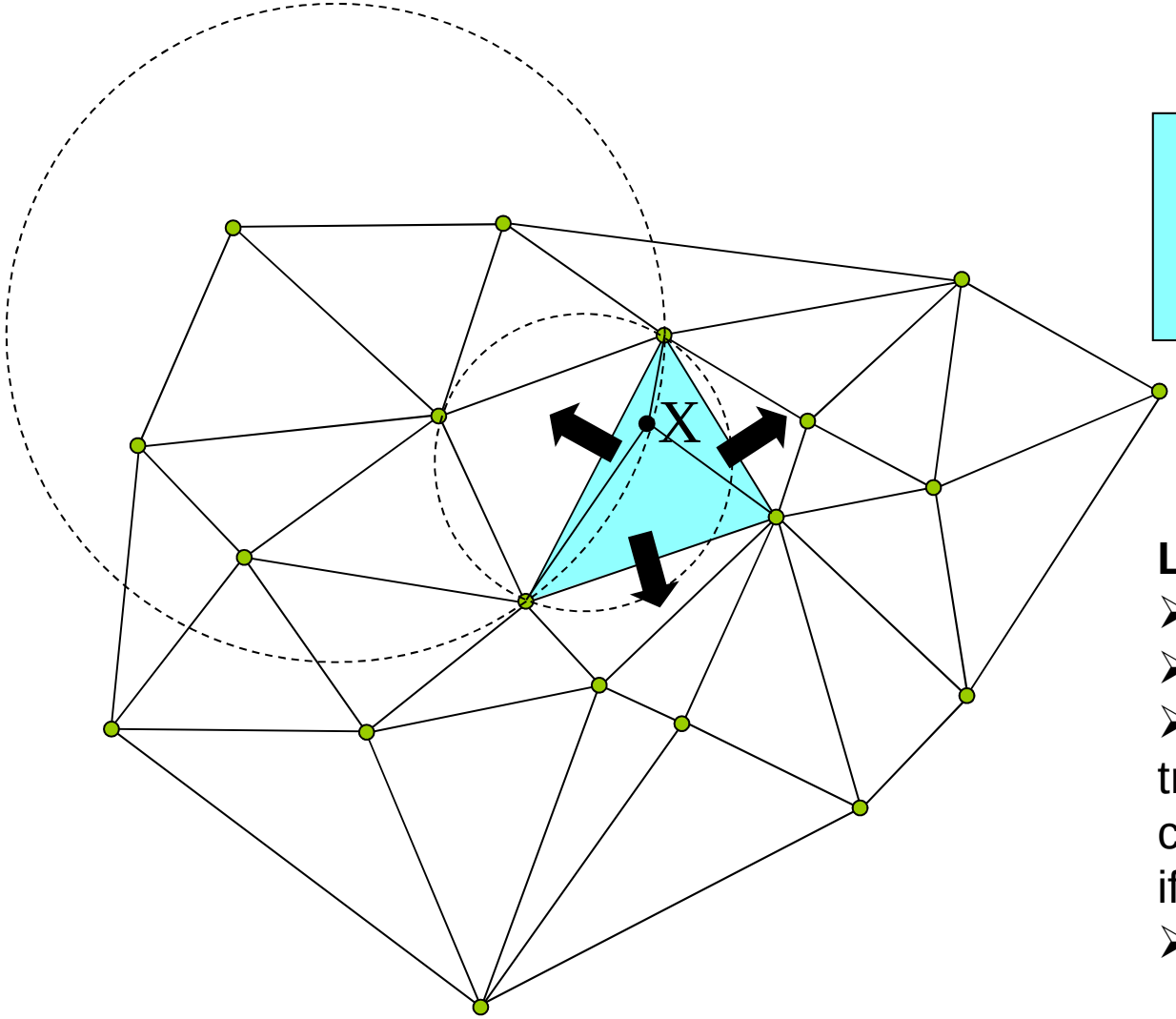
Delaunay Triangulation:
Obeys empty-circle (sphere) property

Delaunay



Non-Delaunay Triangulation

Delaunay

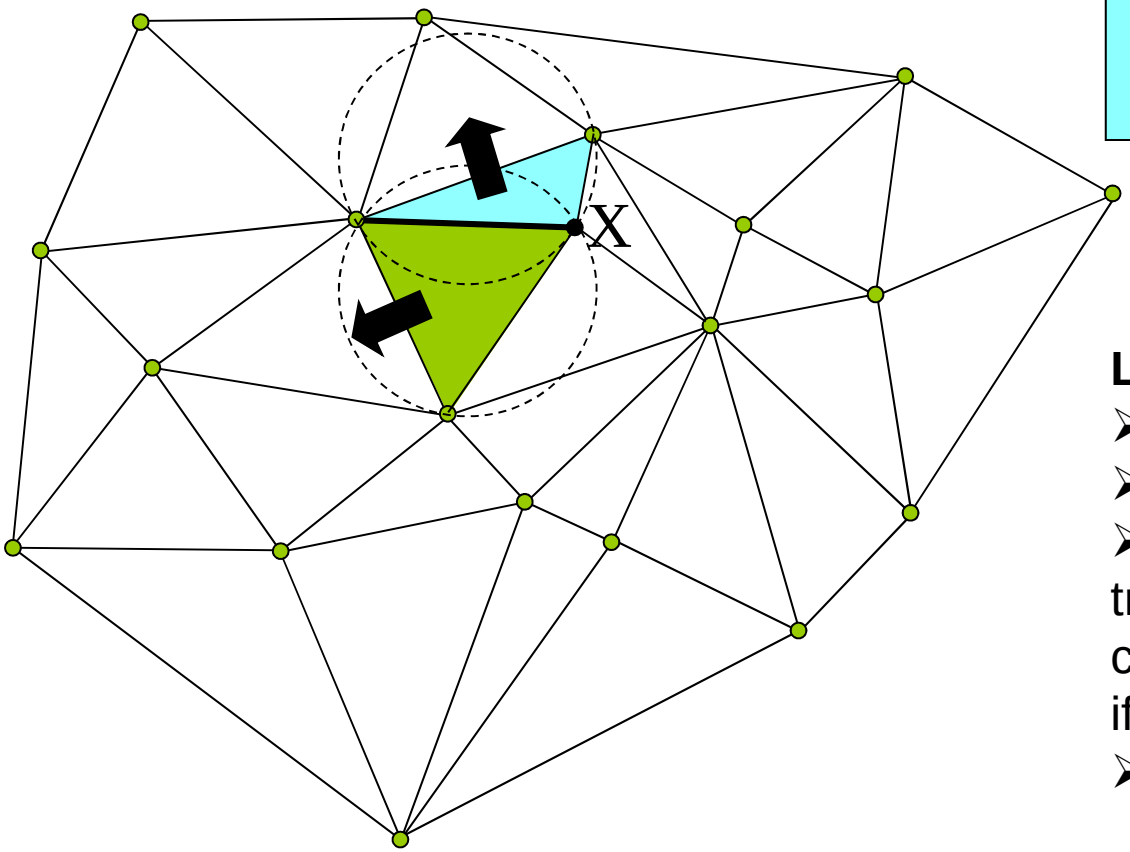


Given a Delaunay Triangulation of n nodes, How do I insert node n+1 ?

Lawson Algorithm

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

Delaunay

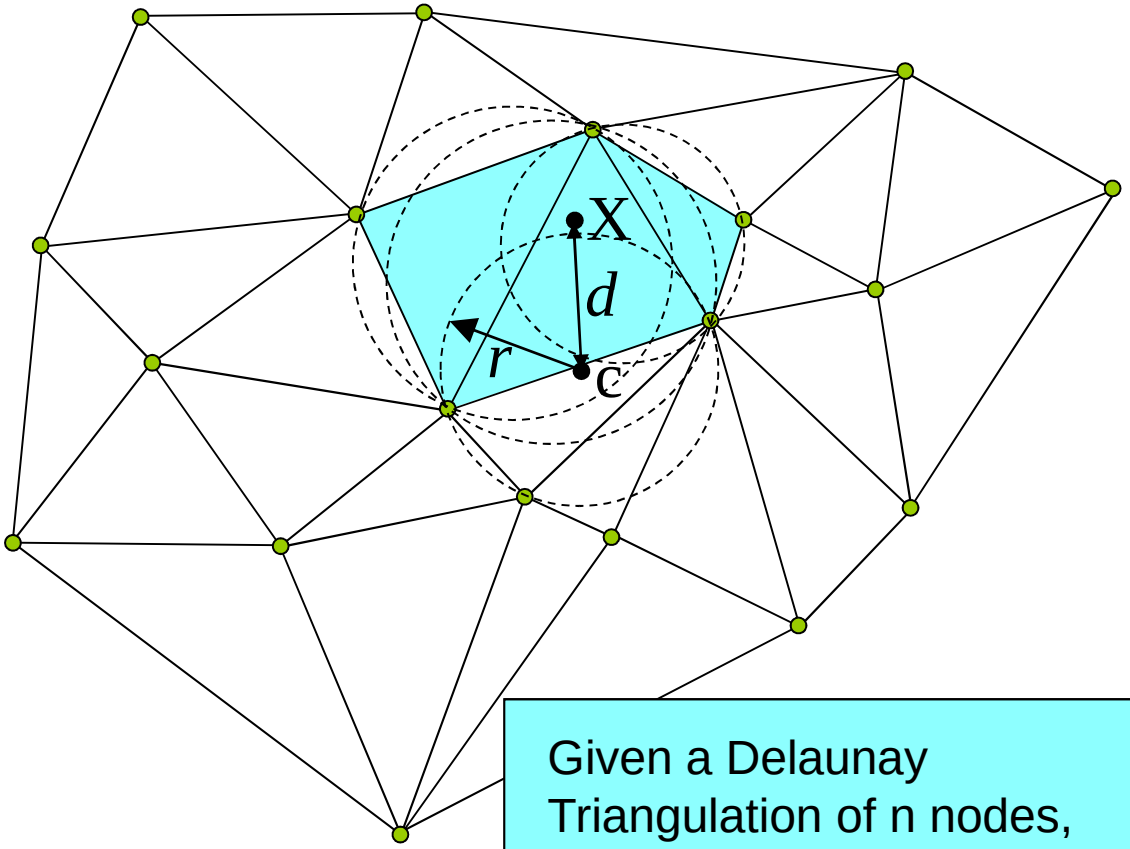


Given a Delaunay Triangulation of n nodes, How do I insert node n+1 ?

Lawson Algorithm

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

Delaunay

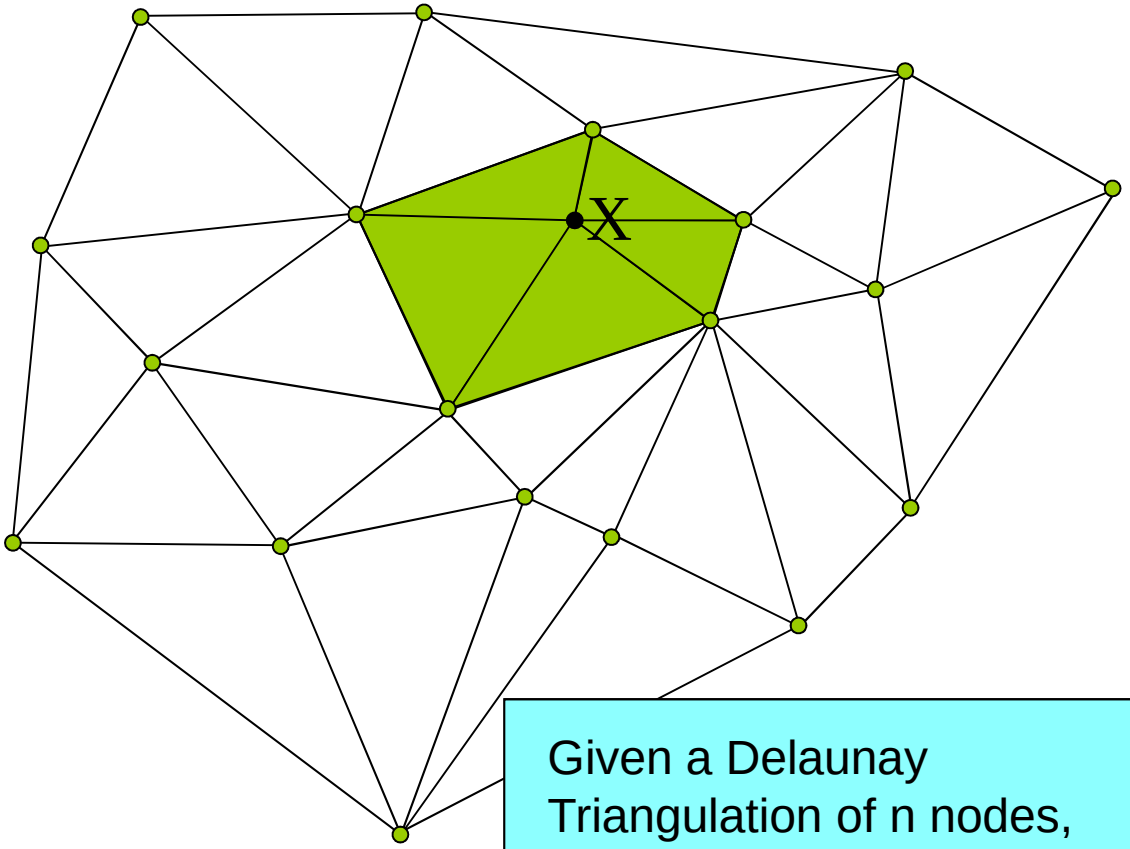


Bowyer-Watson Algorithm

- 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)

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

Delaunay

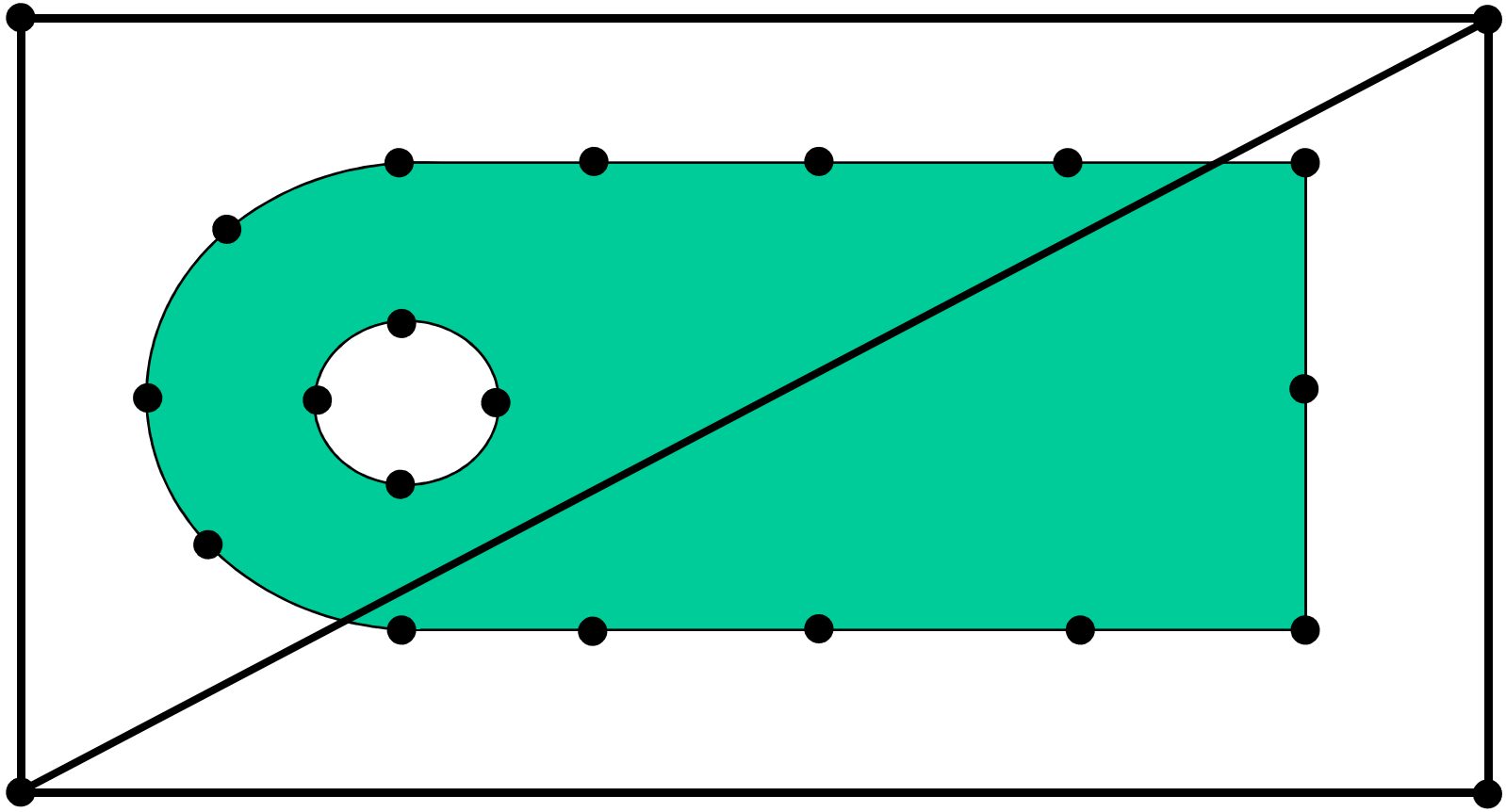


Given a Delaunay Triangulation of n nodes, How do I insert node n+1 ?

Bowyer-Watson Algorithm

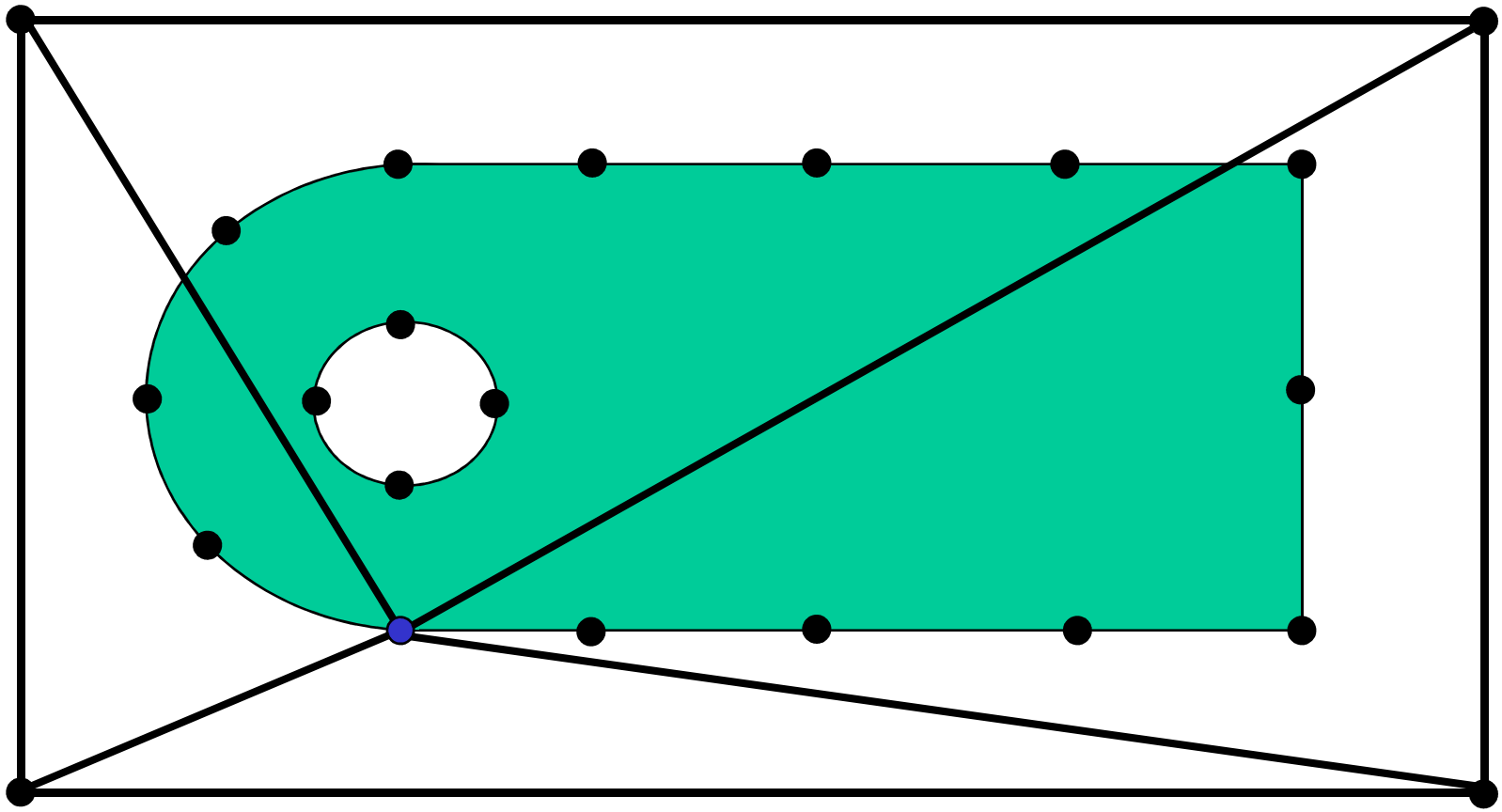
- Locate triangle that contains the point
- Search for all triangles whose circumcircle contain the point ($d<r$)
- Delete the triangles (creating a void in the mesh)
- Form new triangles from the new point and the void boundary
- (Watson,81)

Delaunay



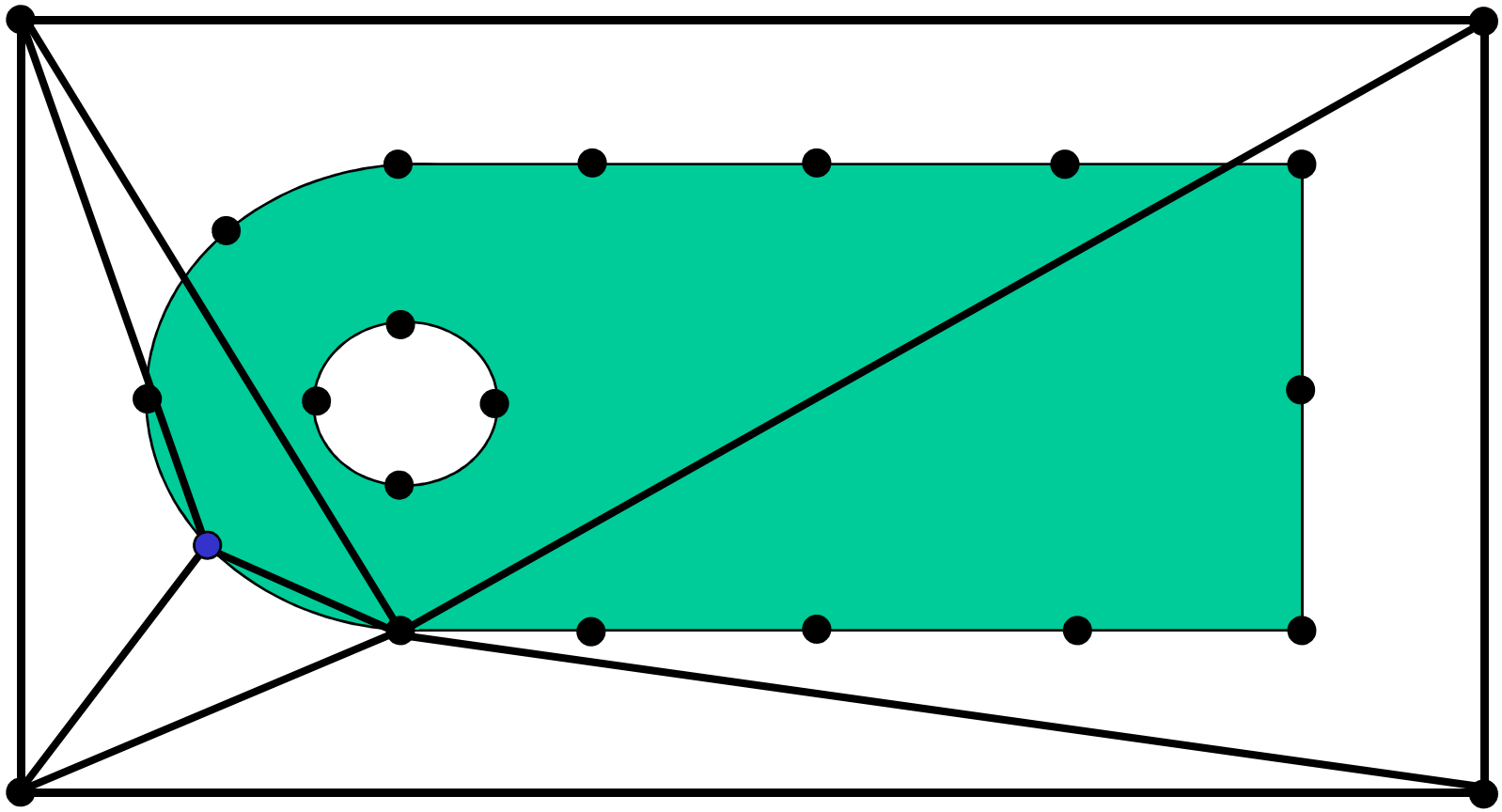
- Begin with Bounding Triangles (or Tetrahedron)

Delaunay



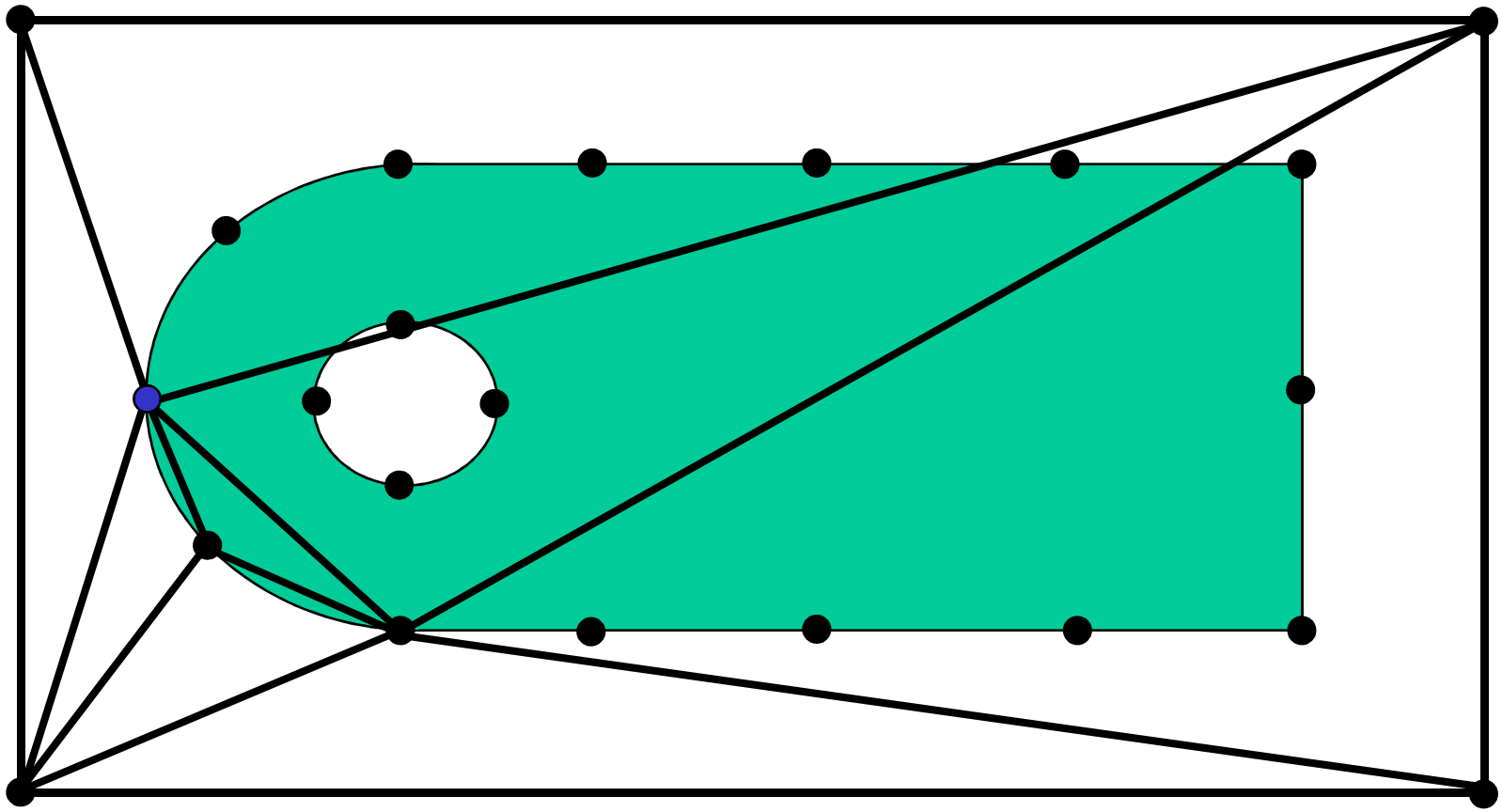
- Insert boundary nodes using Delaunay method (Lawson or Bowyer-Watson)

Delaunay



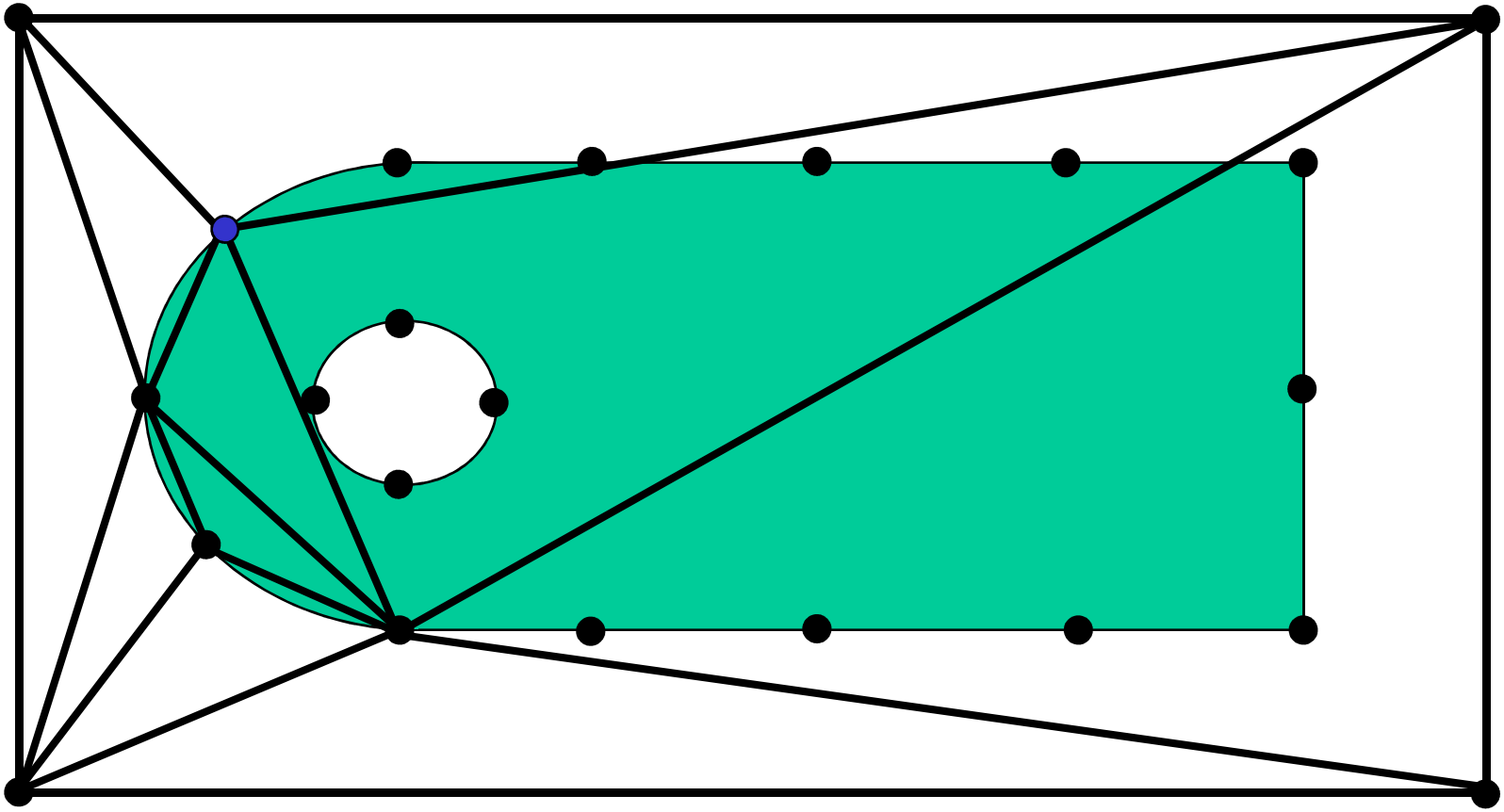
- Insert boundary nodes using Delaunay method (Lawson or Bowyer-Watson)

Delaunay



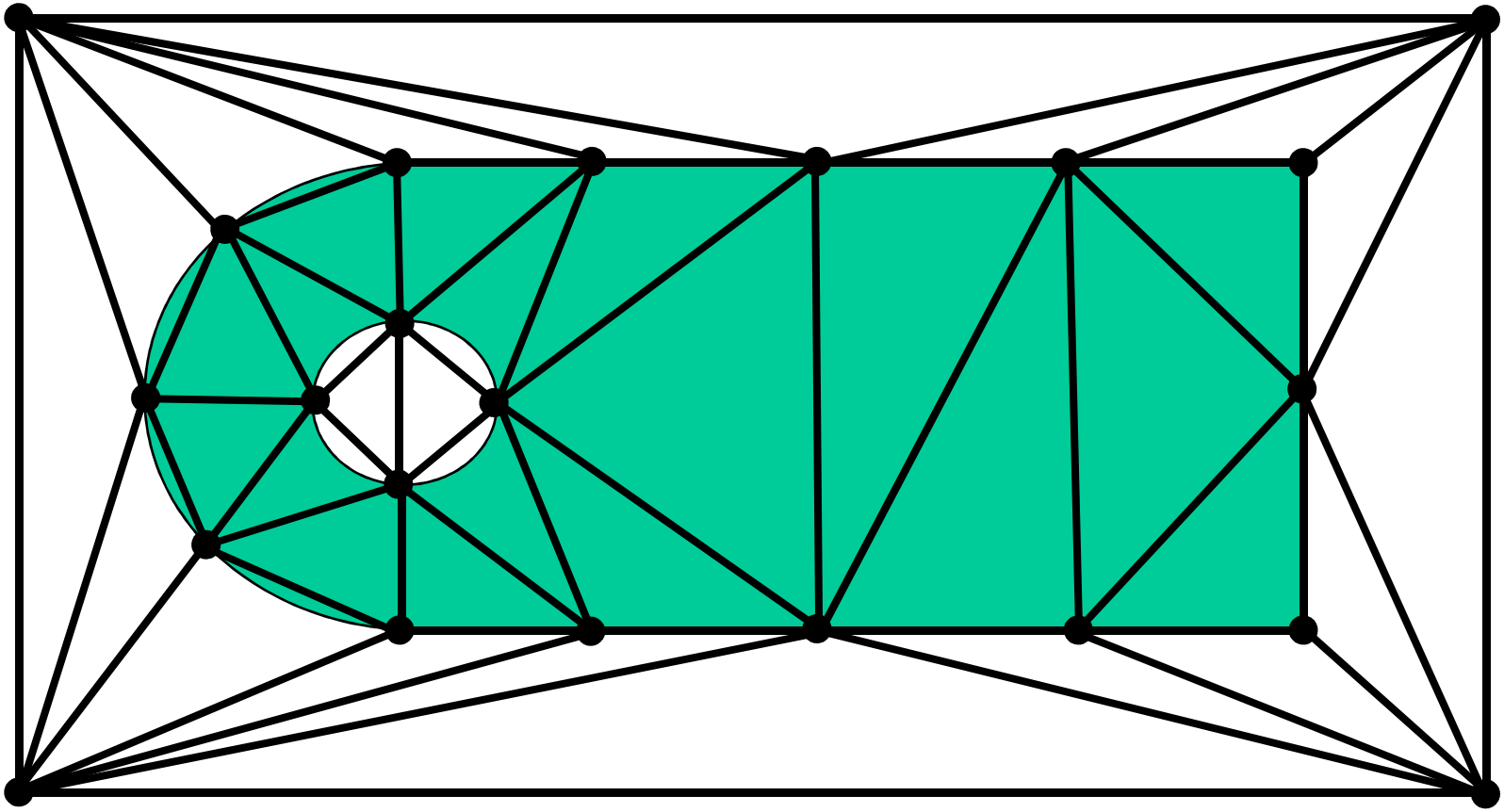
- Insert boundary nodes using Delaunay method (Lawson or Bowyer-Watson)

Delaunay



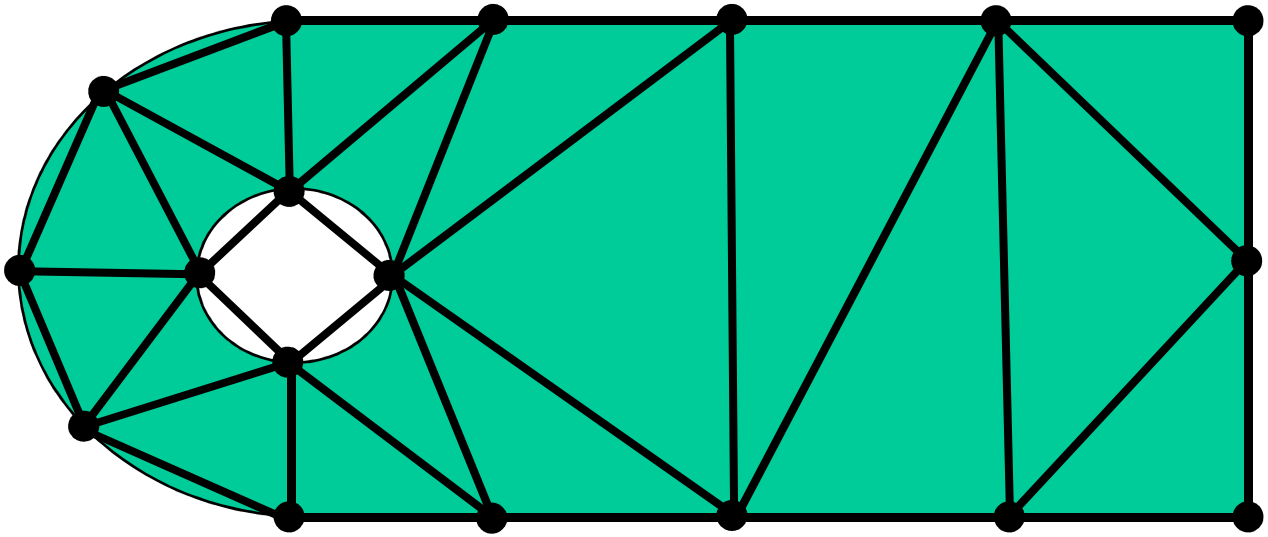
- Insert boundary nodes using Delaunay method (Lawson or Bowyer-Watson)

Delaunay



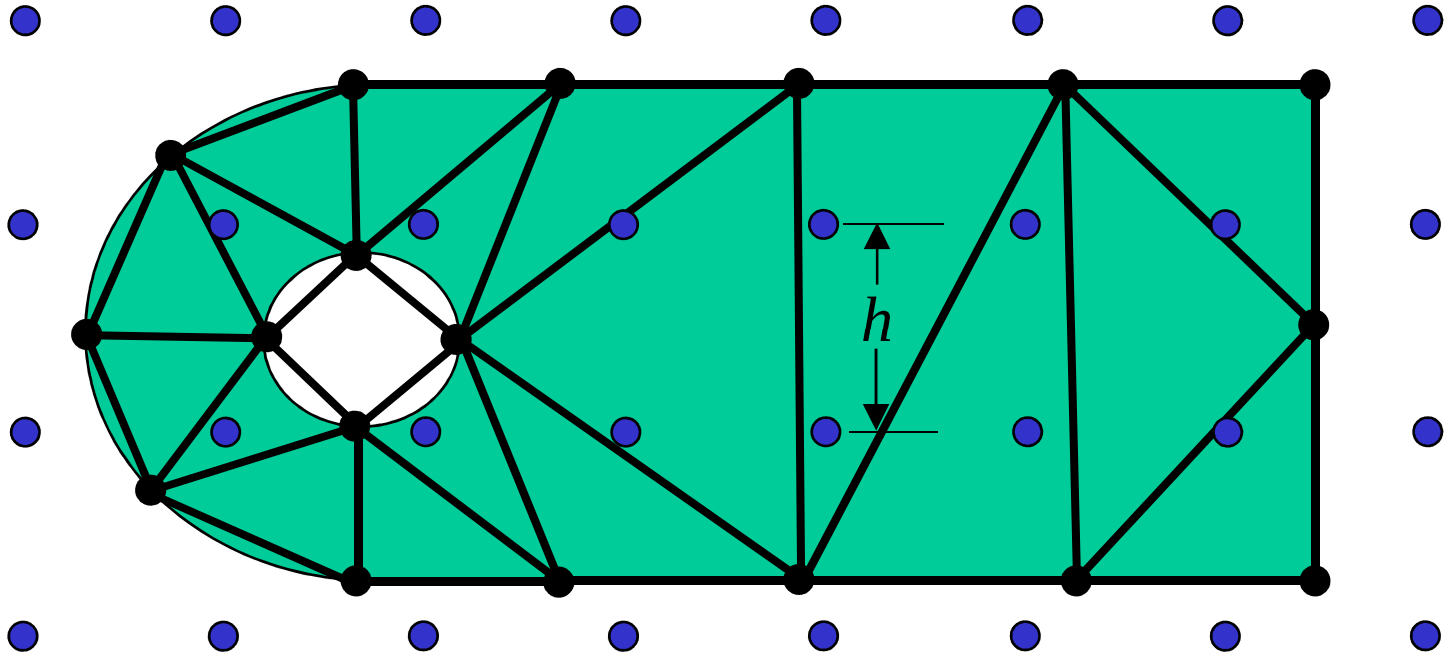
- Insert boundary nodes using Delaunay method (Lawson or Bowyer-Watson)

Delaunay



- Recover boundary
- Delete outside triangles
- Insert internal nodes

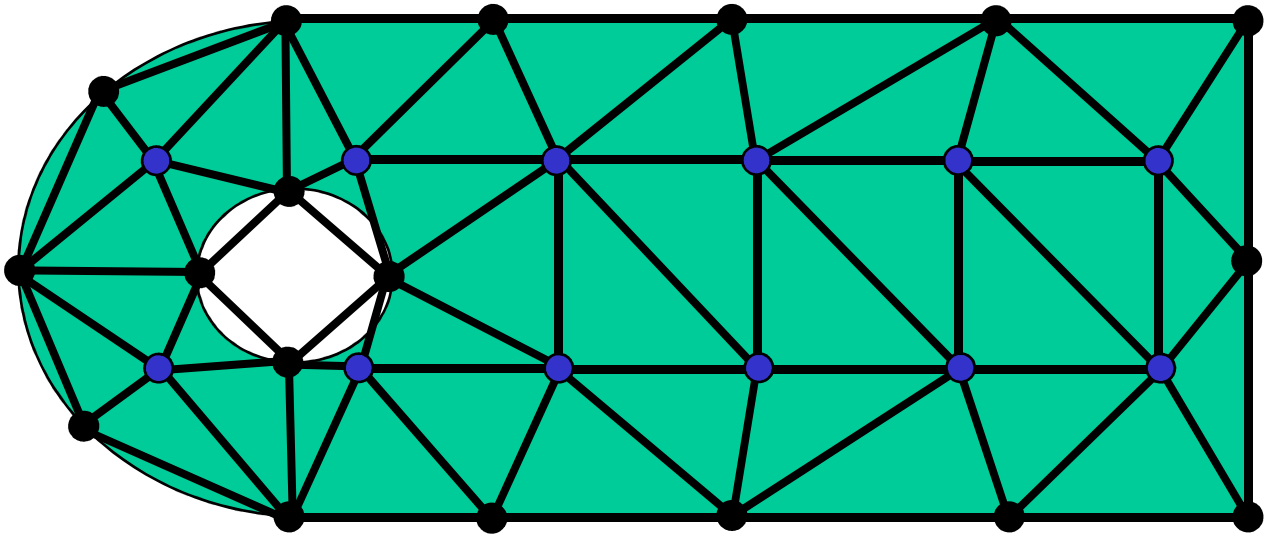
Delaunay: Node Insertion



Grid Based

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

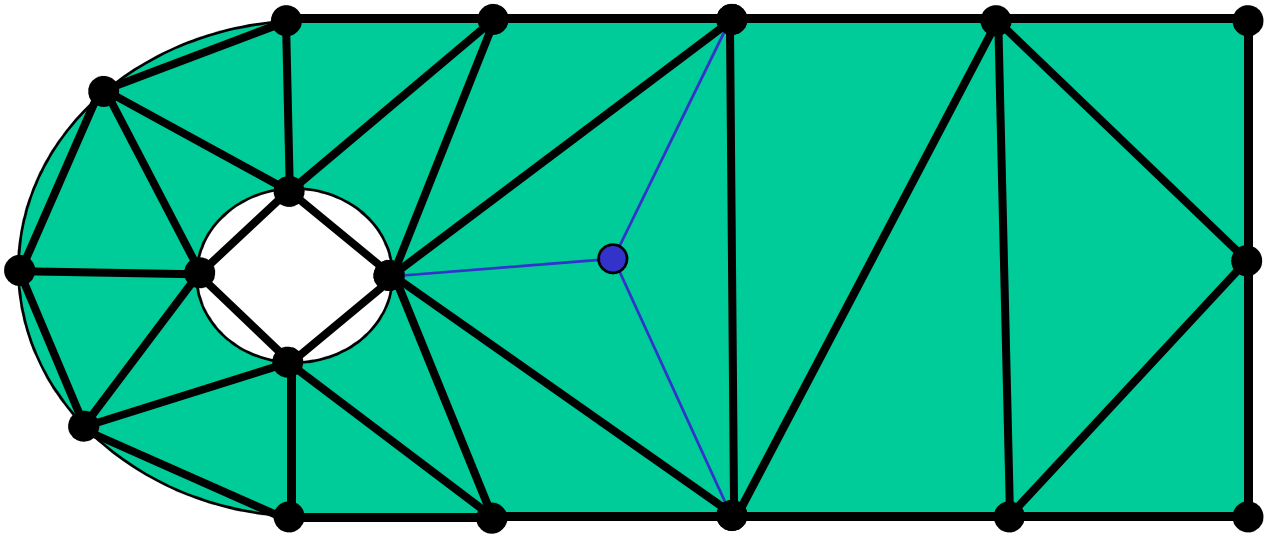
Delaunay: Node Insertion



Grid Based

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

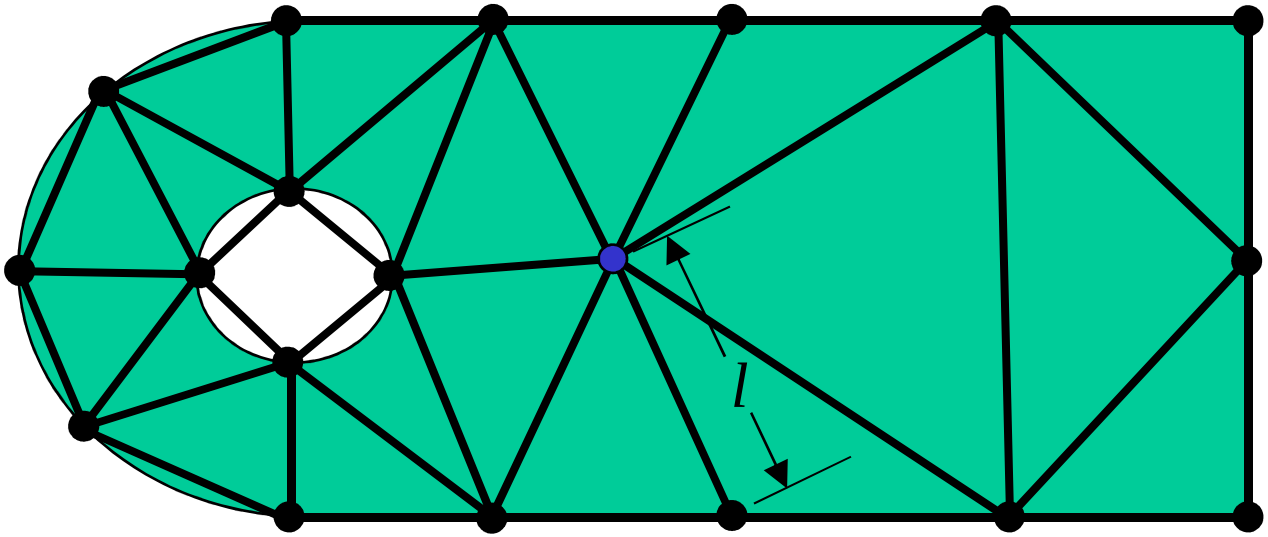
Delaunay: Node Insertion



Centroïd

- Nodes introduced at triangle centroïds
- Continues until edge length, $l \approx h$

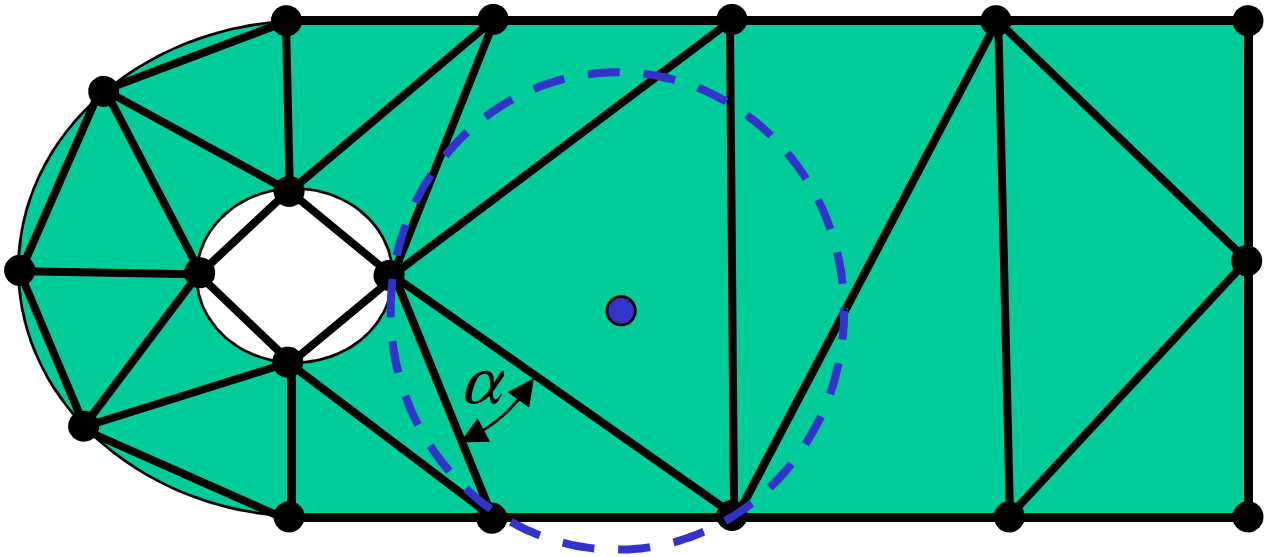
Delaunay: Node Insertion



Centroïd

- Nodes introduced at triangle centroïds
- Continues until edge length, $l \approx h$

Delaunay: Node Insertion

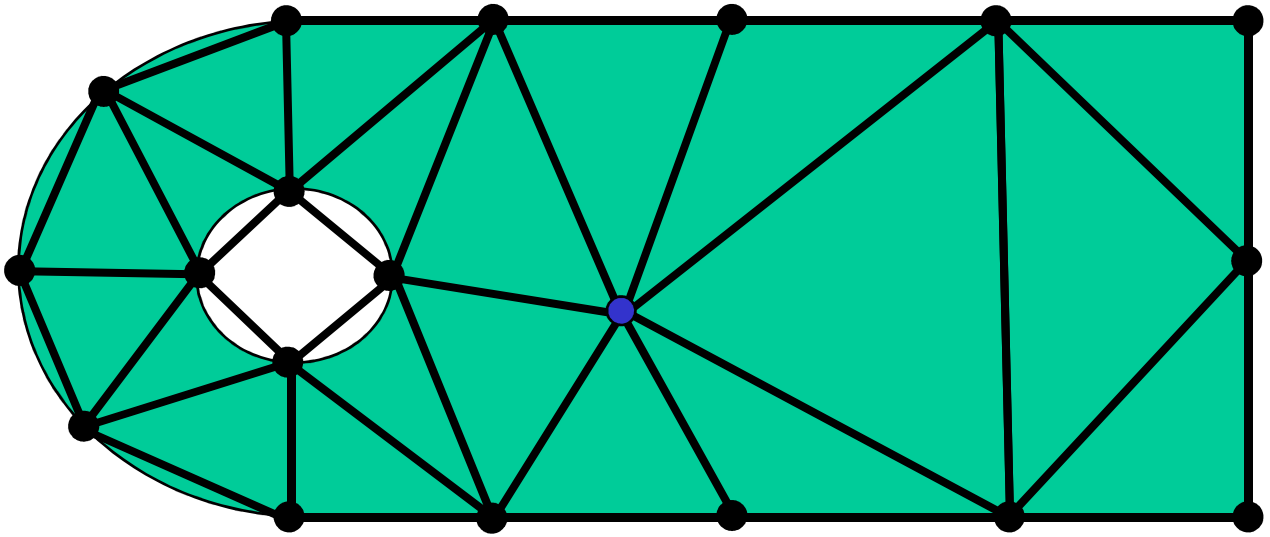


Circumcenter (“Guaranteed Quality”)

- 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: Node Insertion

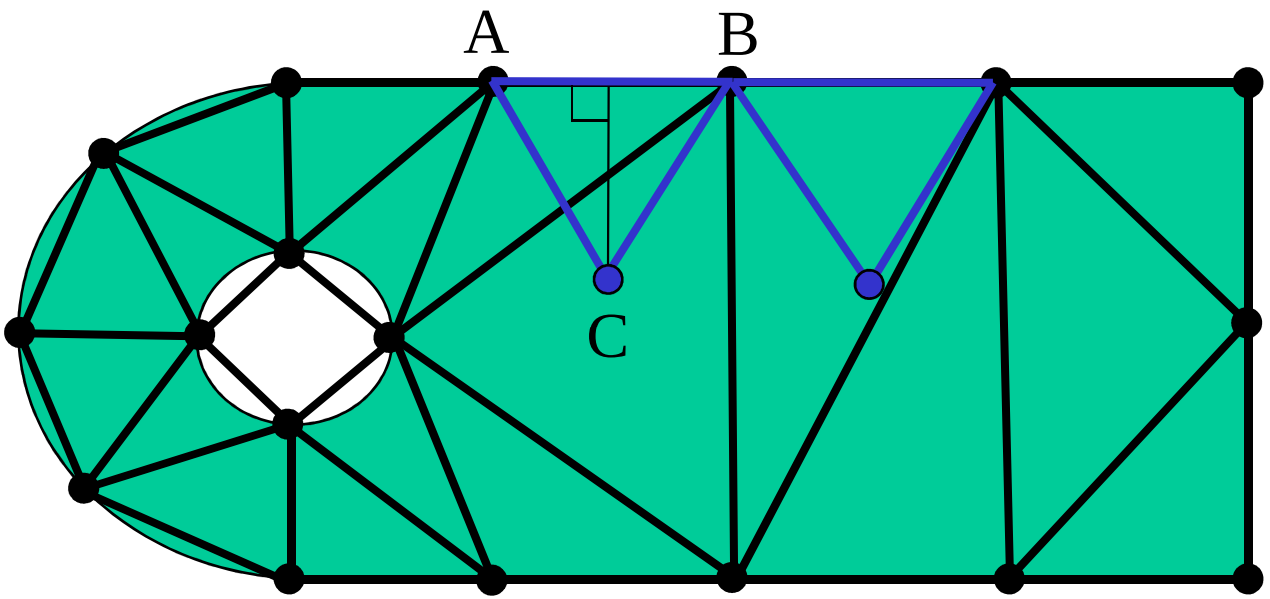


Circumcenter (“Guaranteed Quality”)

- 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: Node Insertion

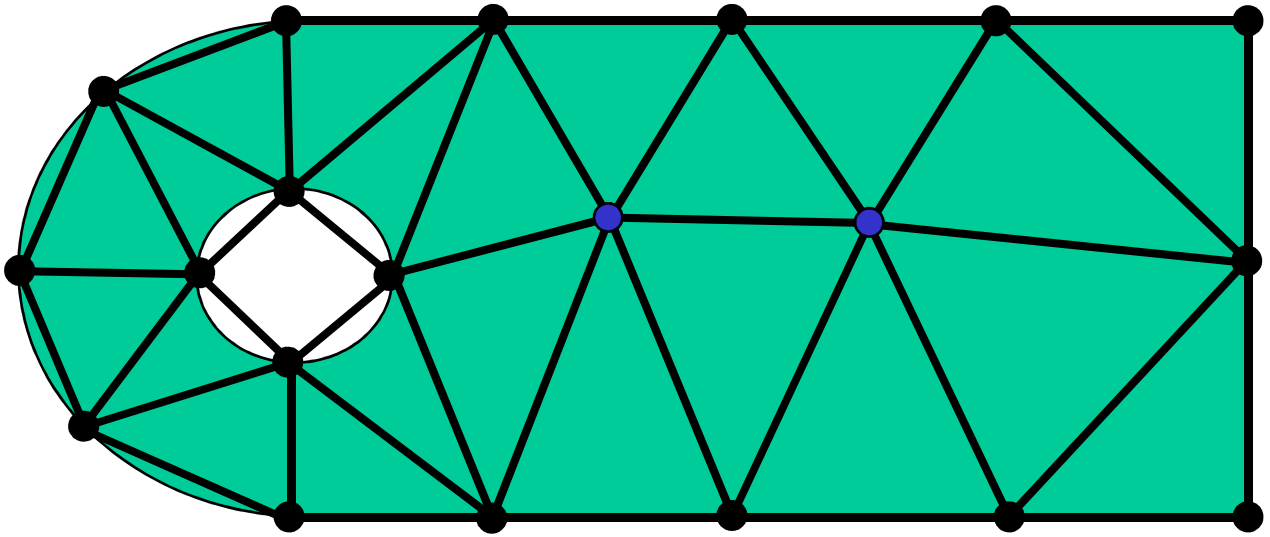


Advancing Front

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

(Marcum,95)

Delaunay: Node Insertion

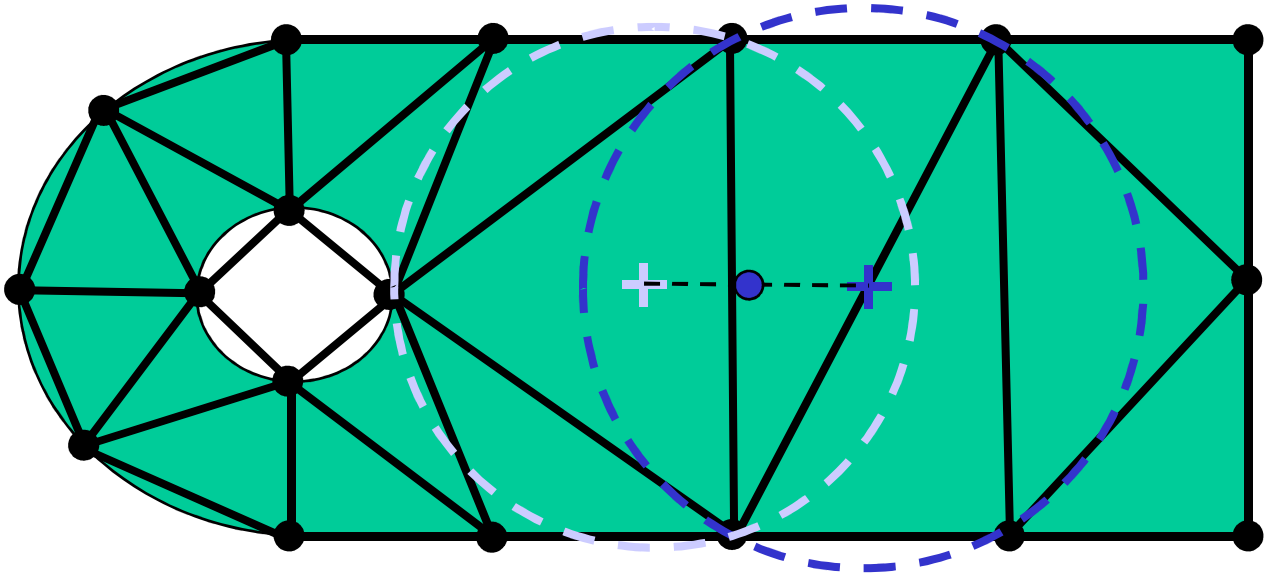


Advancing Front

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

(Marcum,95)

Delaunay: Node Insertion

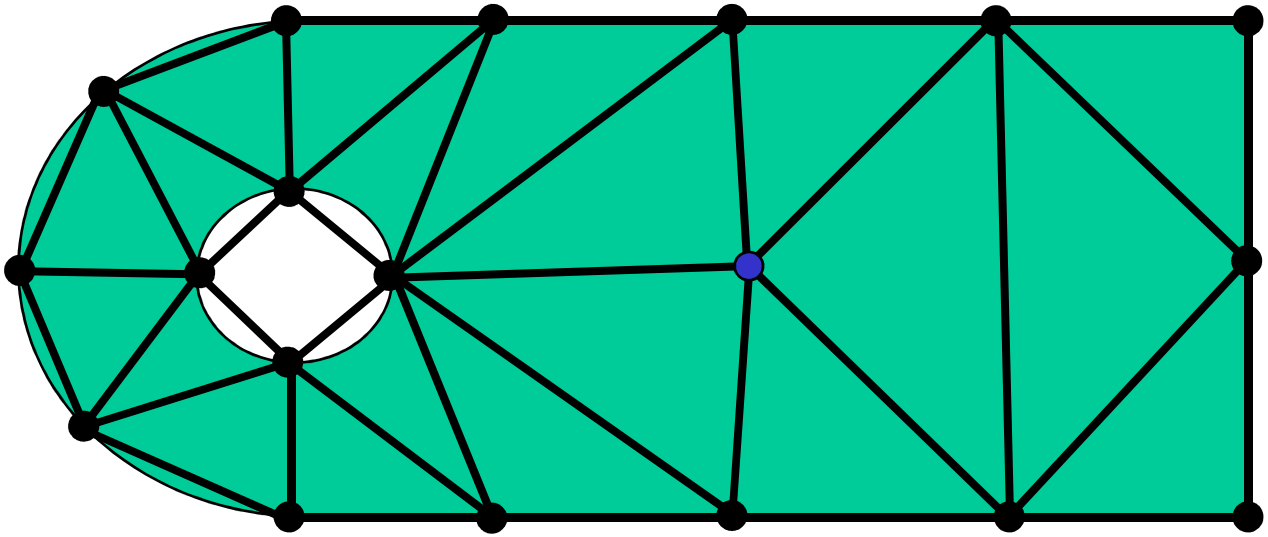


Voronoï-Segment

➢ Nodes introduced at midpoint of segment connecting the circumcircle centers of two adjacent triangles

(Rebay,93)

Delaunay: Node Insertion

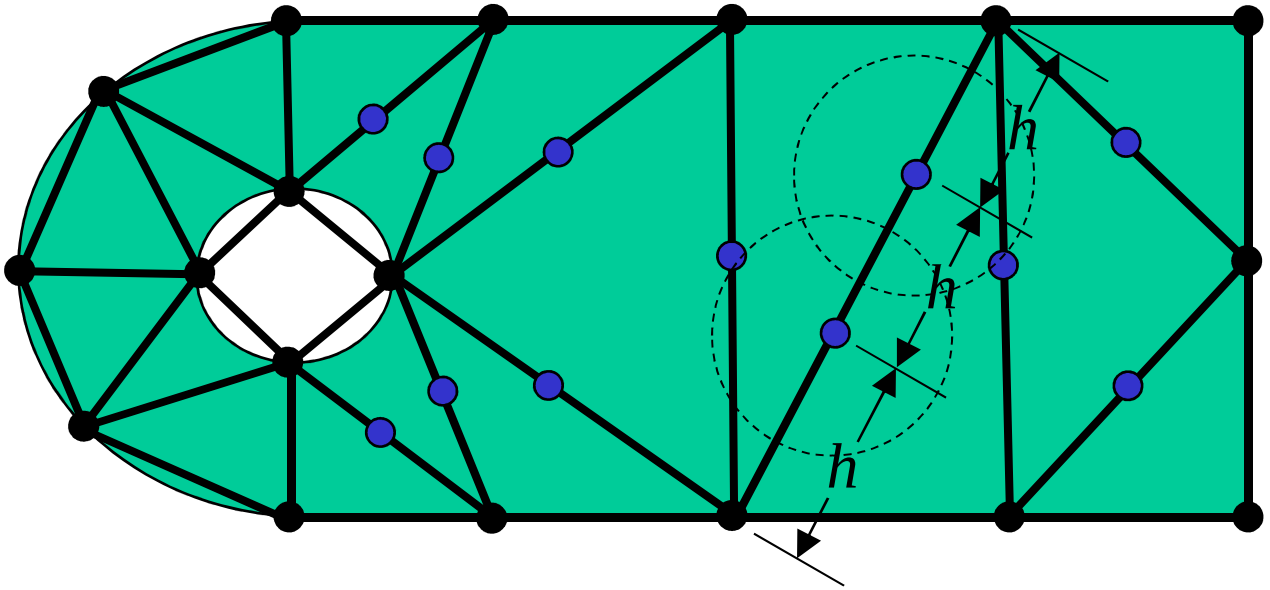


Voronoï-Segment

➢ Nodes introduced at midpoint of segment connecting the circumcircle centers of two adjacent triangles

(Rebay,93)

Delaunay: Node Insertion

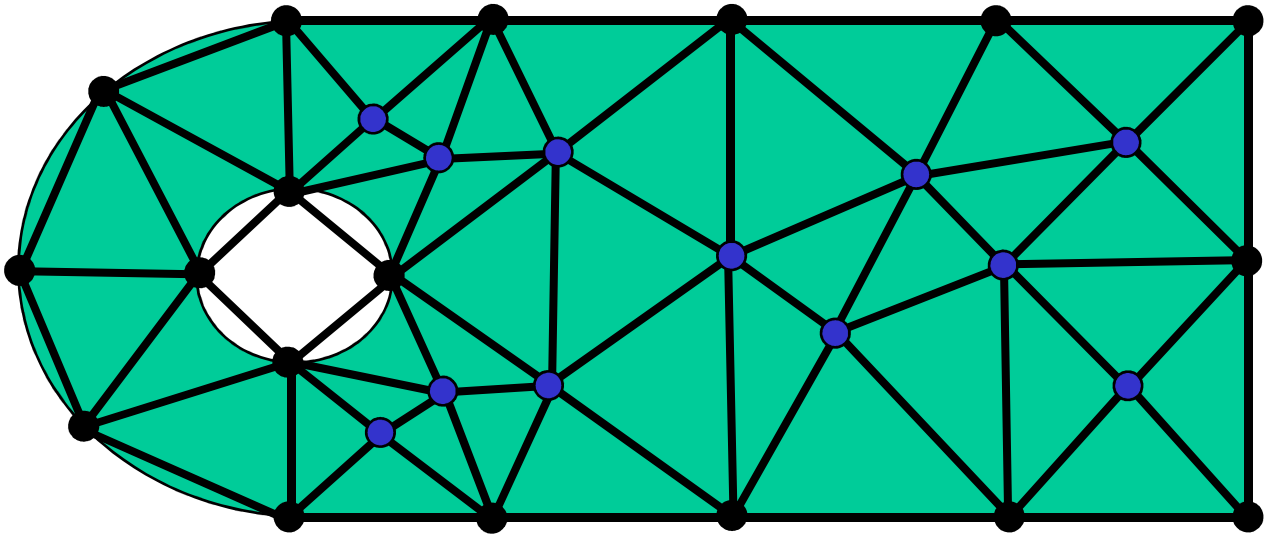


Edges

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

(George,91)

Delaunay: Node Insertion

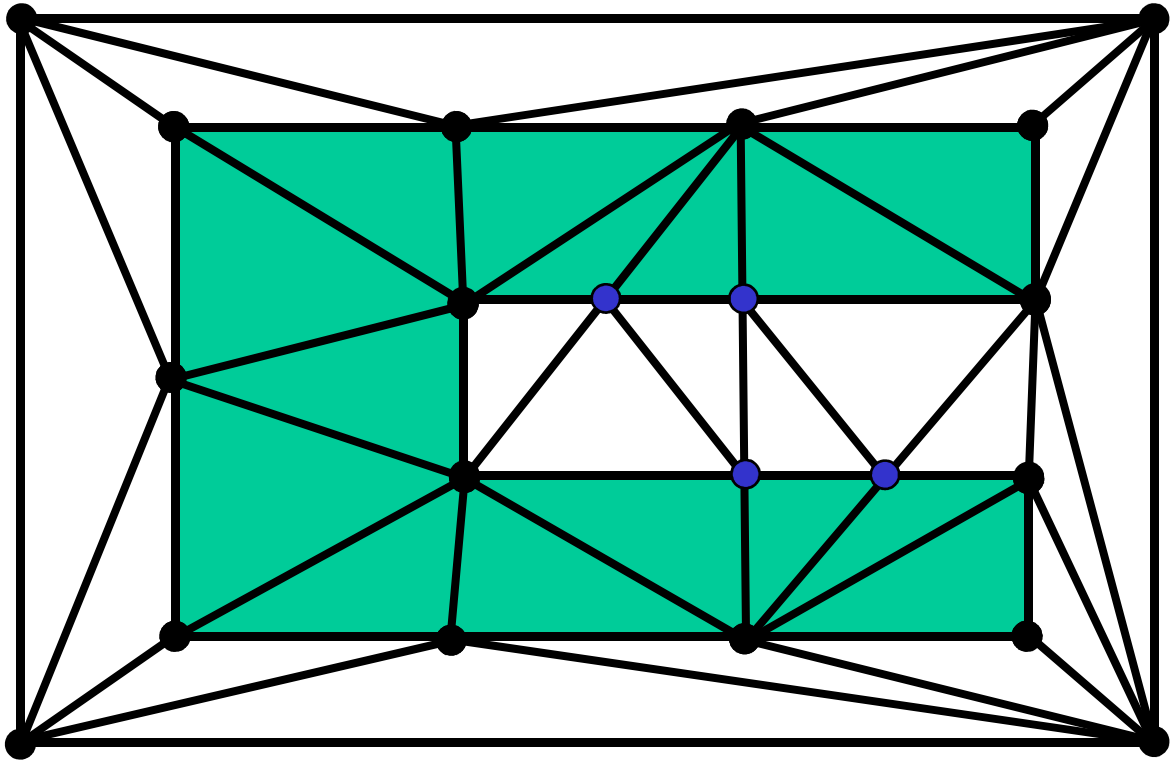


Edges

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

(George,91)

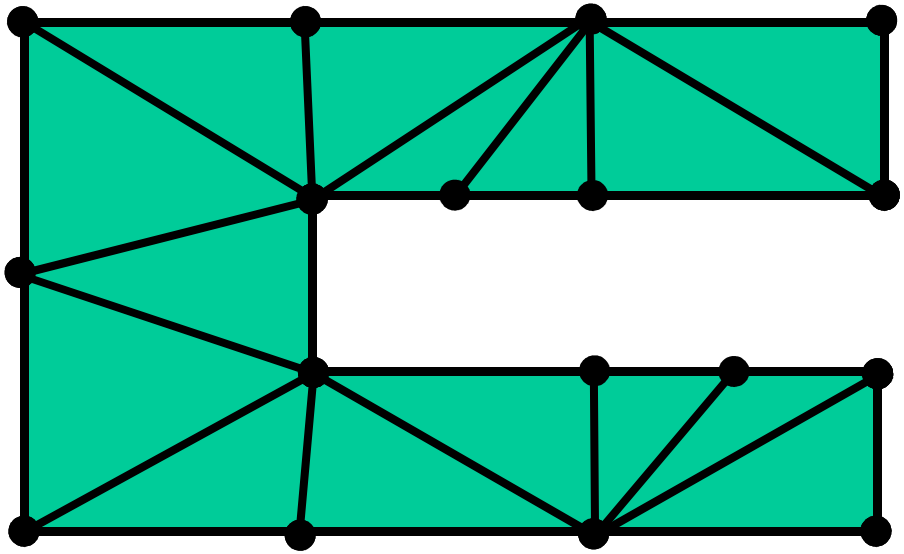
Delaunay: Boundary Constrained



Boundary Intersection

- Nodes and edges introduced where Delaunay edges intersect boundary

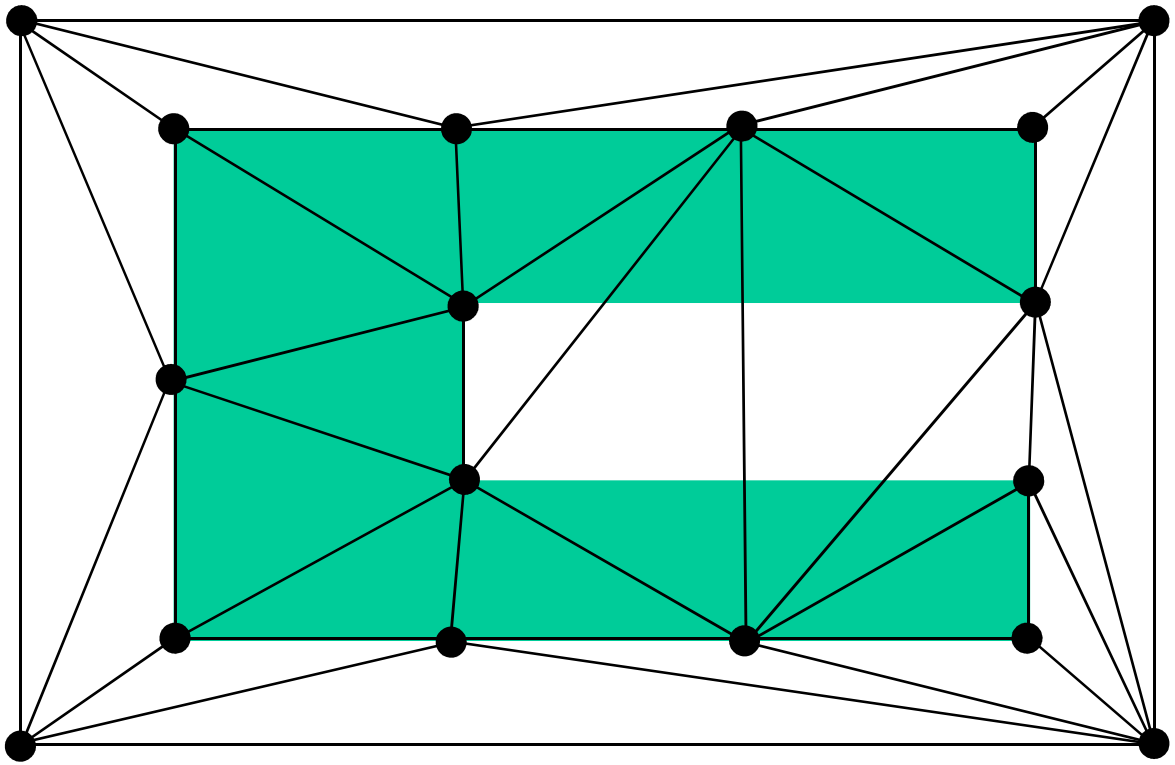
Delaunay: Boundary Constrained



Boundary Intersection

- Nodes and edges introduced where Delaunay edges intersect boundary

Delaunay: Boundary Constrained

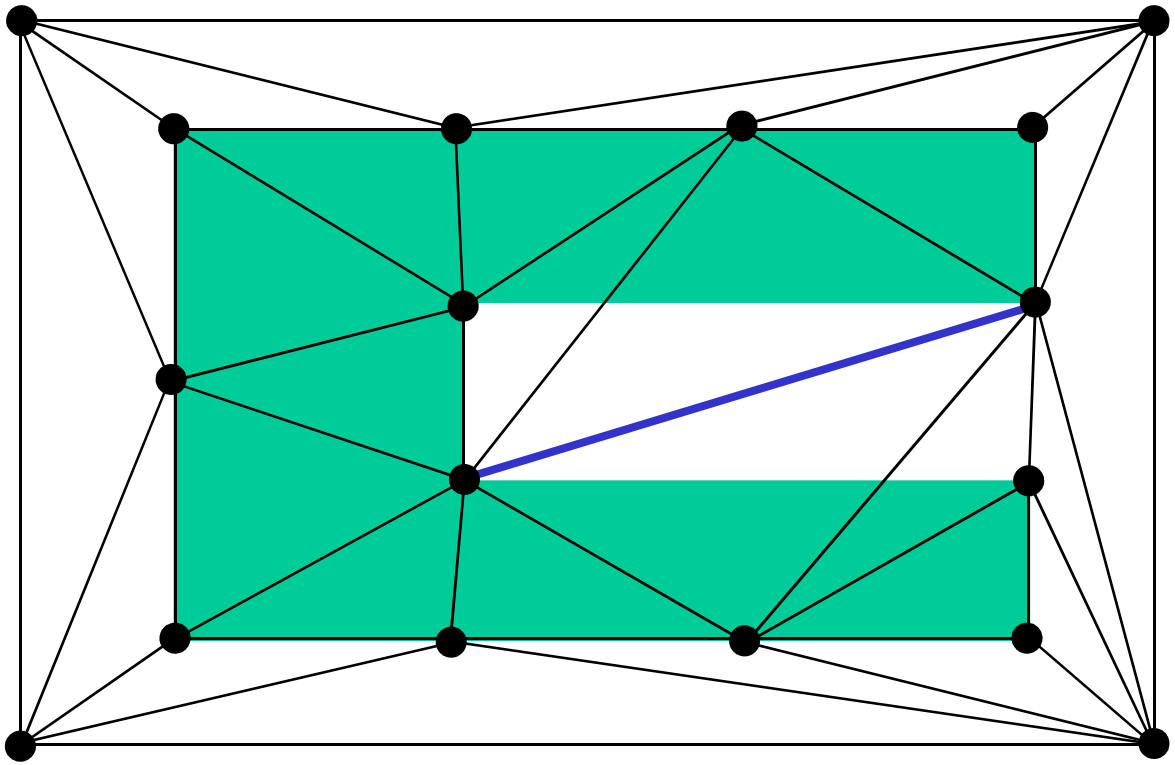


Local Swapping

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

(George,91)(Owen,99)

Delaunay: Boundary Constrained

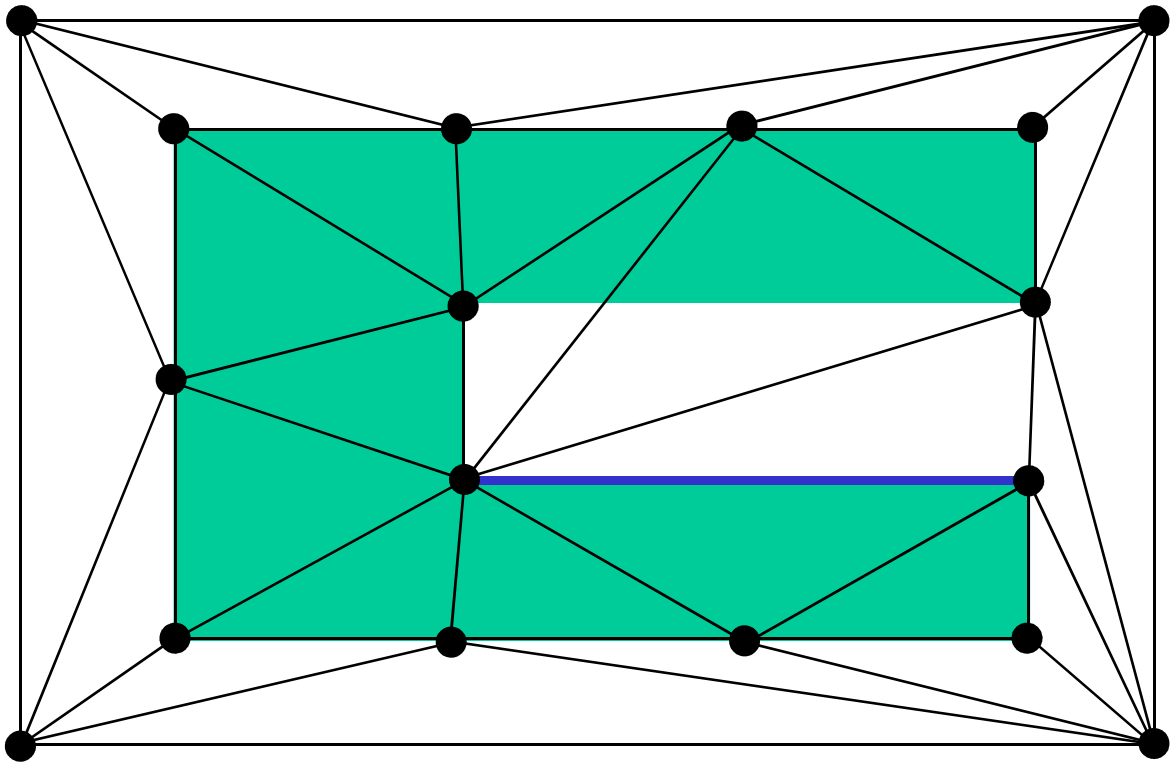


Local Swapping

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

(George,91)(Owen,99)

Delaunay: Boundary Constrained

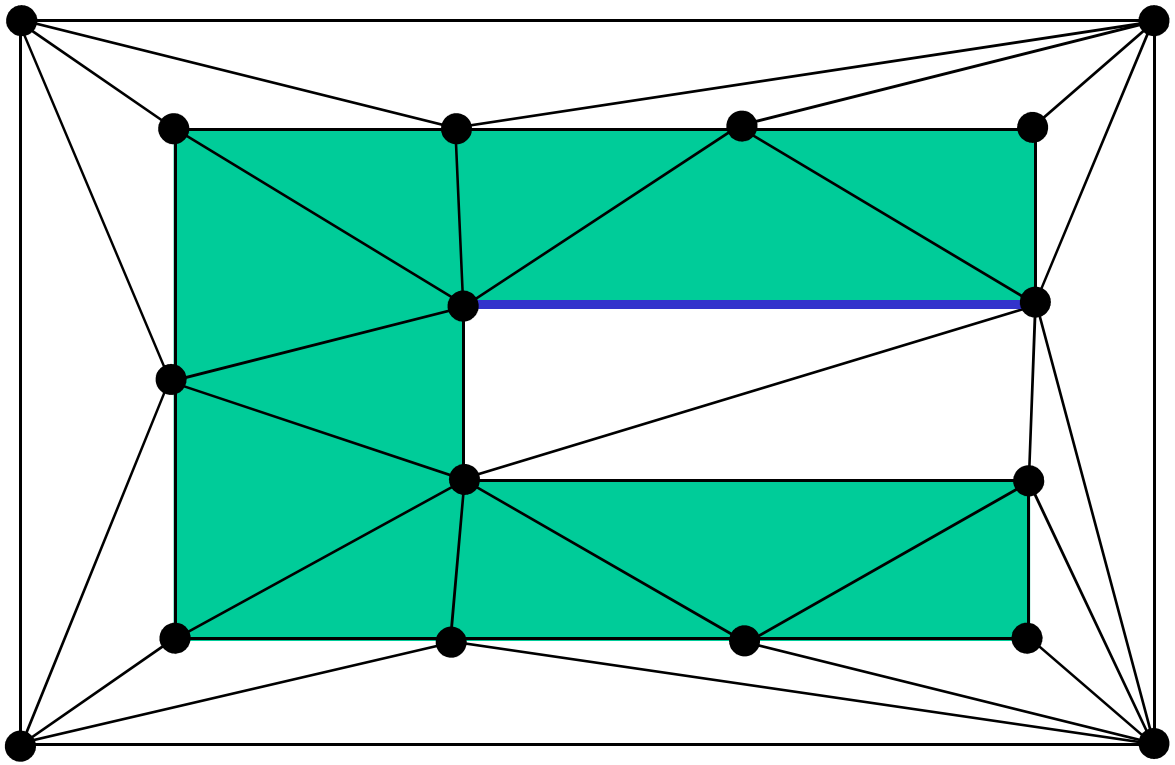


Local Swapping

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

(George,91)(Owen,99)

Delaunay: Boundary Constrained

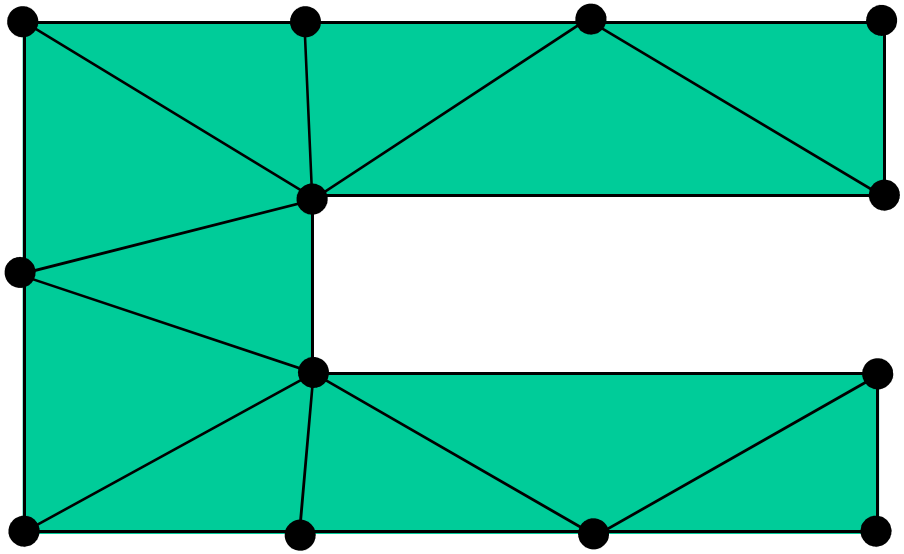


Local Swapping

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

(George,91)(Owen,99)

Delaunay: Boundary Constrained

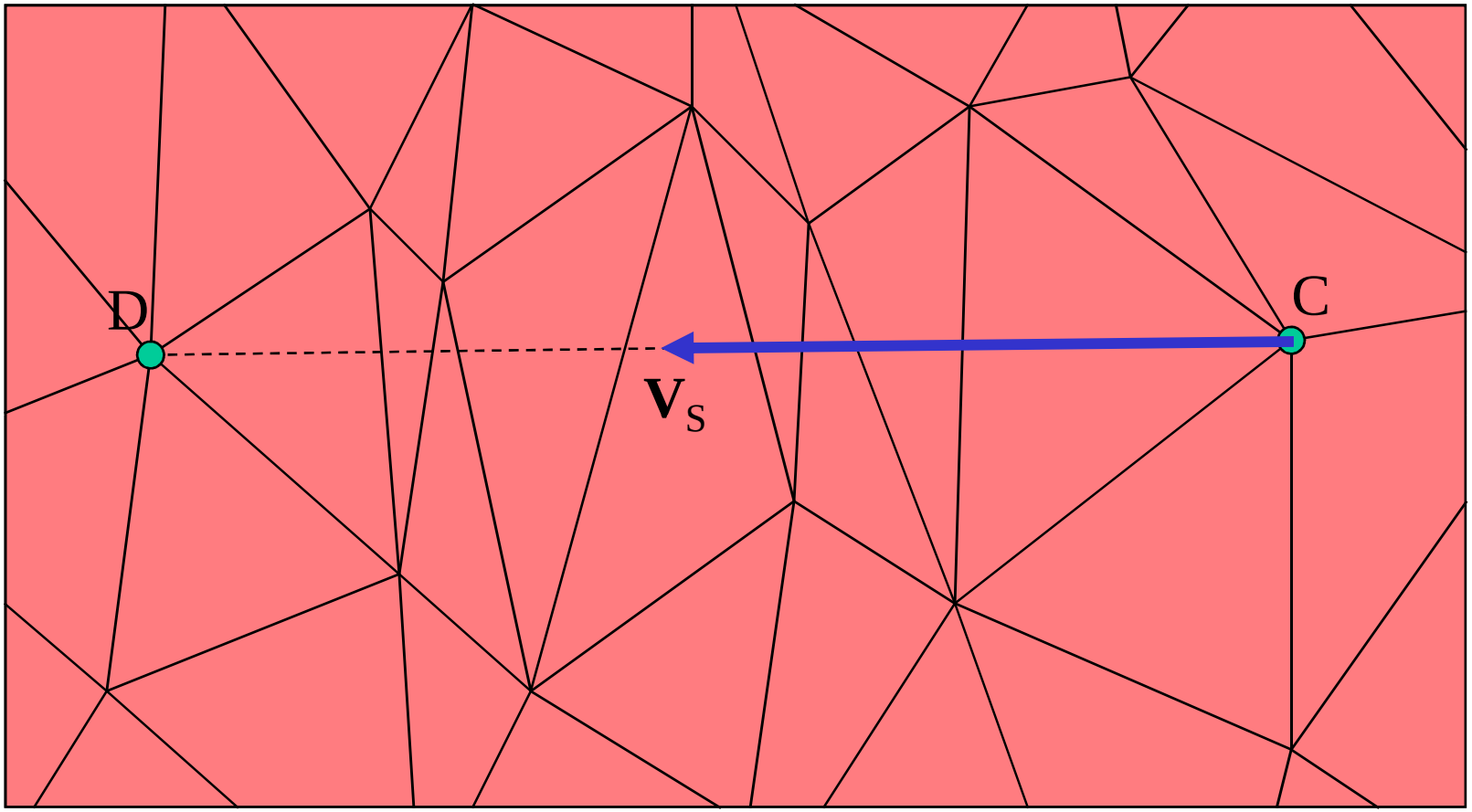


Local Swapping

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

(George,91)(Owen,99)

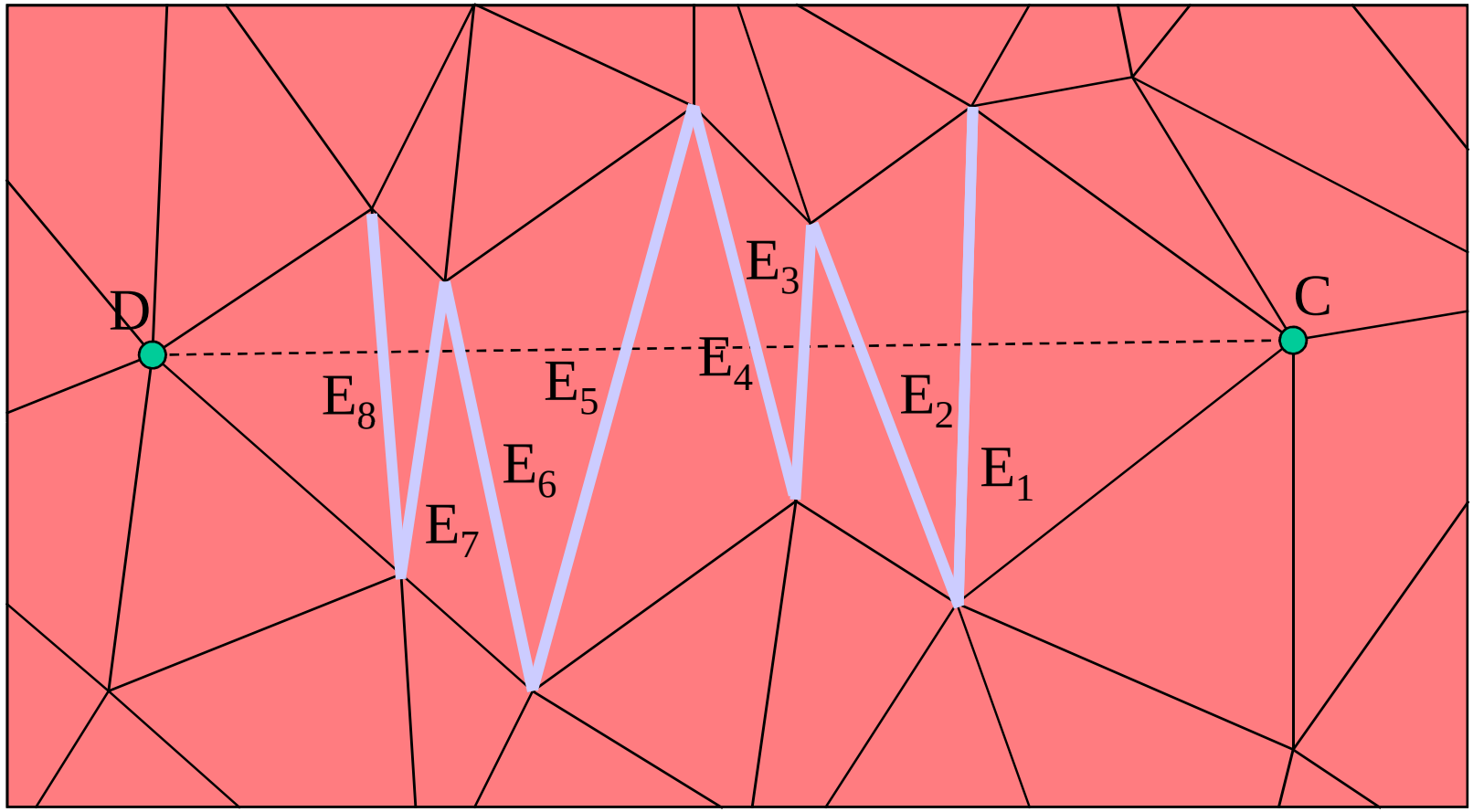
Delaunay: Boundary Constrained



Local Swapping Example

Recover edge CD at vector **V**s

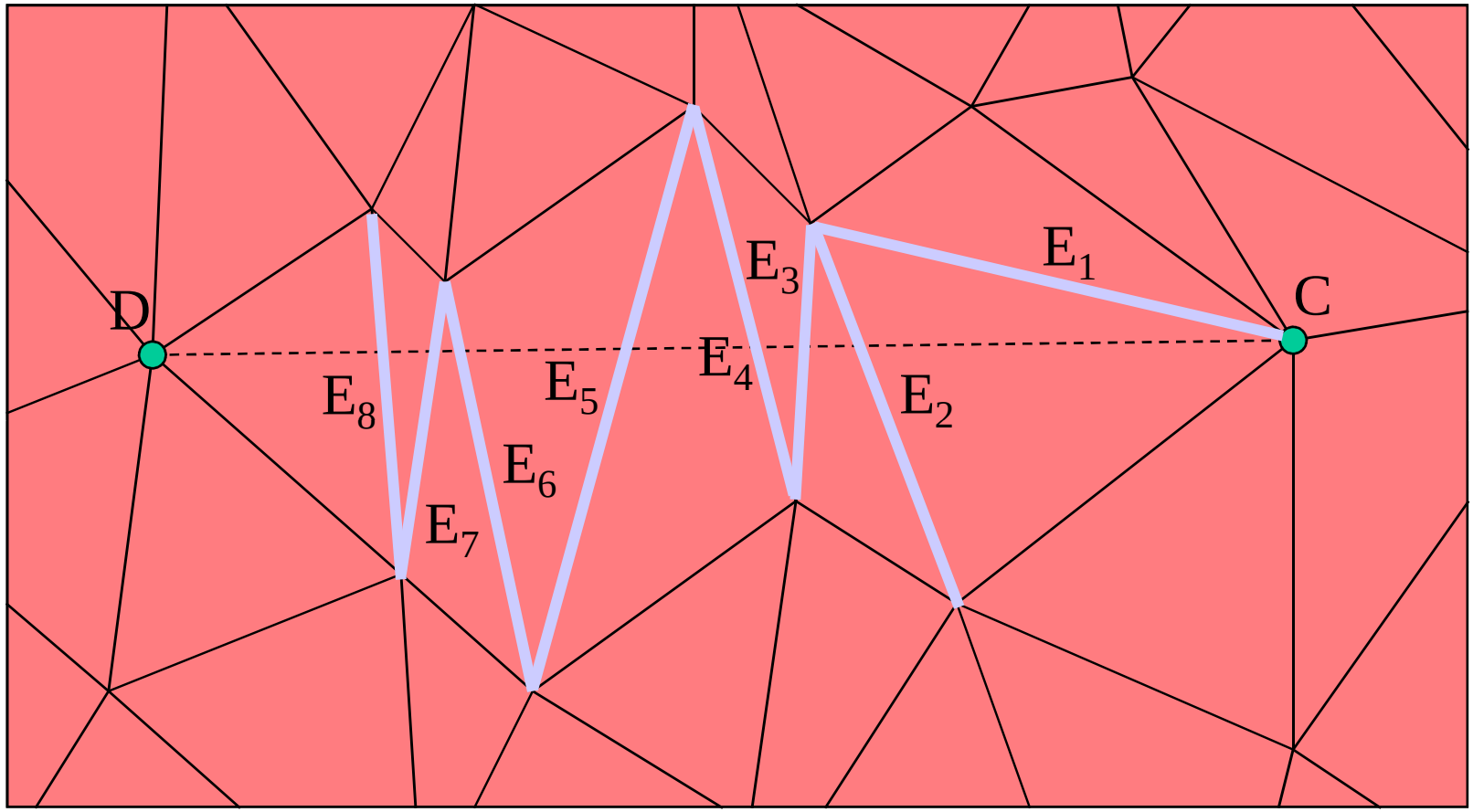
Delaunay: Boundary Constrained



Local Swapping Example

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

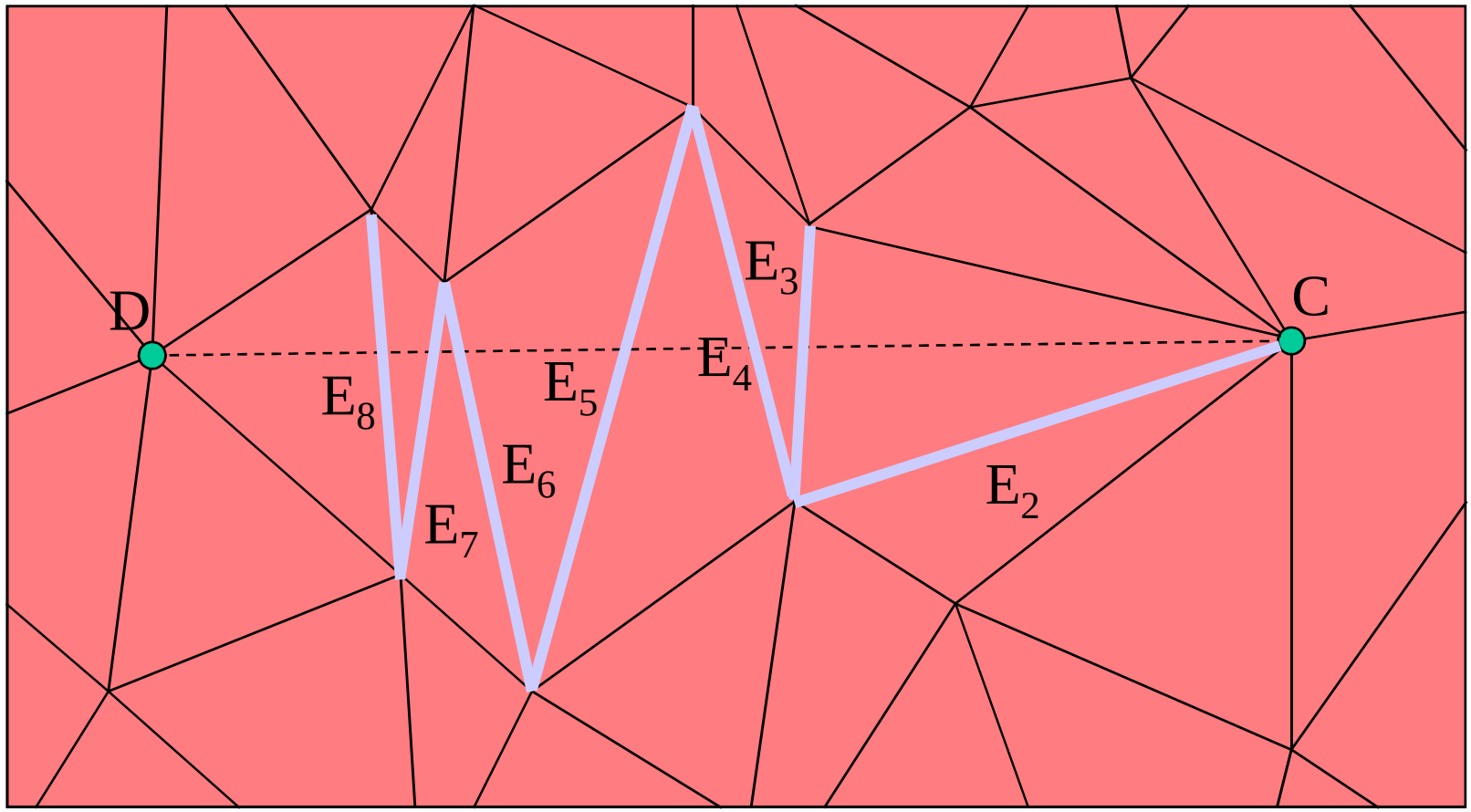
Delaunay: Boundary Constrained



Local Swapping Example

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

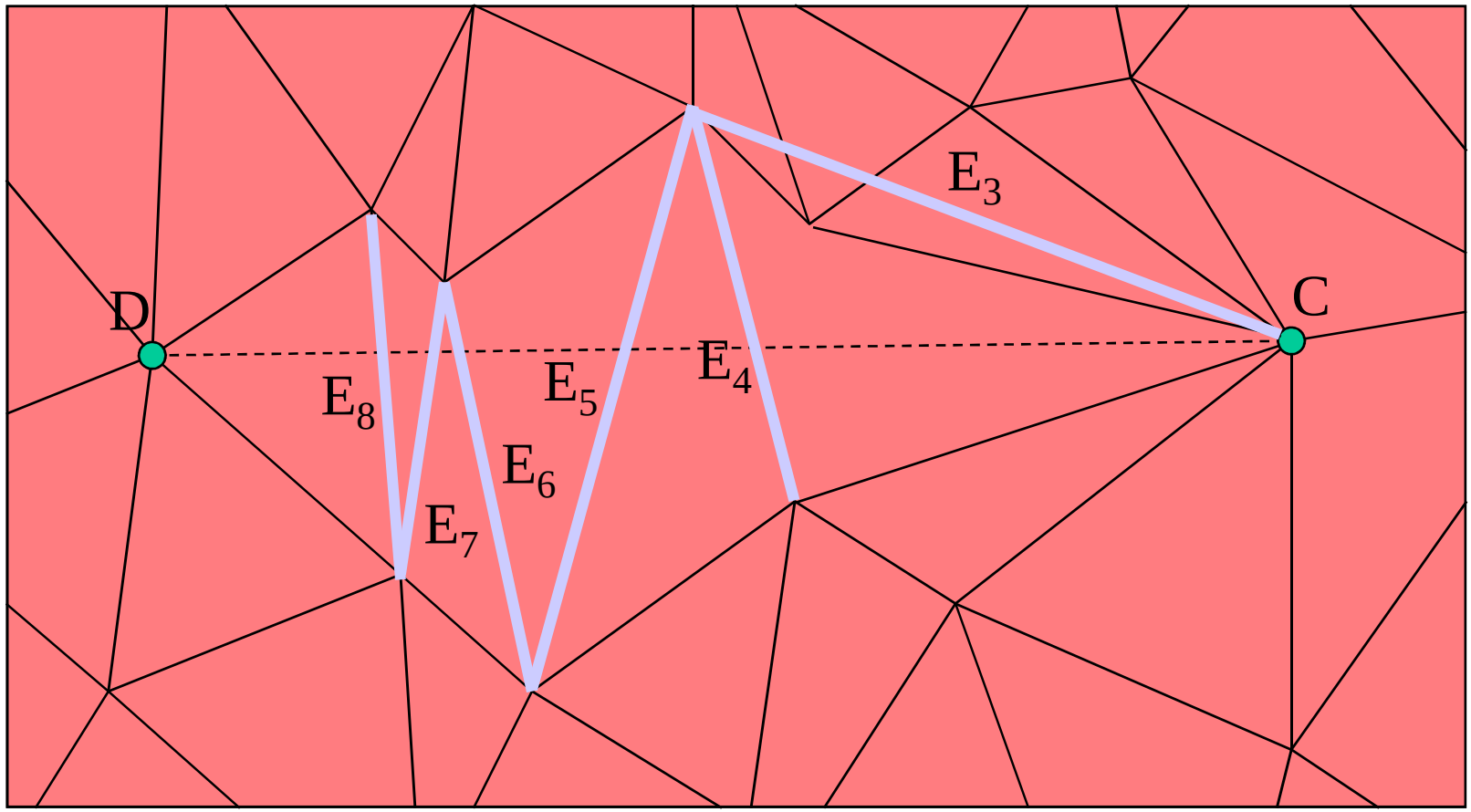
Delaunay: Boundary Constrained



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

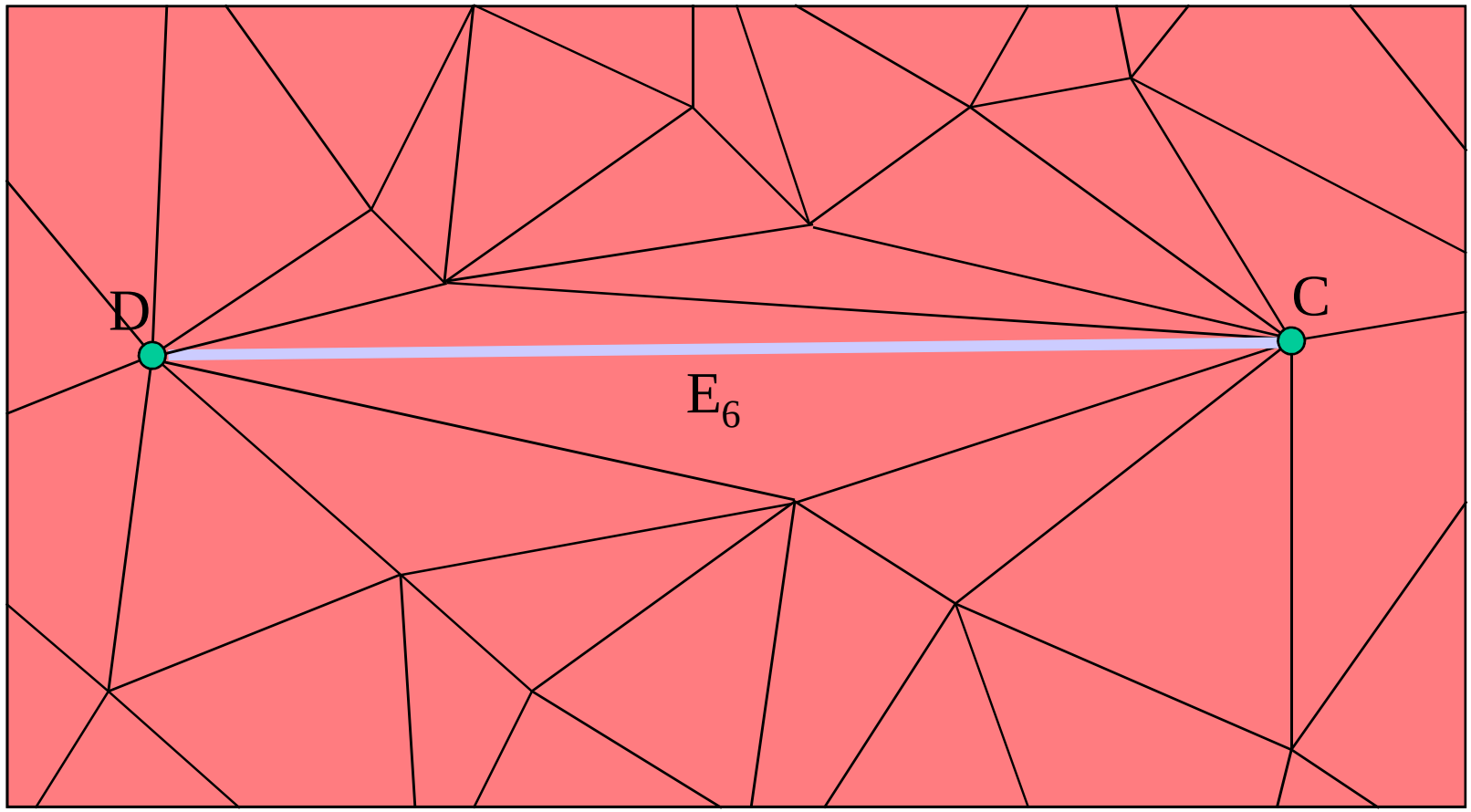
Delaunay: Boundary Constrained



Local Swapping Example

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

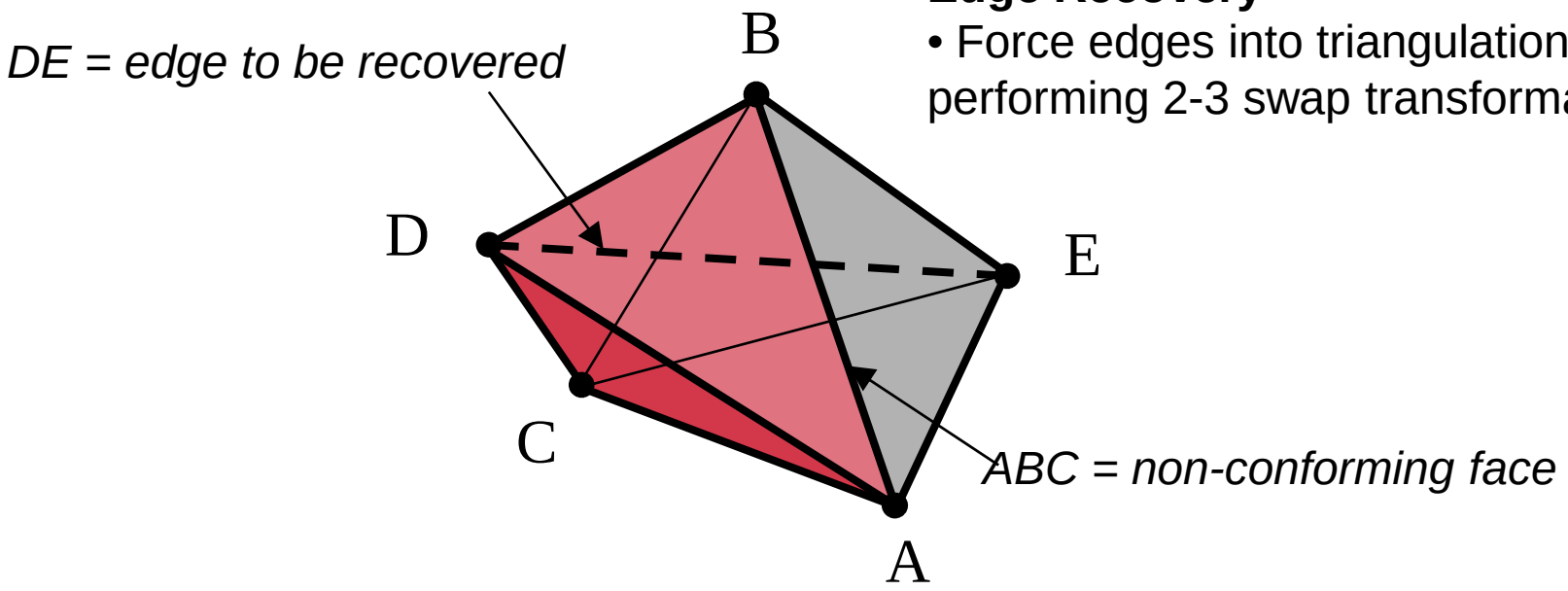
Delaunay: Boundary Constrained



Local Swapping Example

- Final swap will recover the desired edge.
- Resulting triangle quality may be poor if multiple swaps were necessary
- Does not maintain Delaunay criterion!

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

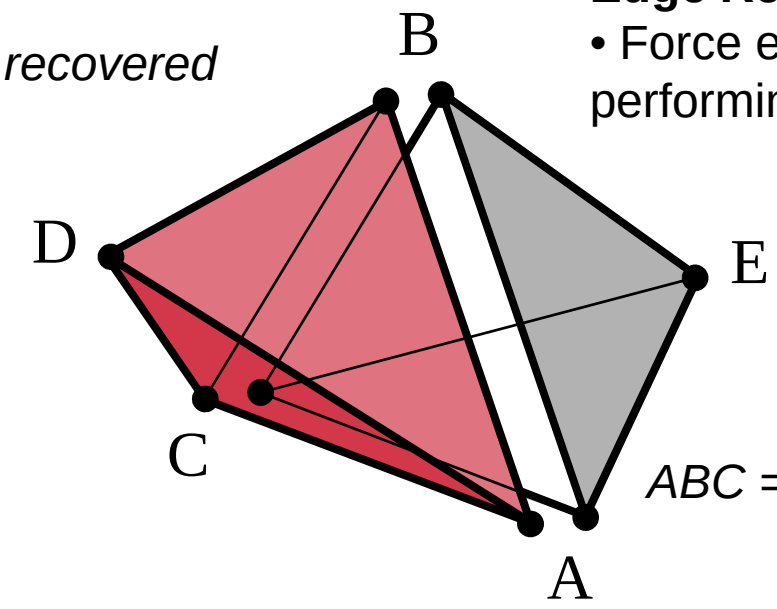
(George,91;99)(Owen,00)

Delaunay: 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

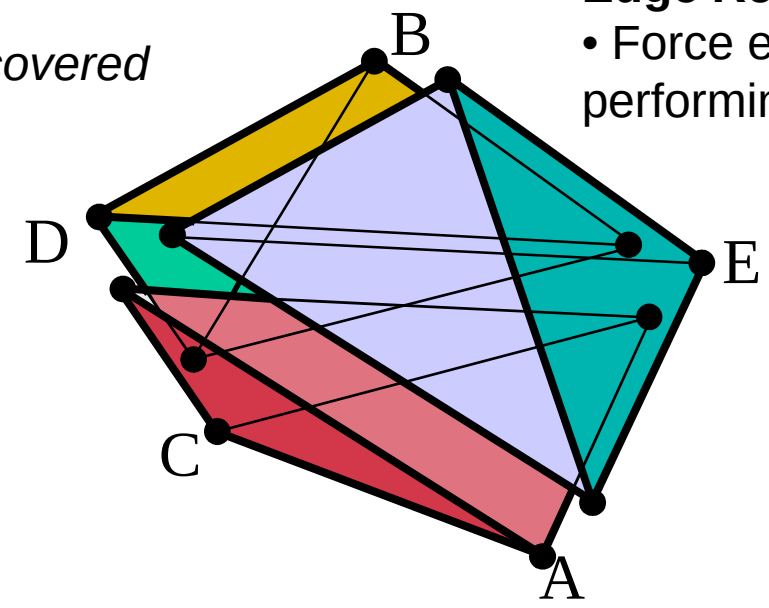
3D Local Swapping

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

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

Delaunay: Boundary Constrained

DE = edge recovered



Edge Recovery

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

2-3 Swap

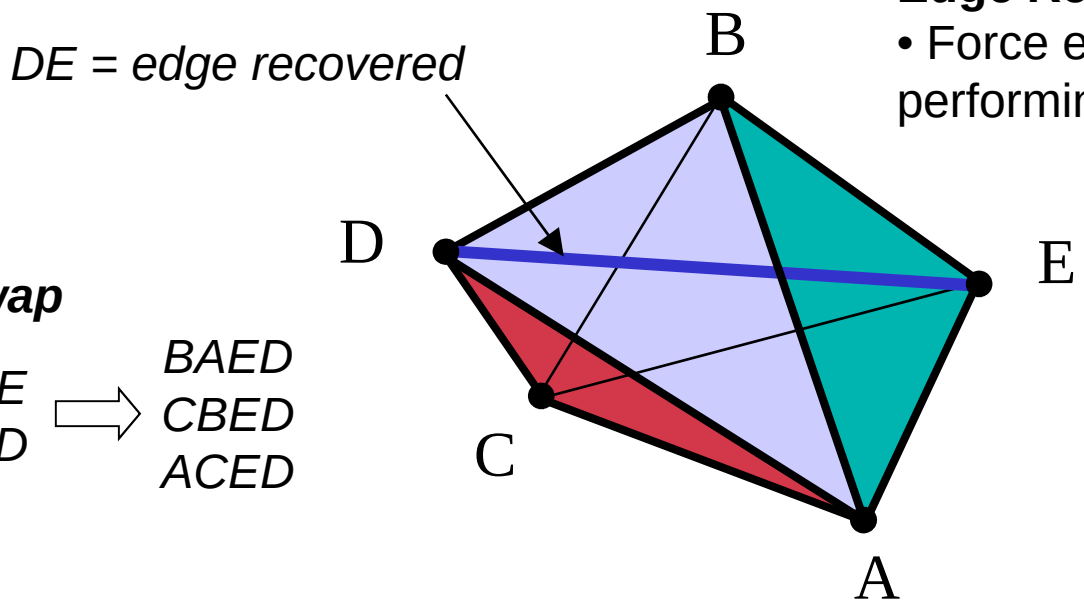
ABCE
ACBD
⇨
BAED
CBED
ACED

3D Local Swapping

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

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

Delaunay: Boundary Constrained



Edge Recovery

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

2-3 Swap

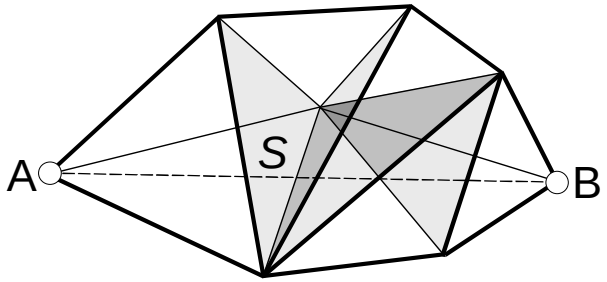
ABCE
ACBD
⇨
BAED
CBED
ACED

3D Local Swapping

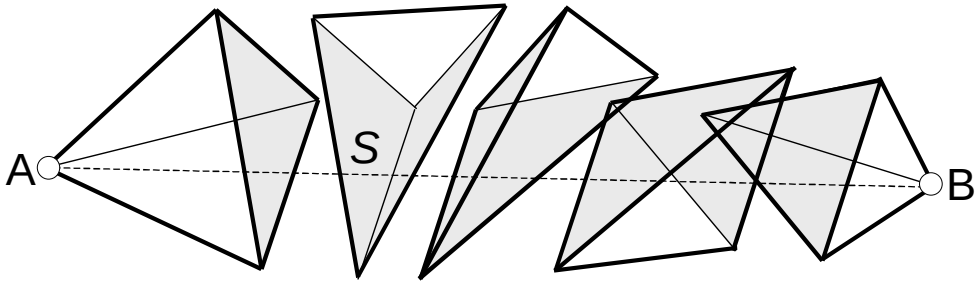
➢Requires both boundary *edge* recovery and boundary *face* recovery

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

Delaunay: Boundary Constrained



Edge AB to be recovered



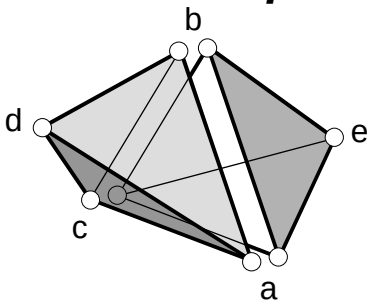
Exploded view of tets intersected by AB

3D Edge Recovery

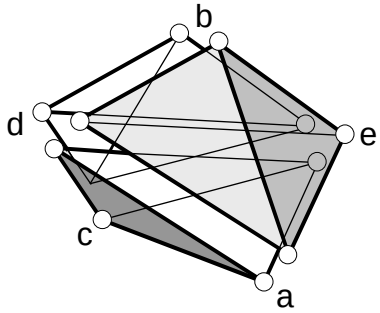
- Form queue of faces through which edge AB will pass
- Perform 2-3 swap transformations on all faces in the list
- If overlapping tets result, place back on queue and try again later
- If still cannot recover edge, then insert “Steiner” point

Delaunay

2-3 Swap

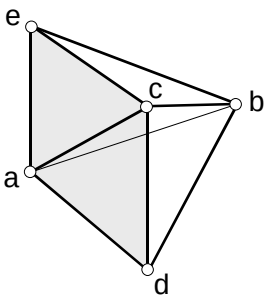


abce, acbd

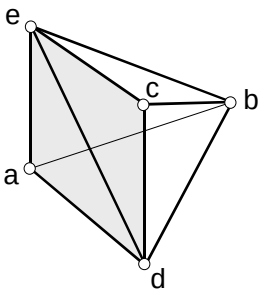


abde, bcde, cade

2-2 Swap

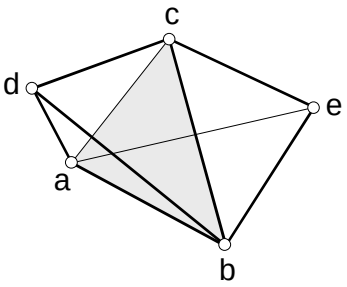


aceb, adcb

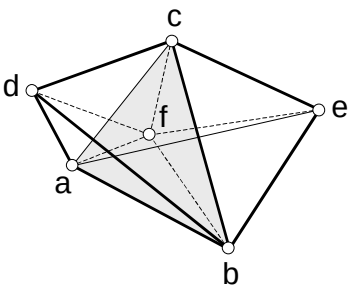


adeb, edcb

Face Split

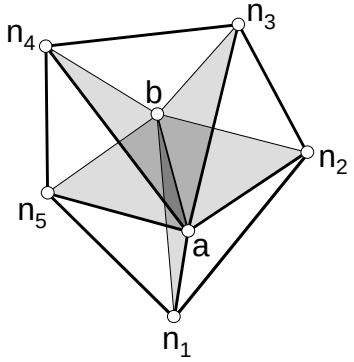


abce, acbd

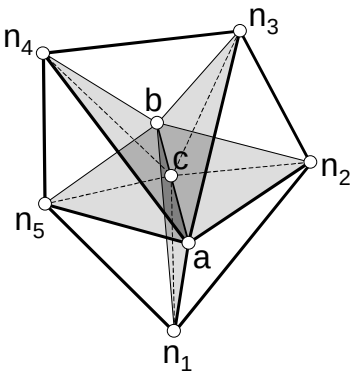


abfe, bcfe, cafe
bafd, cbfd, acfd

Edge Split



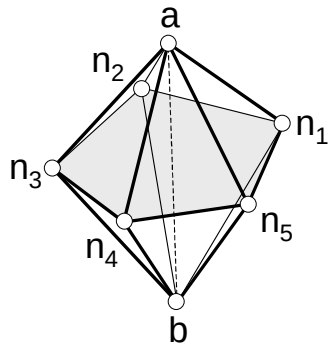
abn_in_{i+1} {i=1…N}



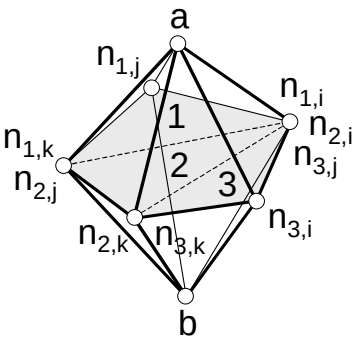
abn_in_{i+1},
cbn_in_{i+1} {i=1…N}

N = no. adj. tets at edge ab

Edge Suppress



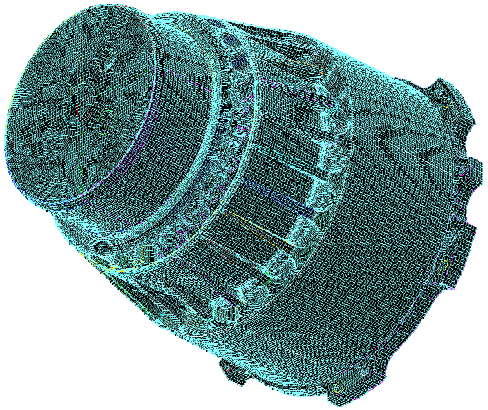
abn_in_{i+1} {i=1…N}

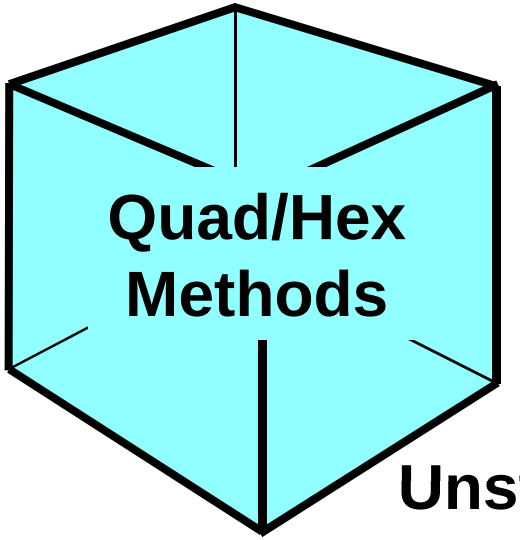


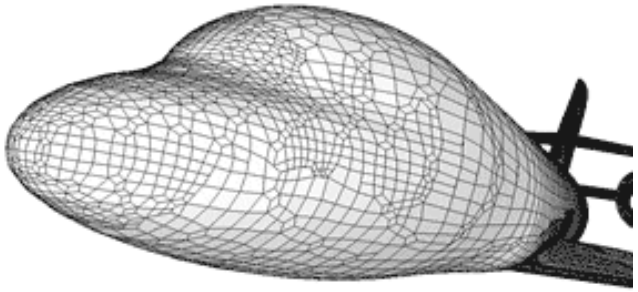
$n_{m,k}n_{m,j}n_{m,i}a$,
$n_{m,k}n_{m,j}n_{m,i}b$ {m=1…M}

M = no. unique trias in polygon P={n_1,n_2,n_3}

Mesh: Structured / Unstructured

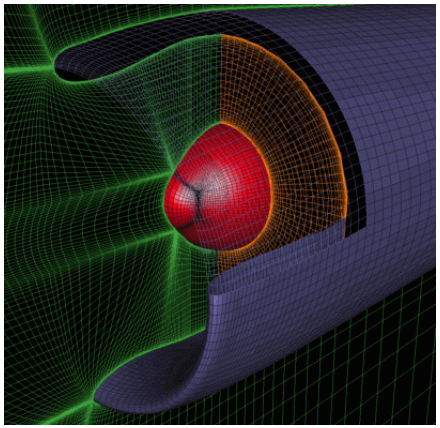






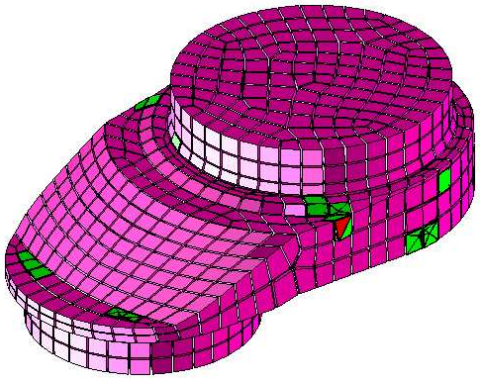
Structured

- Requires geometry to conform to specific characteristics
- Regular patterns of quads/hexes formed based on characteristics of geometry
- Internal nodes always attached to same number of elements

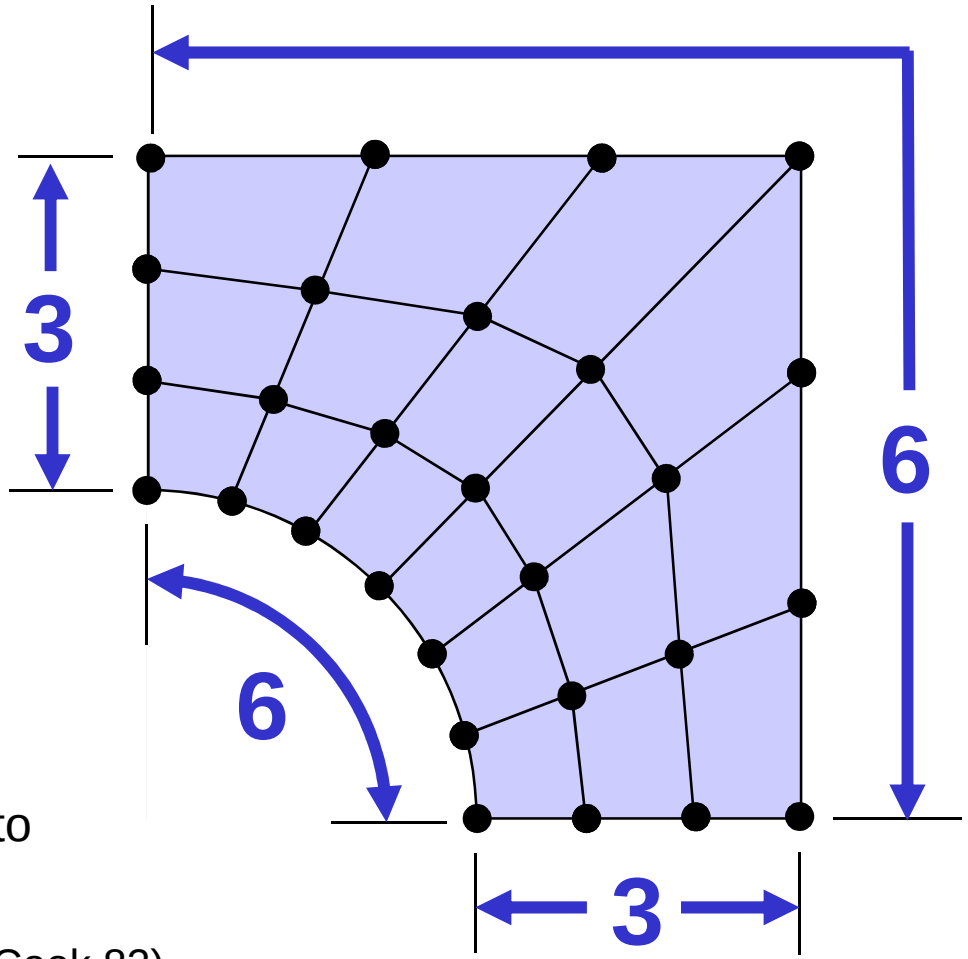


Unstructured

- No specific requirements for geometry
- quads/hexes placed to conform to geometry.
- No connectivity requirement (although optimization of connectivity is beneficial)



Structured: Mapped Meshing



Algorithm

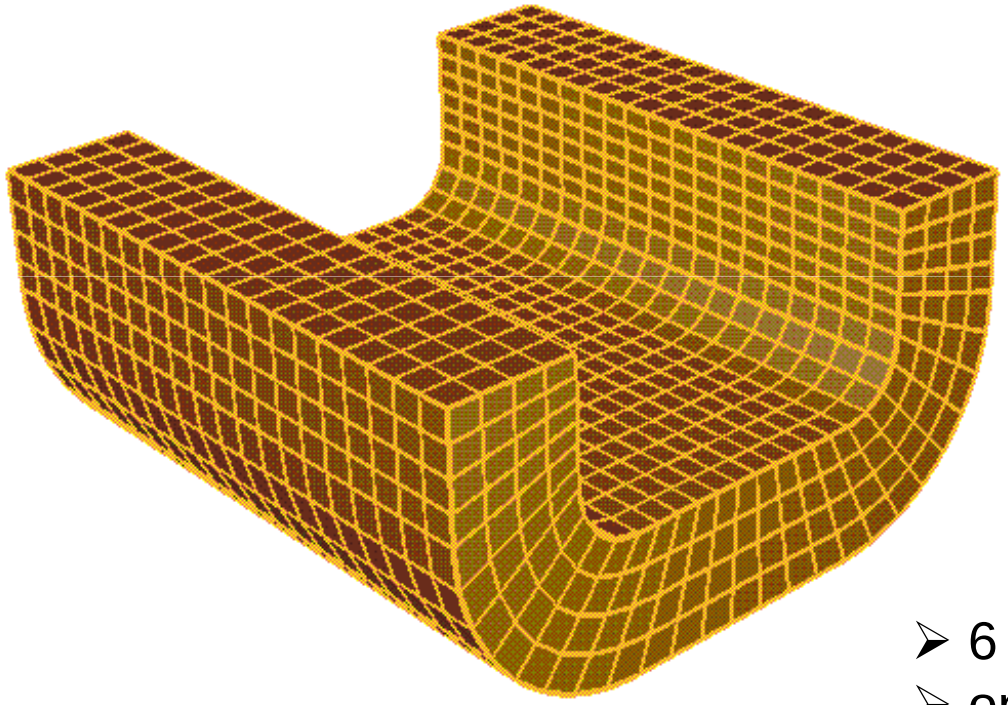
- Trans-finite Interpolation (TFI)
- Maps a regular lattice of quads onto polygon

(Thompson,88;99), (Cook,82)

Geometry Requirements

- 4 topological sides
- Opposite sides must have similar intervals

Structured: **3D Mapped Meshing**

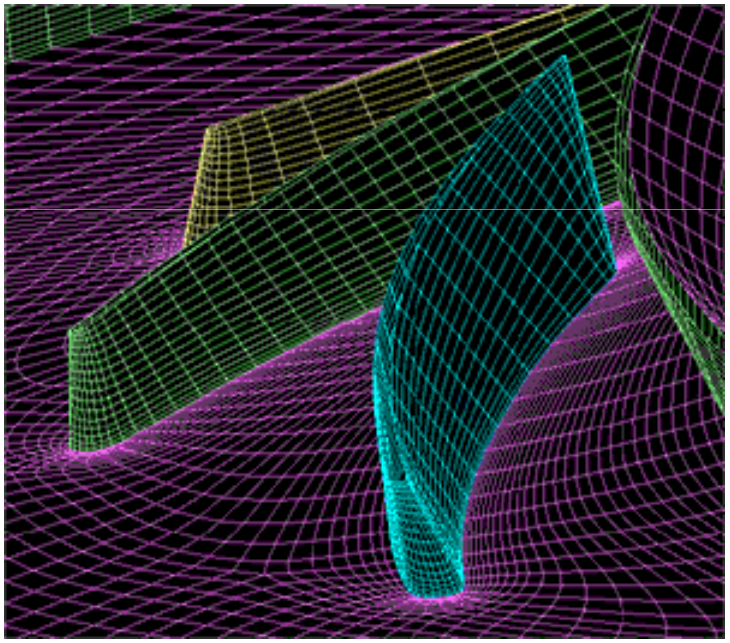


Geometry Requirements

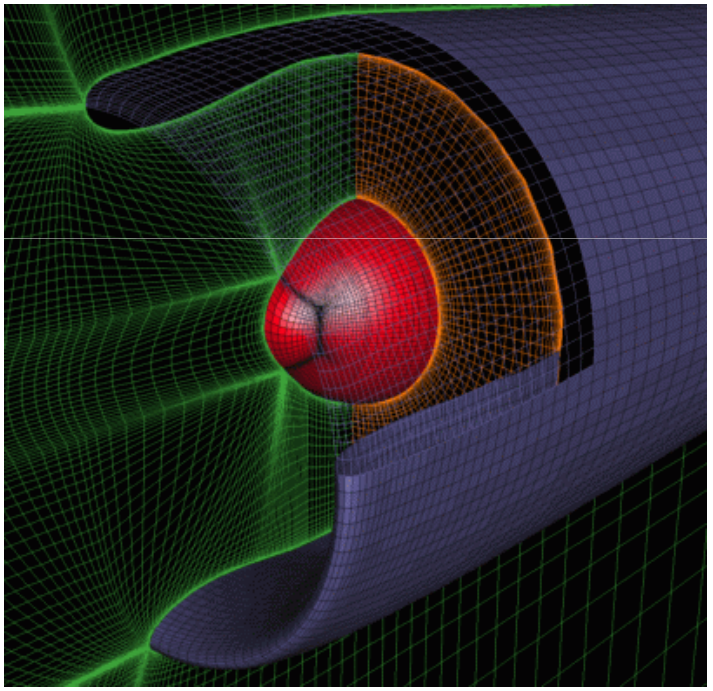
- 6 topological surfaces
- opposite surfaces must have similar mapped meshes

Structured: Mapped Meshing

Block-Structured

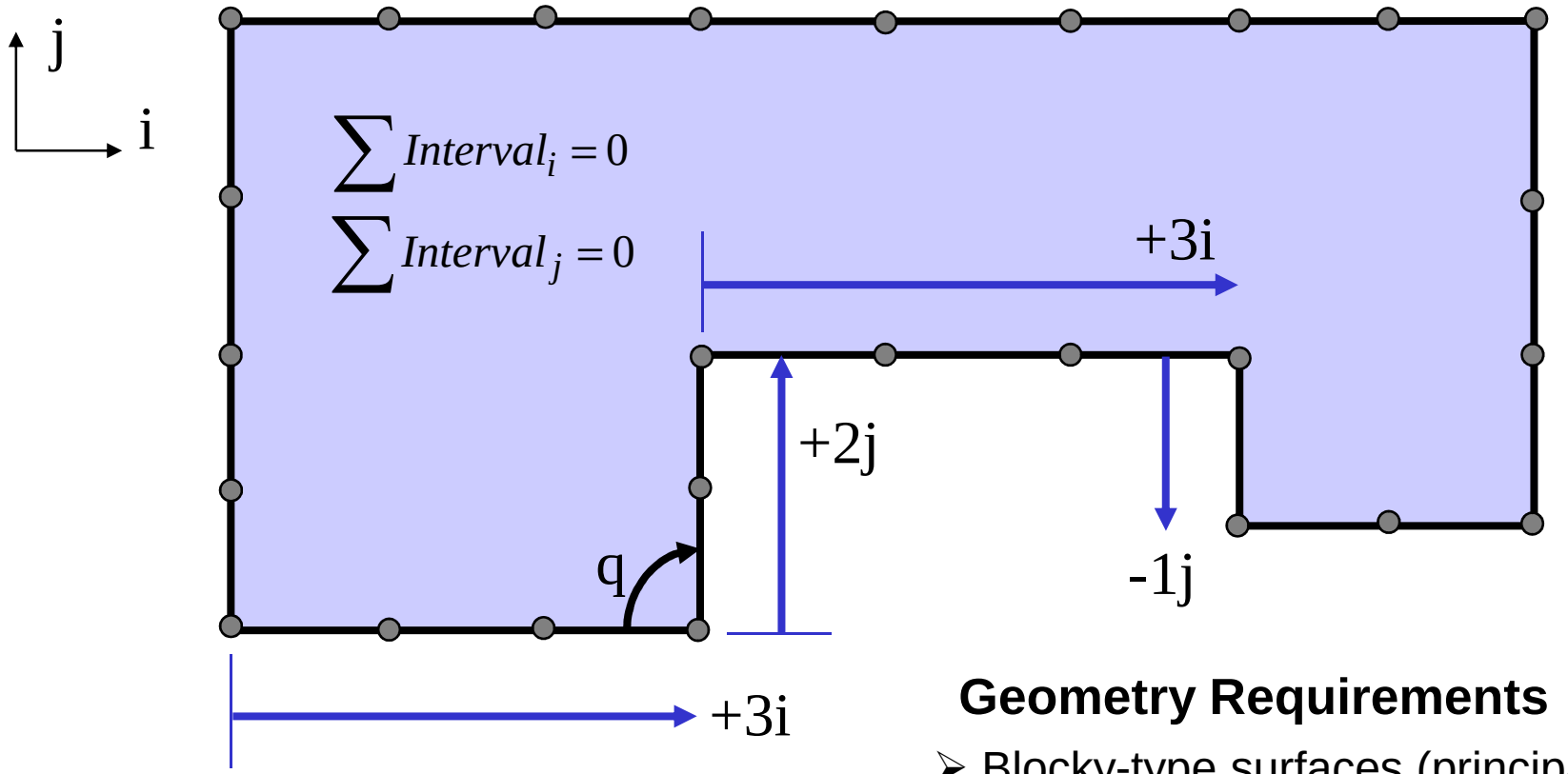


http://www.gridpro.com/gridgallery/tmachinery.html



http://www.pointwise.com/case/747.htm

Structured: Sub-Mapping

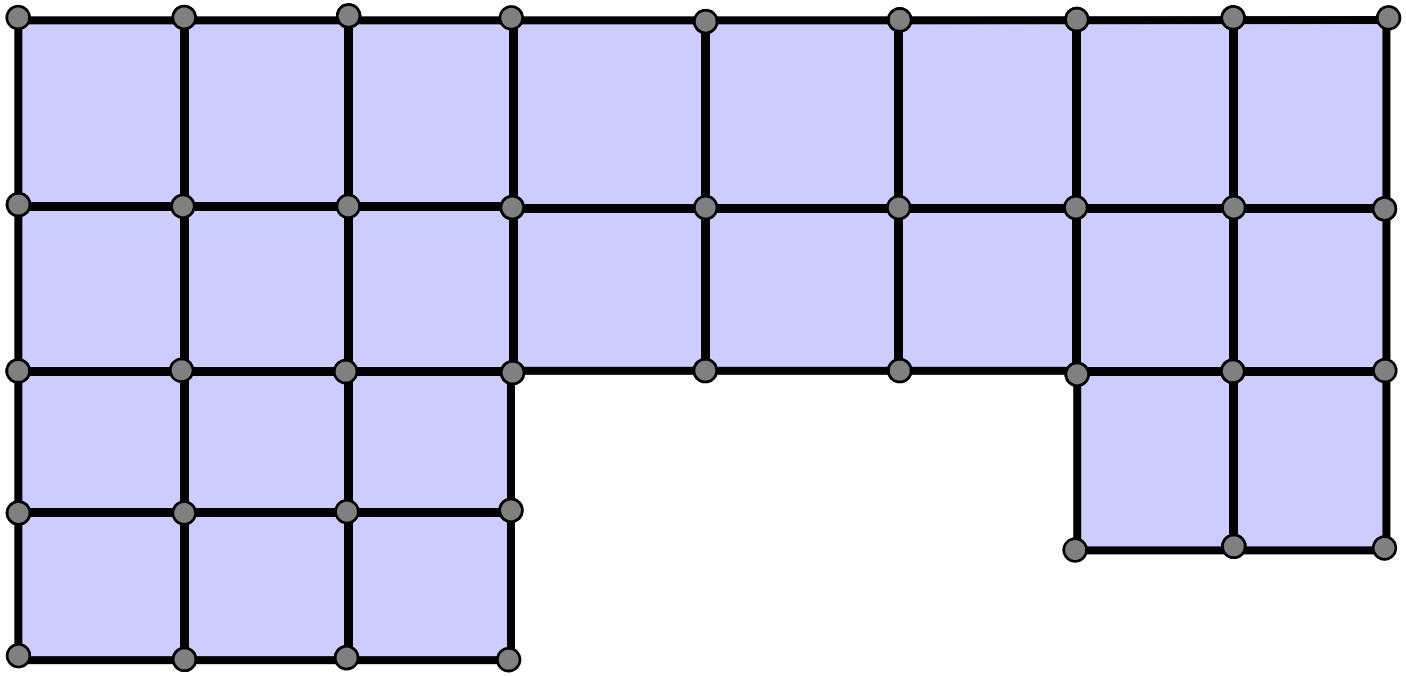


Geometry Requirements

- Blocky-type surfaces (principally 90 degree angles)

(White,95)

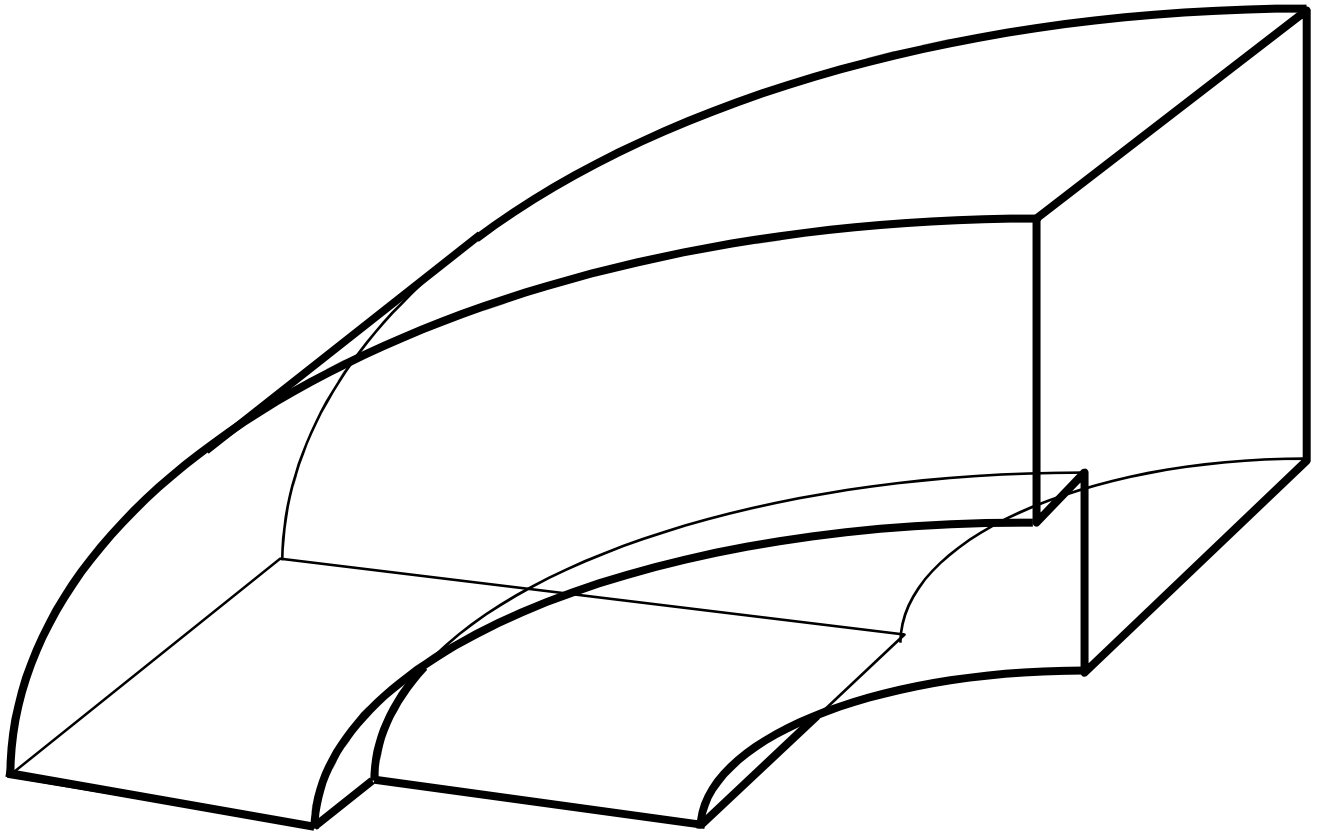
Structured: Sub-Mapping



- Automatically decomposes surface into mapable regions based on assigned intervals

(White,95)

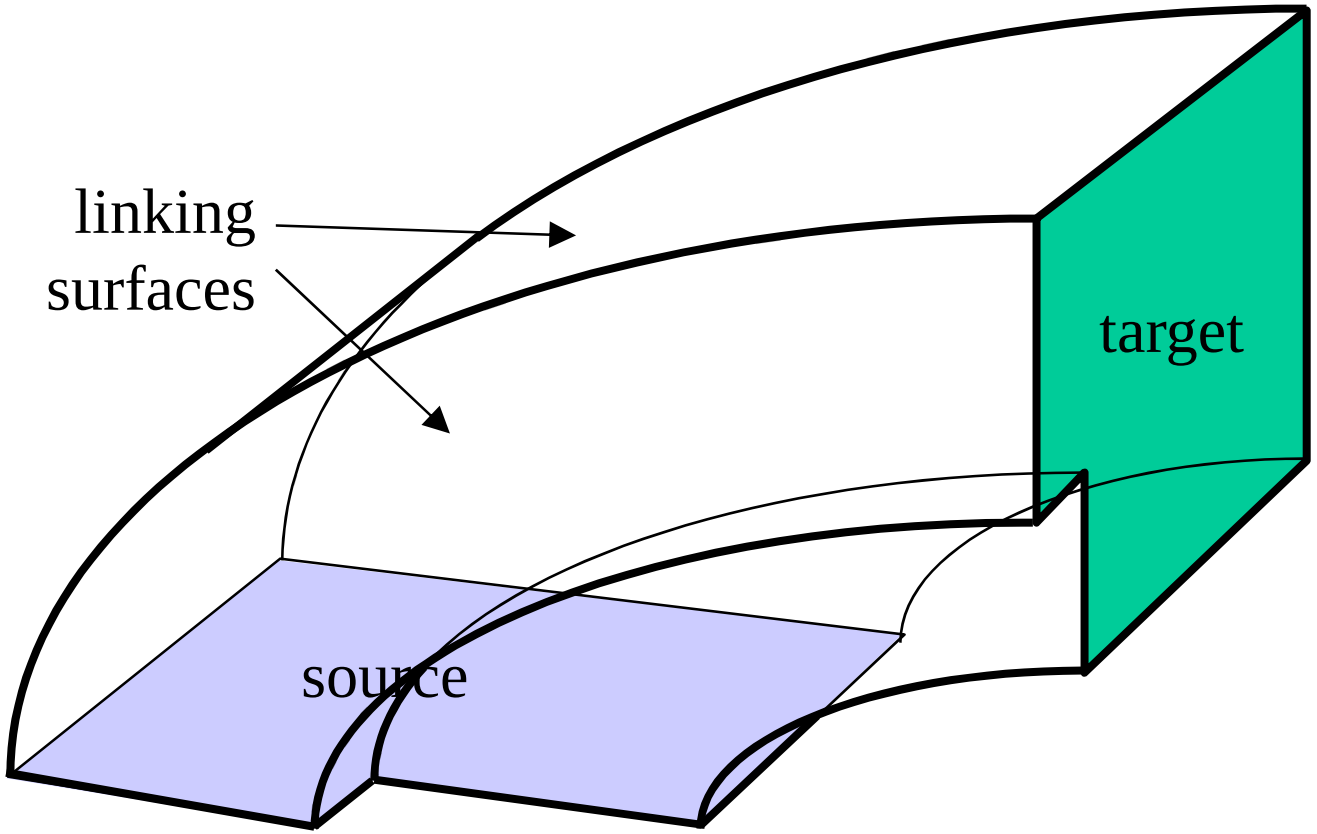
Structured: Sweeping



Geometry Requirements

- Source and target surfaces topologically similar
- Linking surfaces *mapable* or *submapable*

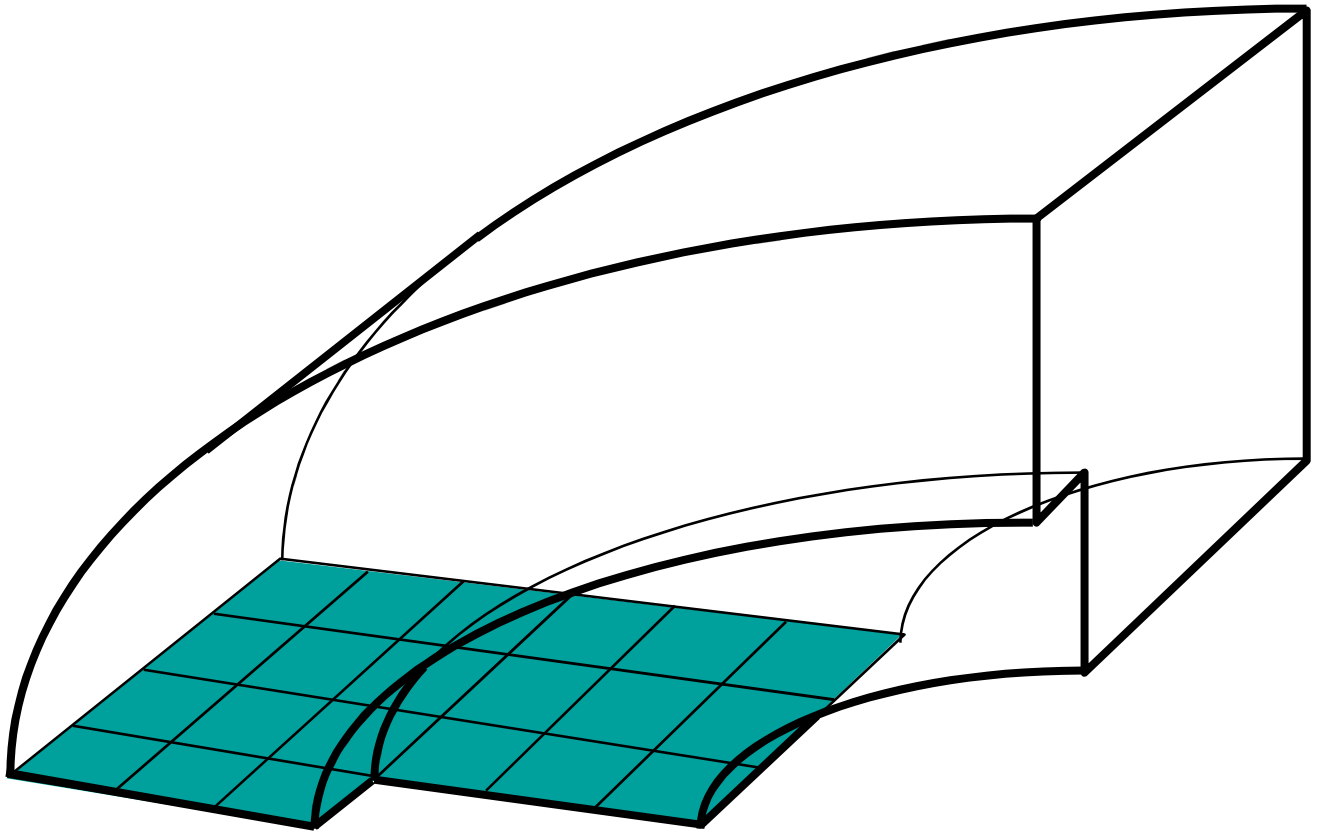
Structured: Sweeping



Geometry Requirements

- Source and target surfaces topologically similar
- Linking surfaces *mapable* or *submapable*

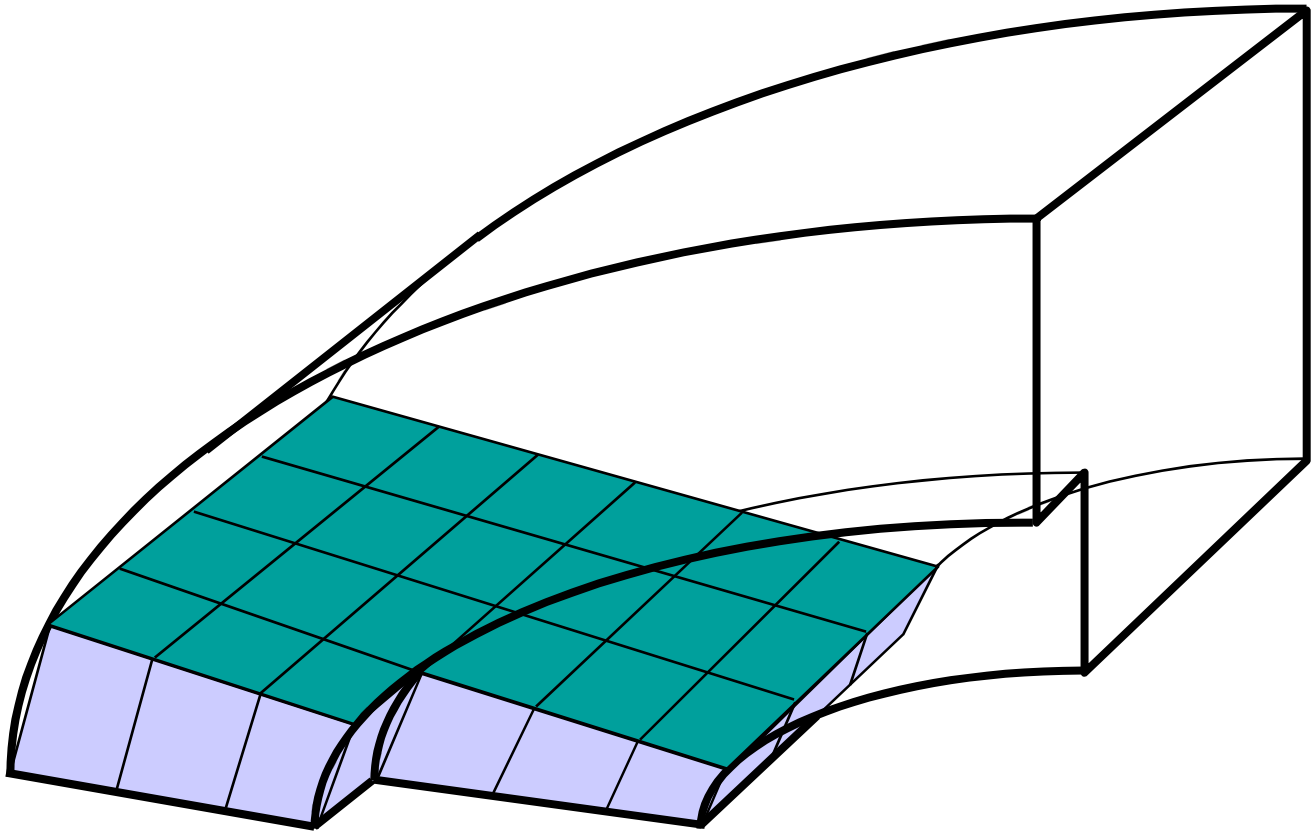
Structured: Sweeping



Geometry Requirements

- Source and target surfaces topologically similar
- Linking surfaces *mapable* or *submapable*

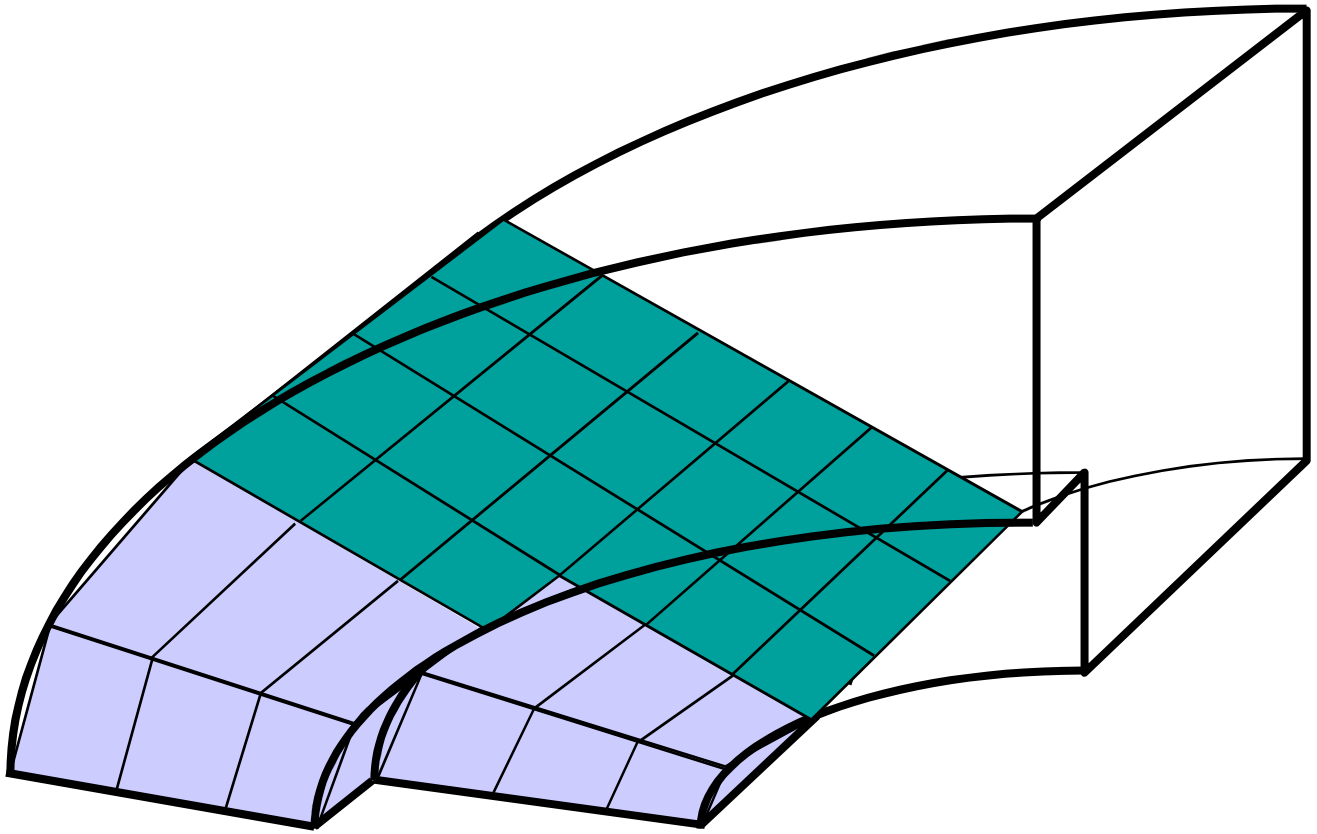
Structured: Sweeping



Geometry Requirements

- Source and target surfaces topologically similar
- Linking surfaces *mapable* or *submapable*

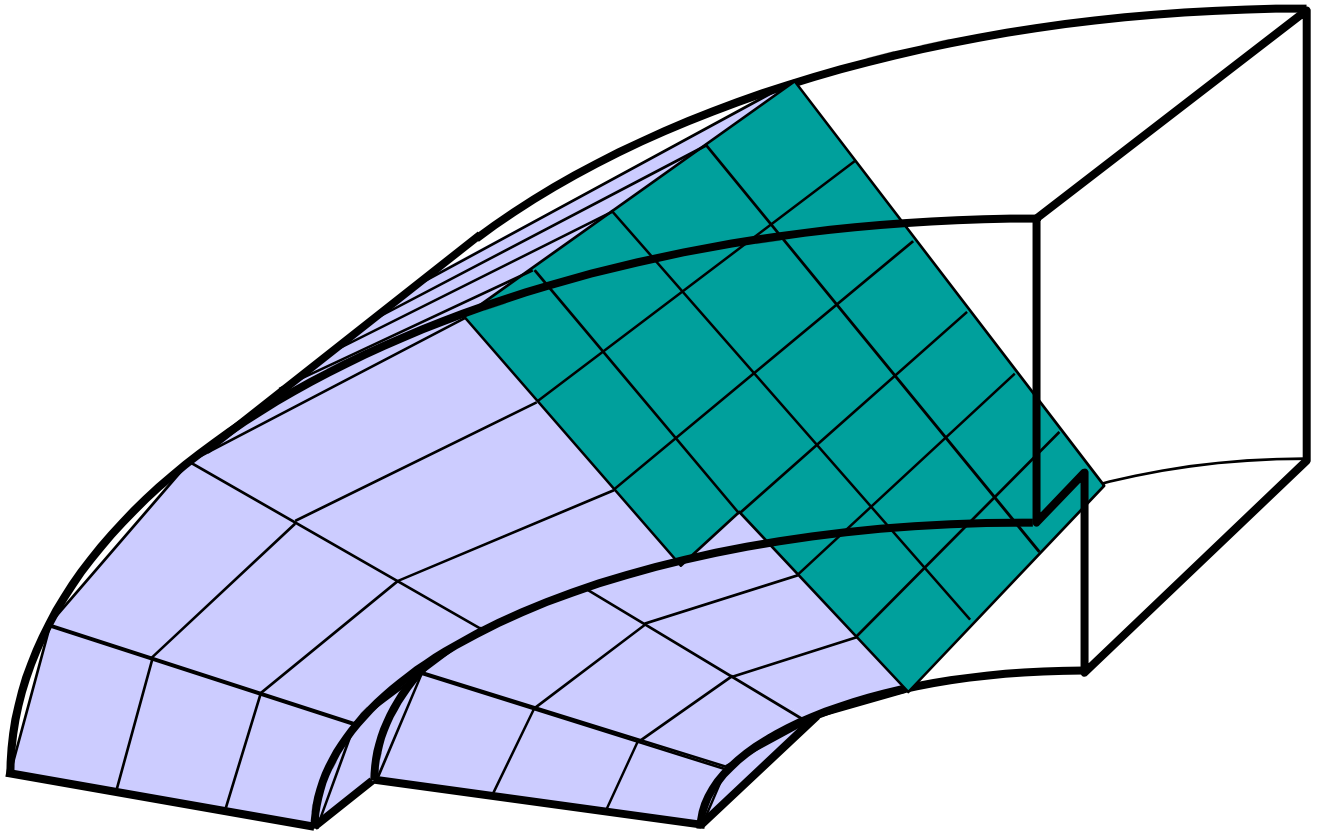
Structured: Sweeping



Geometry Requirements

- Source and target surfaces topologically similar
- Linking surfaces *mapable* or *submapable*

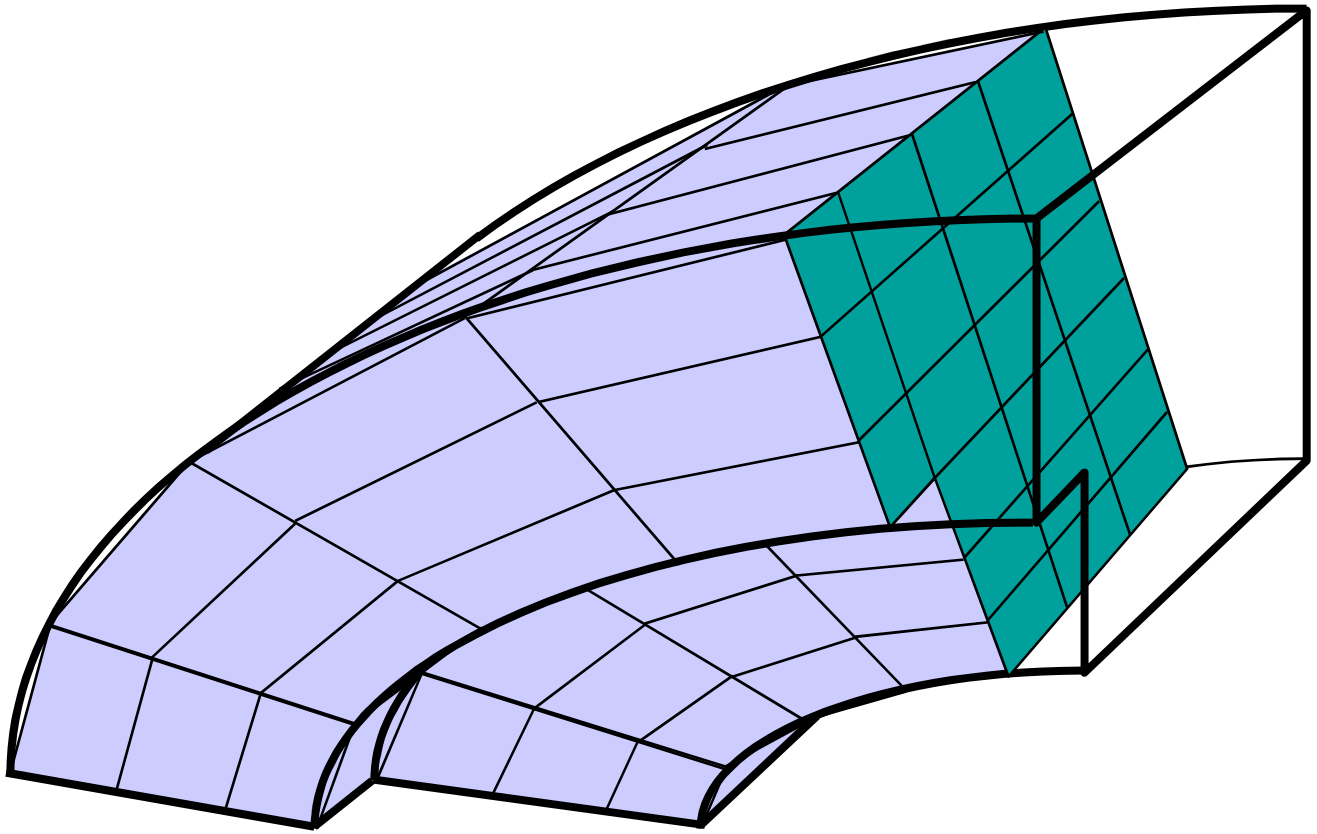
Structured: Sweeping



Geometry Requirements

- Source and target surfaces topologically similar
- Linking surfaces *mapable* or *submapable*

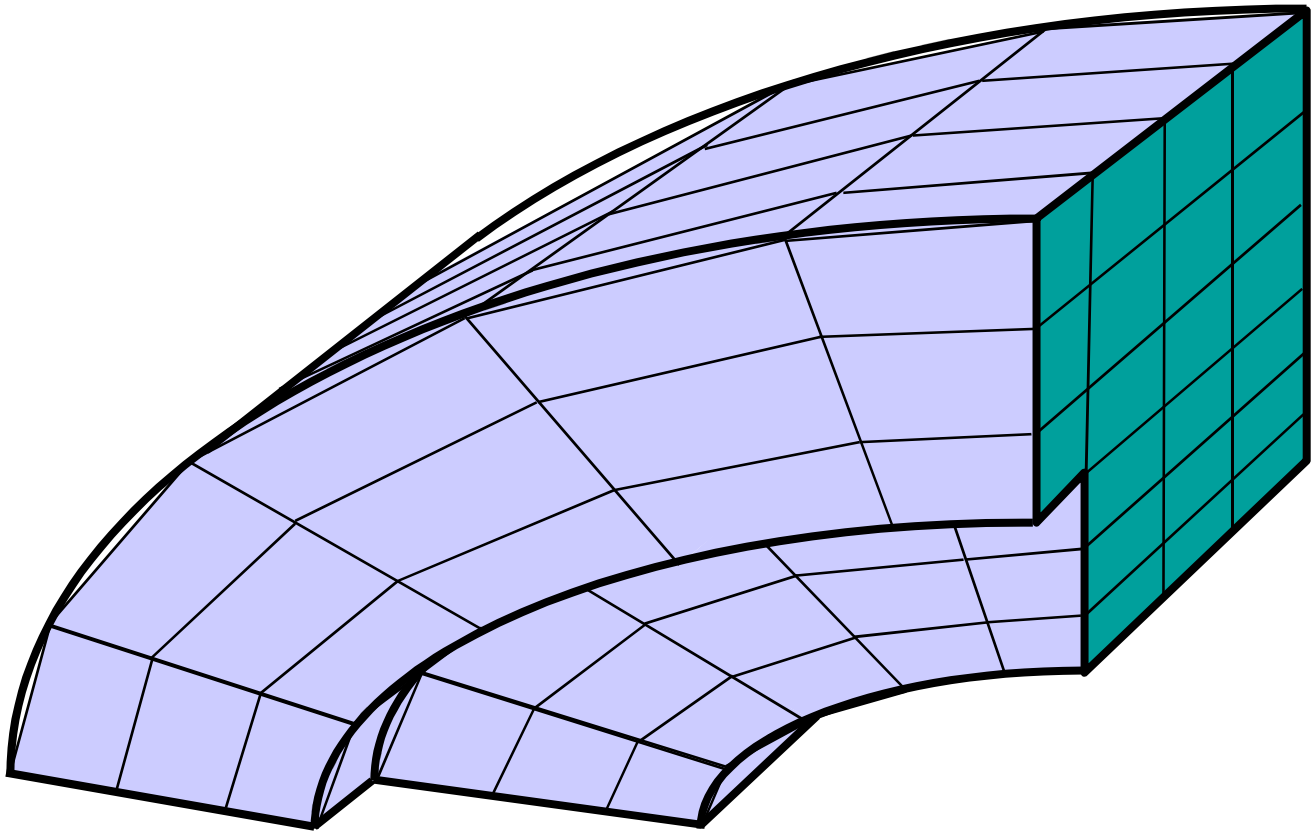
Structured: Sweeping



Geometry Requirements

- Source and target surfaces topologically similar
- Linking surfaces *mapable* or *submapable*

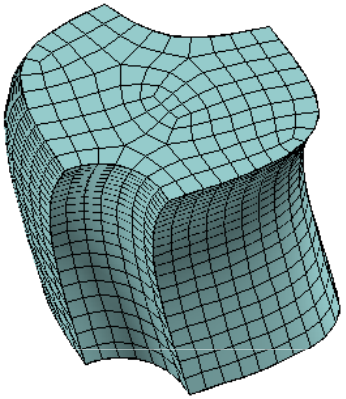
Structured: Sweeping

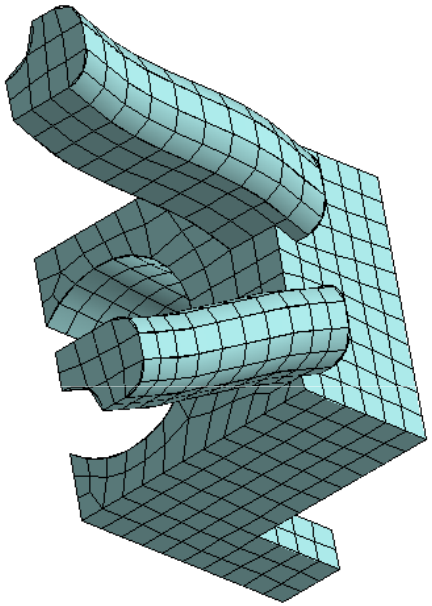


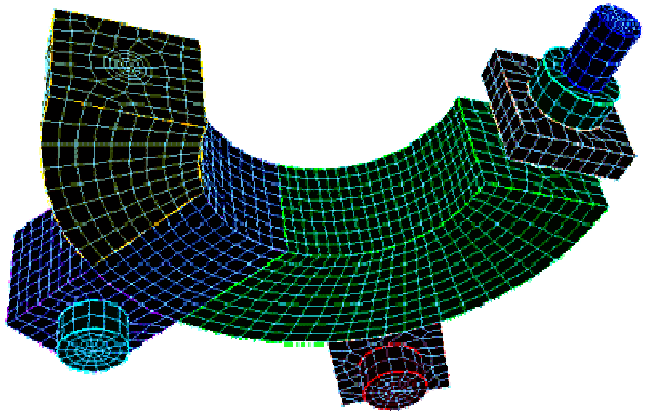
Geometry Requirements

- Source and target surfaces topologically similar
- Linking surfaces *mapable* or *submapable*

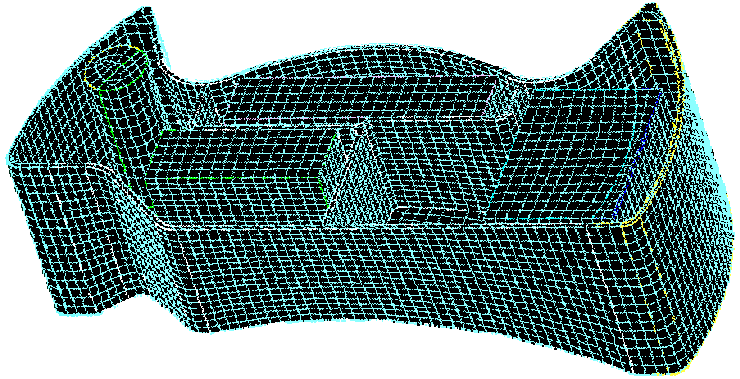
Structured: Sweeping

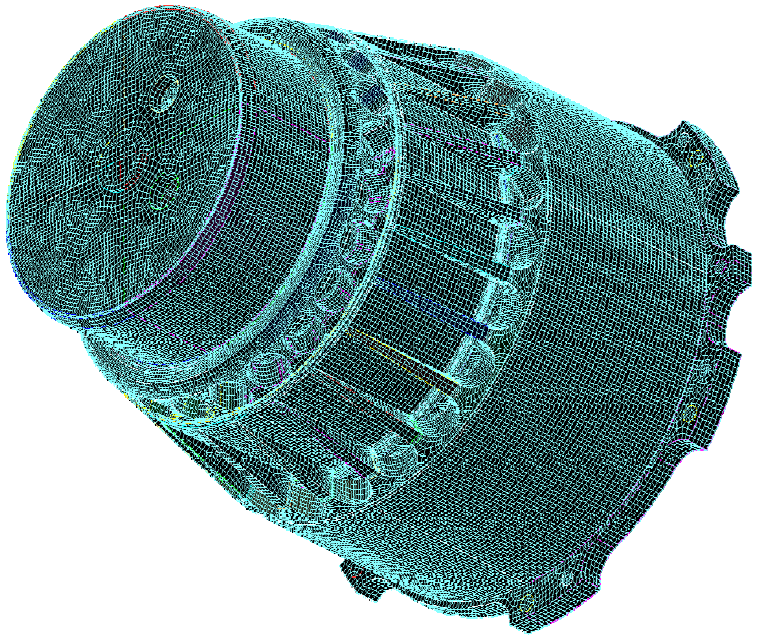






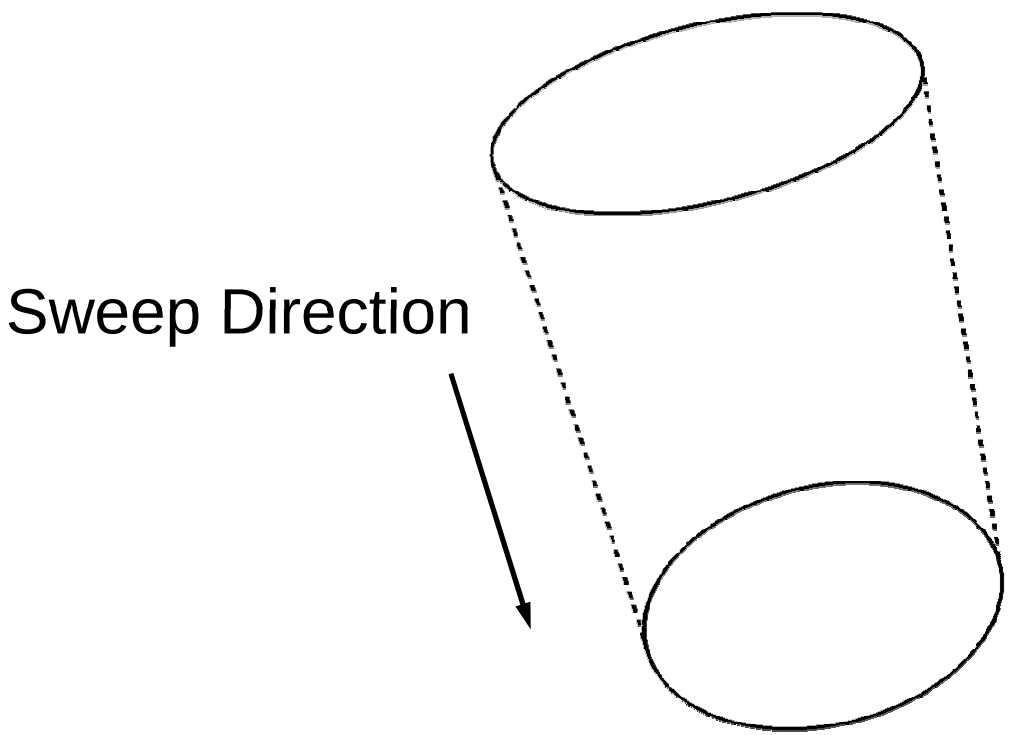
Gambit, Fluent Inc.





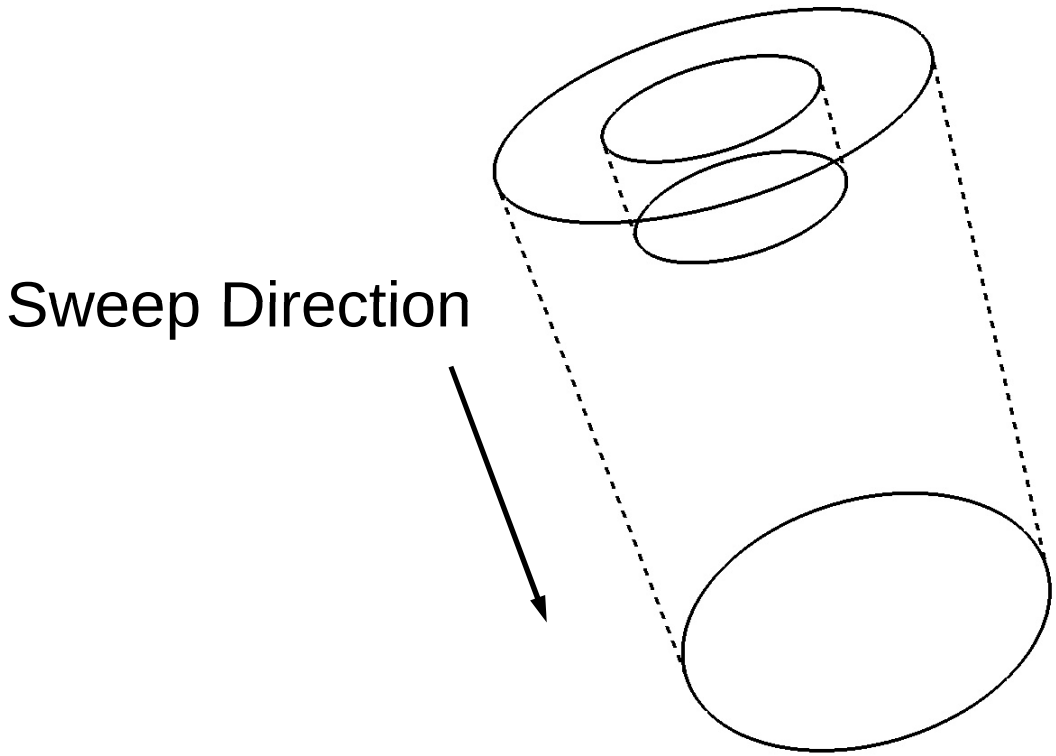
Cubit, Sandia National Labs

Sweeping



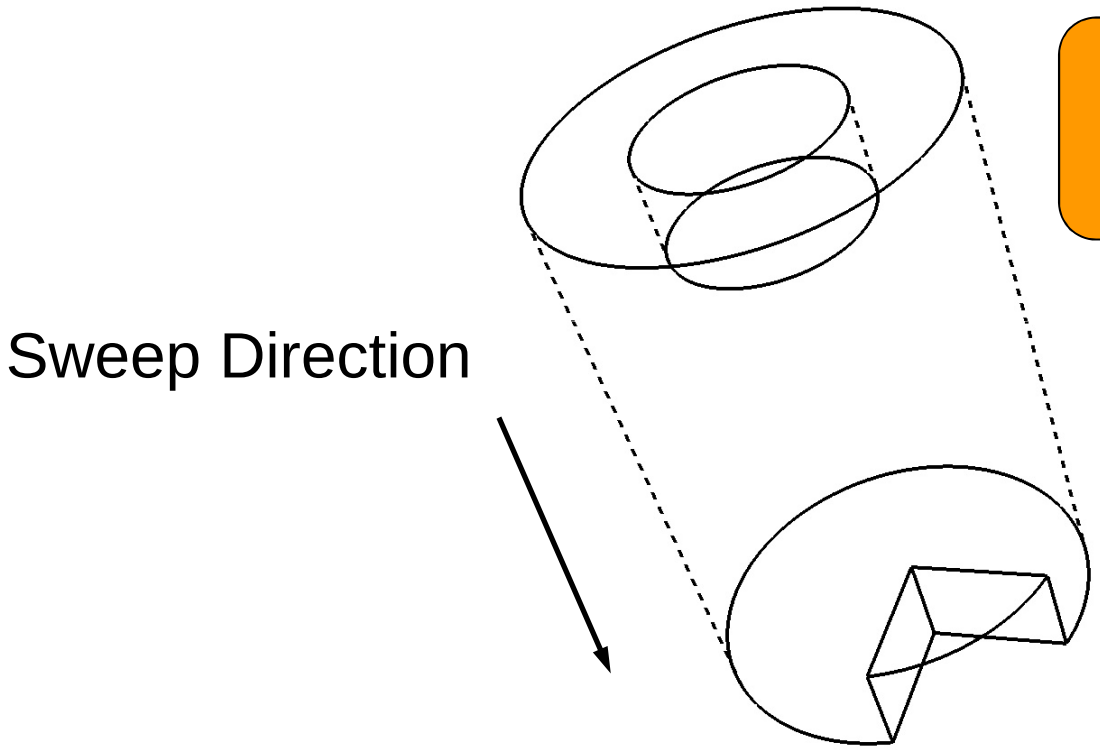
1-to-1 sweepable

Sweeping



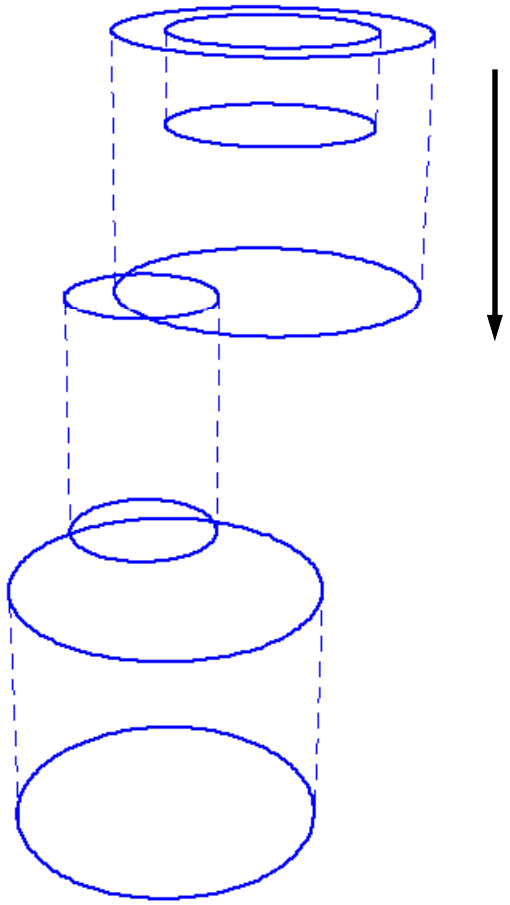
n-to-1 sweepable

Sweeping

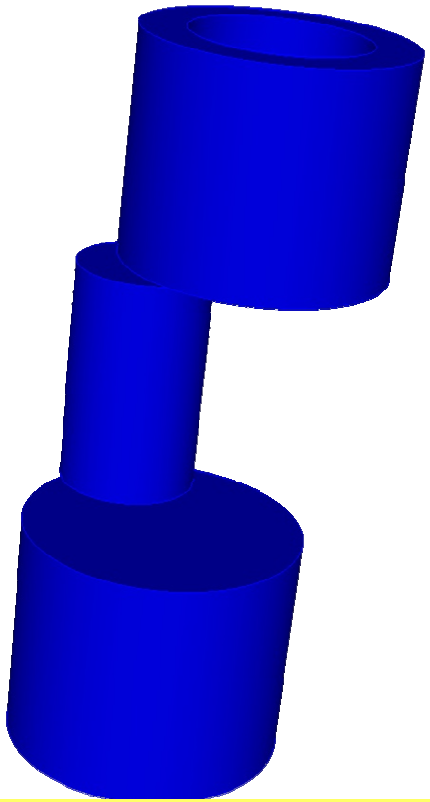


n-to-m sweepable
Multi-Sweep

Sweeping

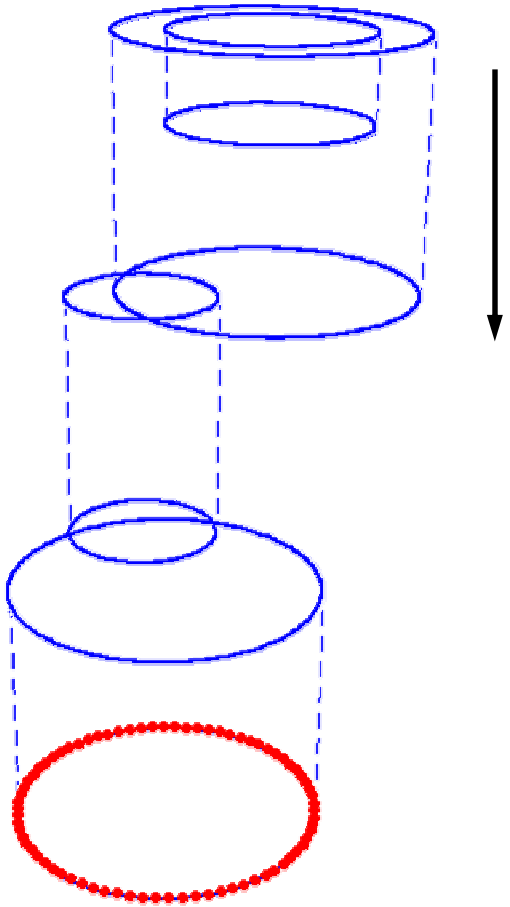


Sweep
Direction

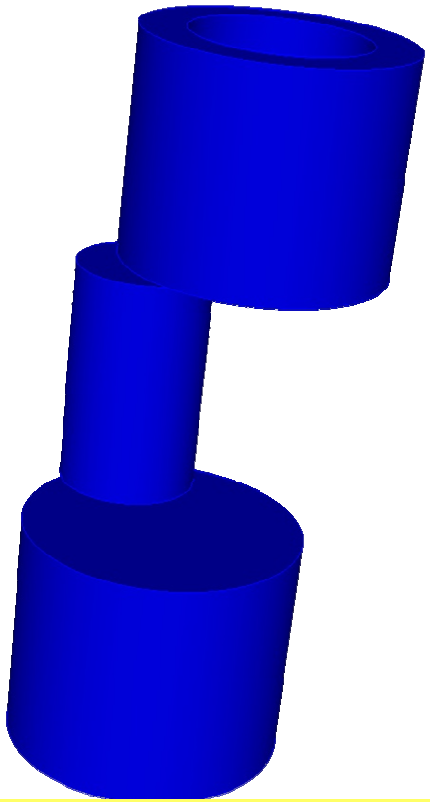


The fundamental strategy of multi-sweep is to convert an n-to-m sweepable volume into a number of n-to-1 sweepable volumes.

Sweeping

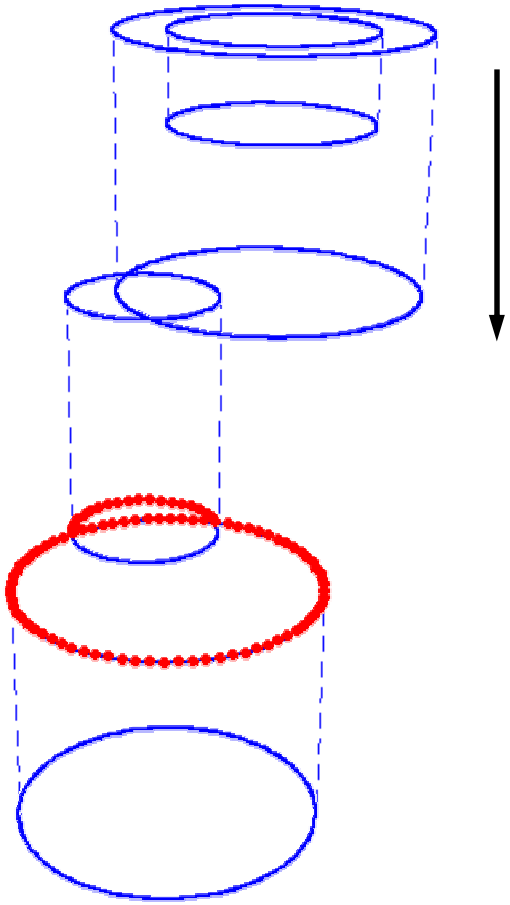


Sweep
Direction

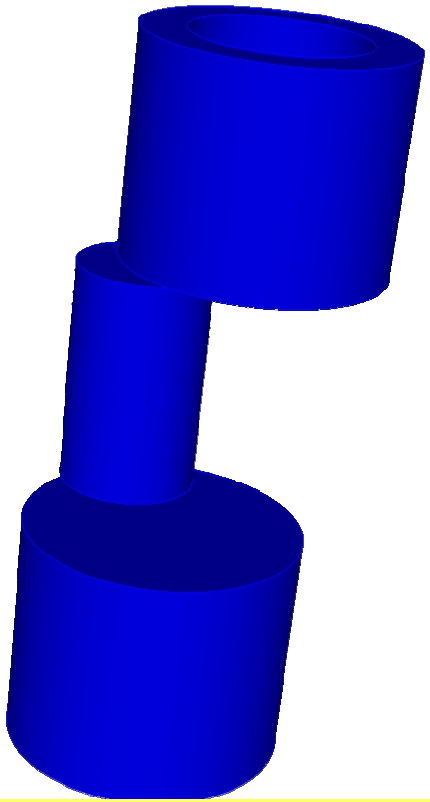


The fundamental strategy of multi-sweep is to convert an n-to-m sweepable volume into a number of n-to-1 sweepable volumes.

Sweeping

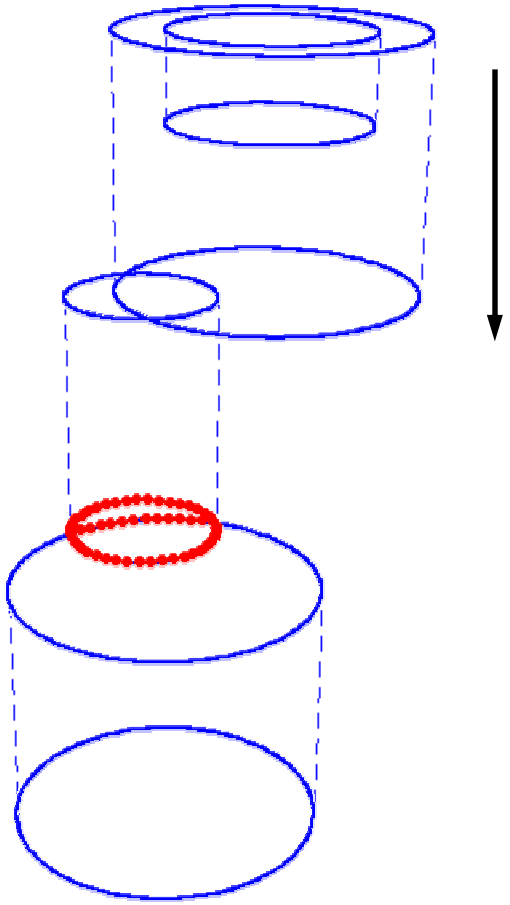


Sweep
Direction

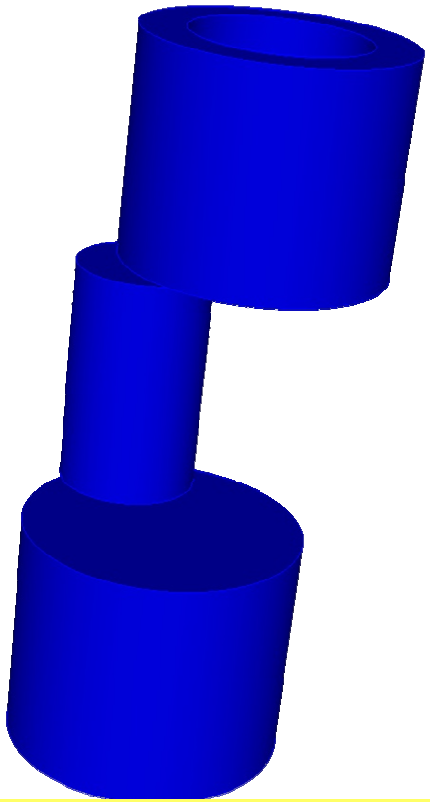


The fundamental strategy of multi-sweep is to convert an n-to-m sweepable volume into a number of n-to-1 sweepable volumes.

Sweeping

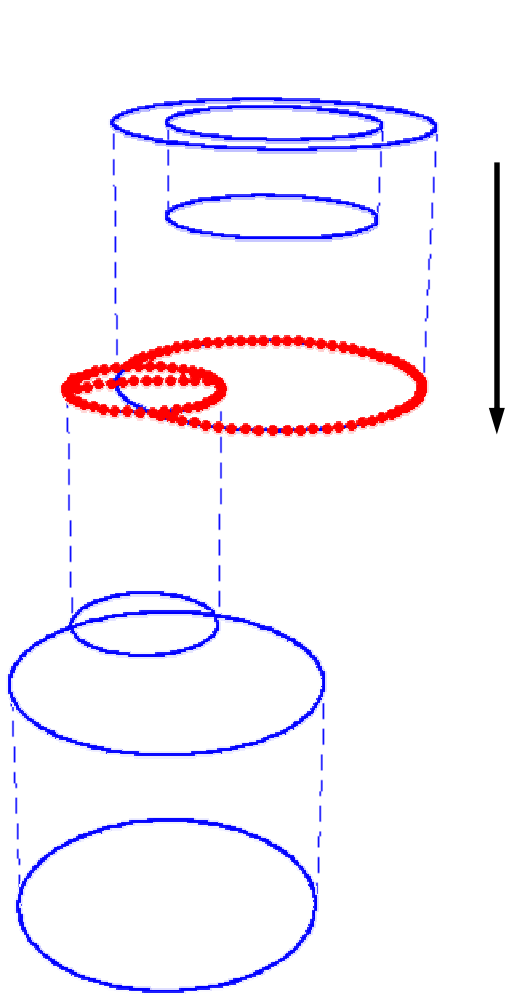


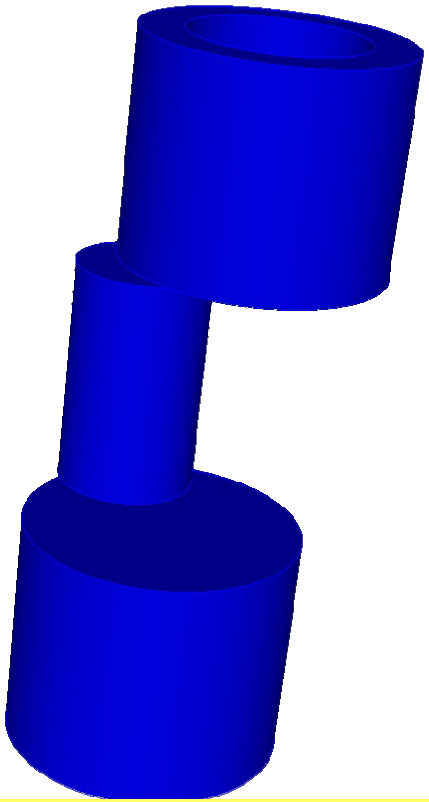
Sweep
Direction



The fundamental strategy of multi-sweep is to convert an n-to-m sweepable volume into a number of n-to-1 sweepable volumes.

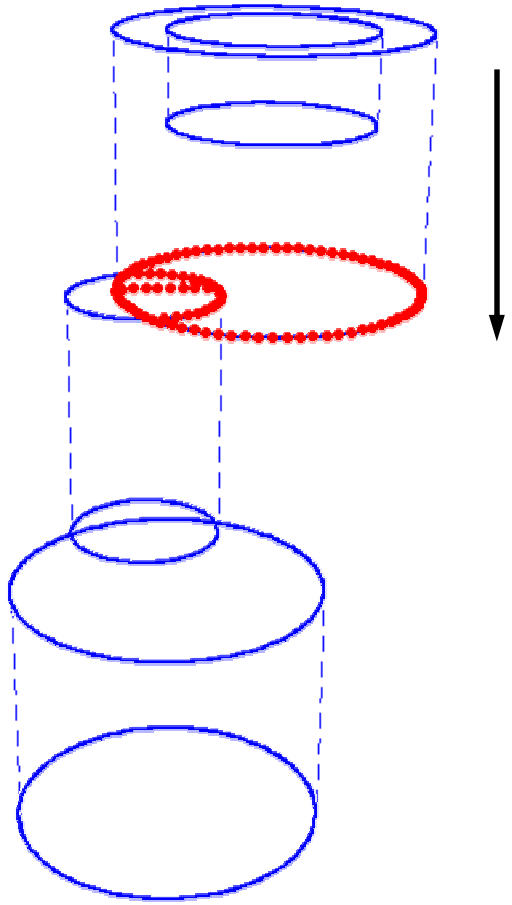
Sweeping



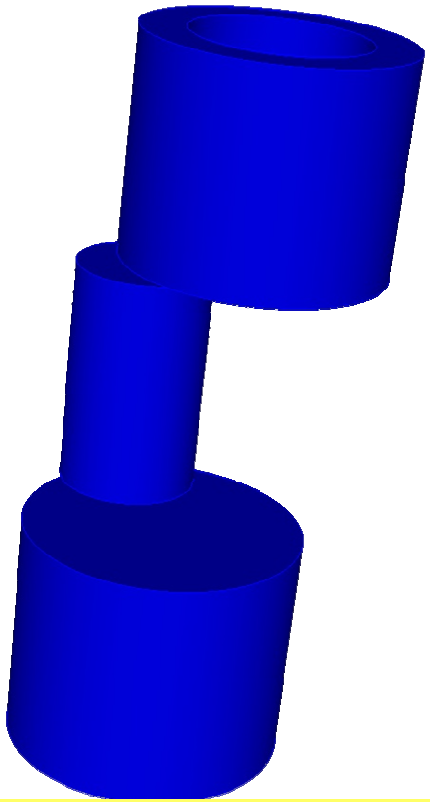


The fundamental strategy of multi-sweep is to convert an n-to-m sweepable volume into a number of n-to-1 sweepable volumes.

Sweeping

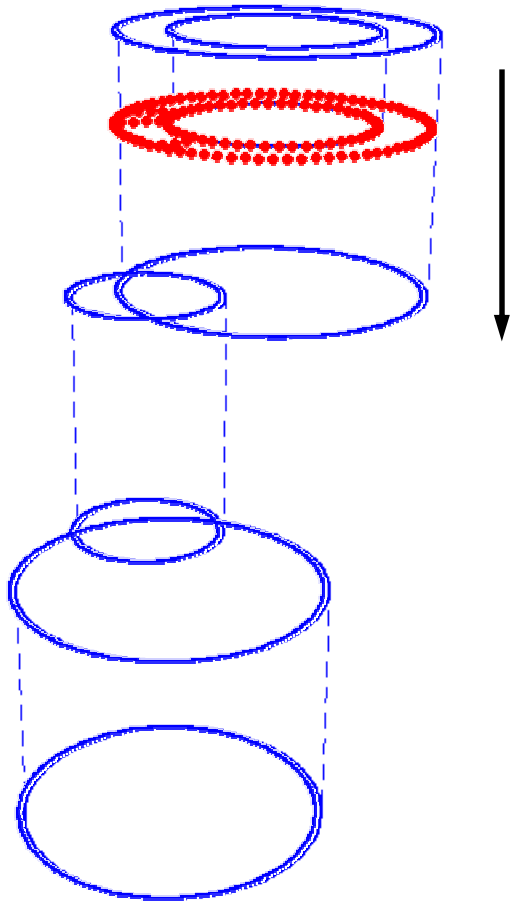


Sweep
Direction

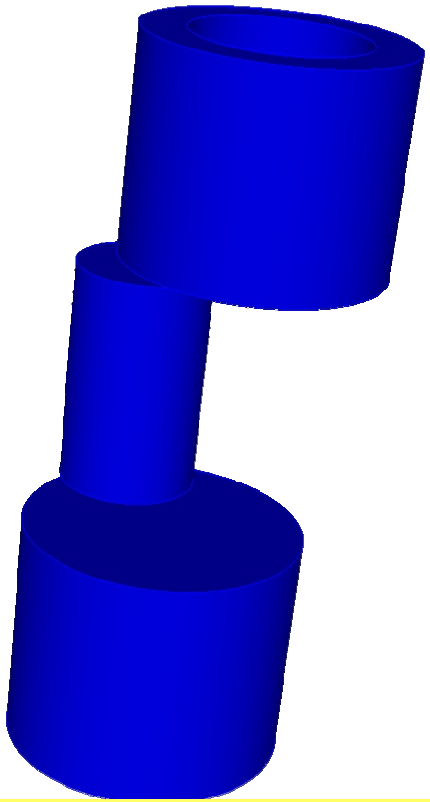


The fundamental strategy of multi-sweep is to convert an n-to-m sweepable volume into a number of n-to-1 sweepable volumes.

Sweeping

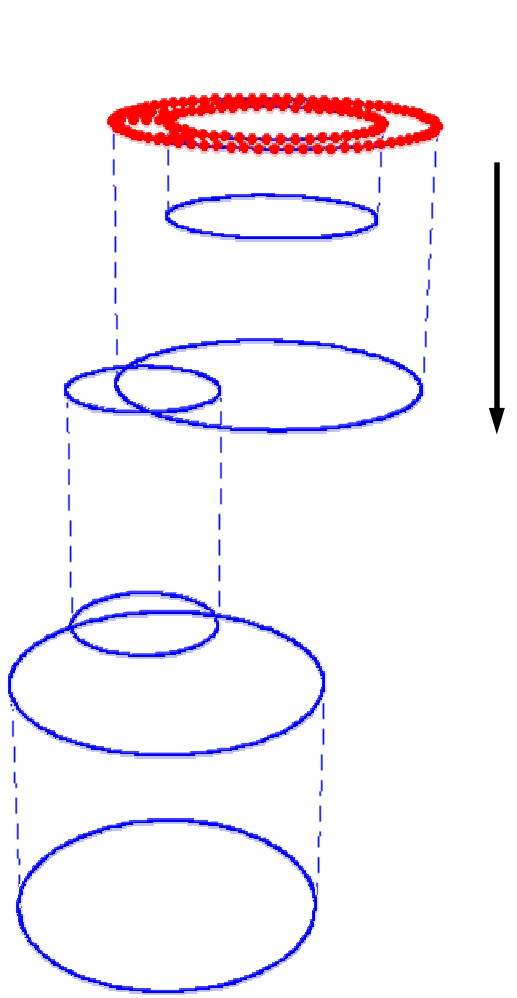


Sweep
Direction

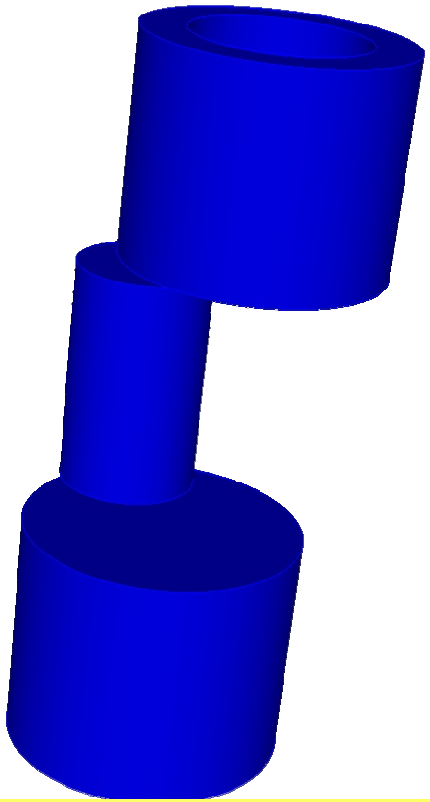


The fundamental strategy of multi-sweep is to convert an n-to-m sweepable volume into a number of n-to-1 sweepable volumes.

Sweeping

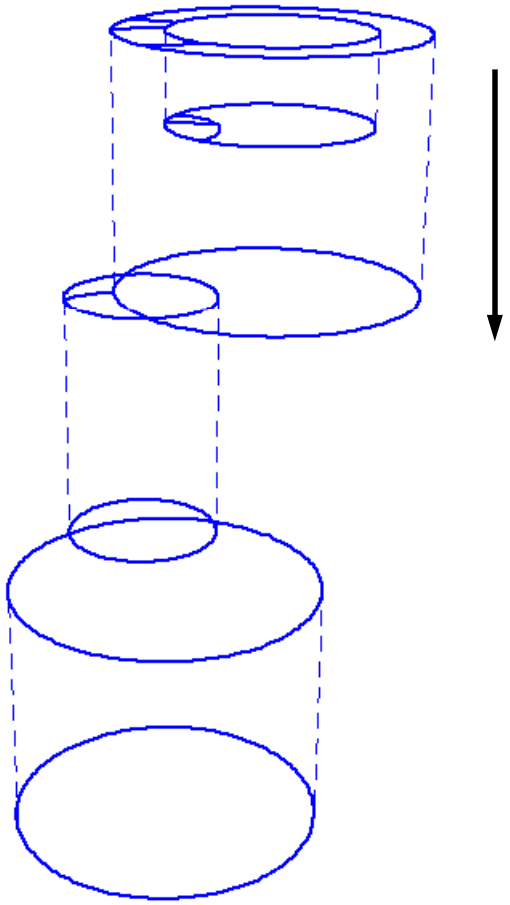


Sweep
Direction

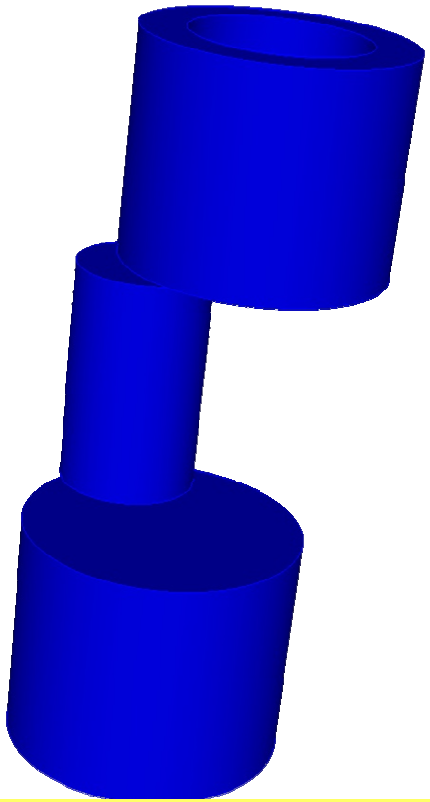


The fundamental strategy of multi-sweep is to convert an n-to-m sweepable volume into a number of n-to-1 sweepable volumes.

Sweeping

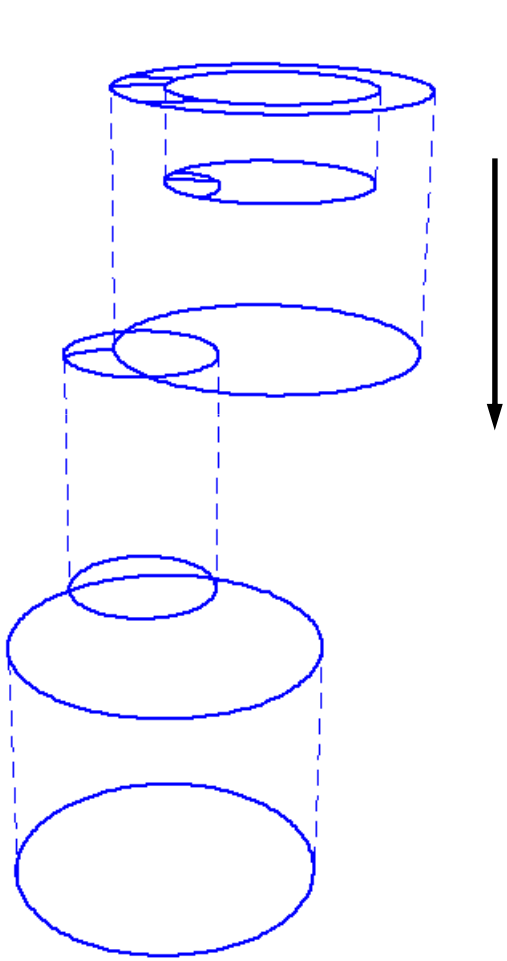


Sweep
Direction

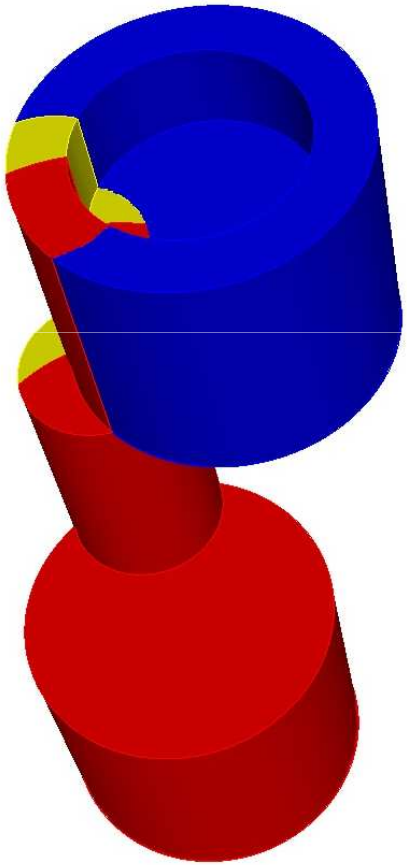


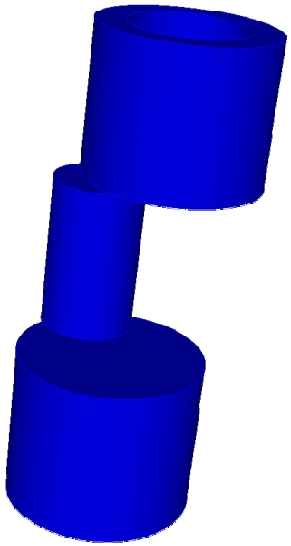
The fundamental strategy of multi-sweep is to convert an n-to-m sweepable volume into a number of n-to-1 sweepable volumes.

Decomposition / Sweep Overview

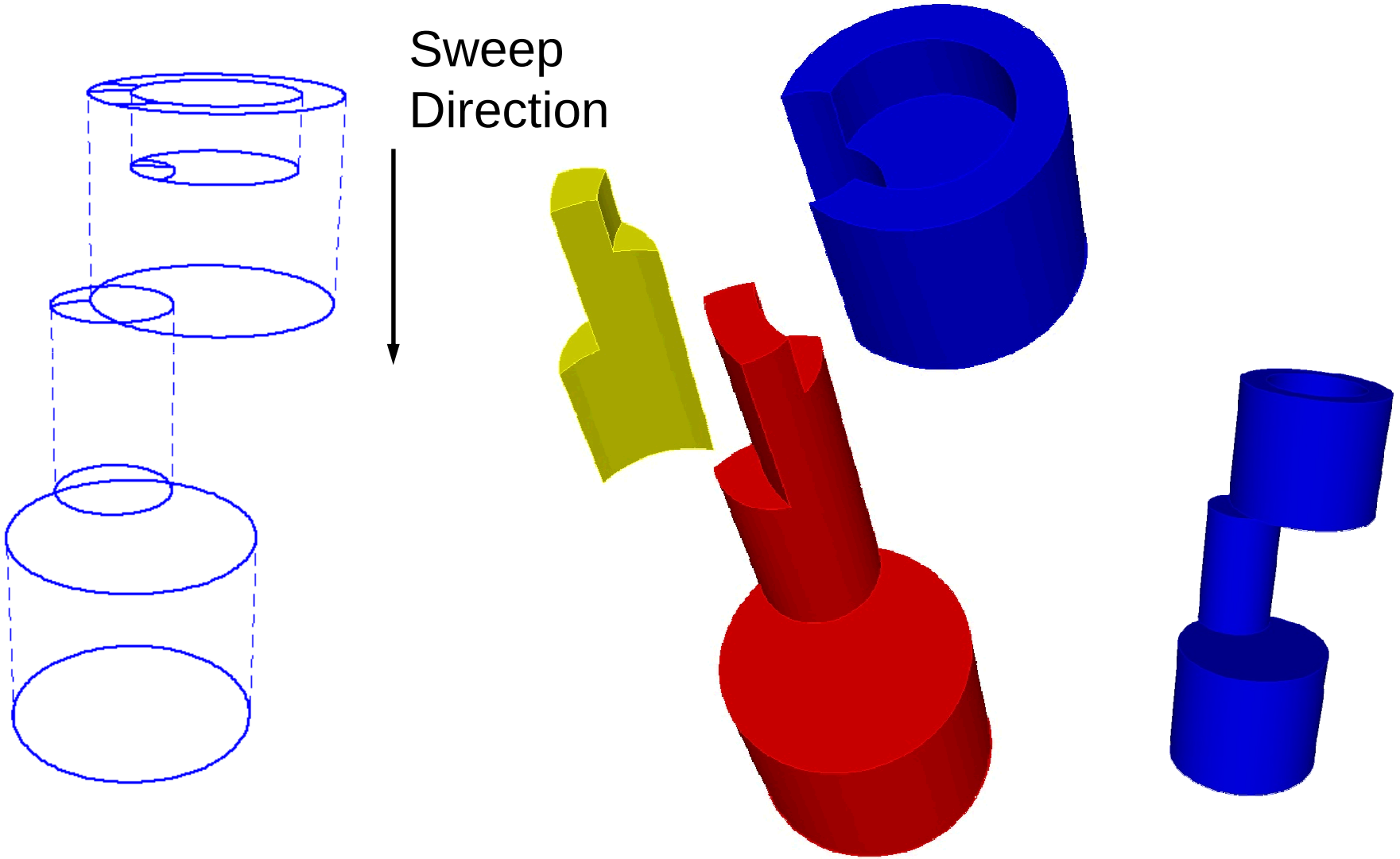


Sweep
Direction

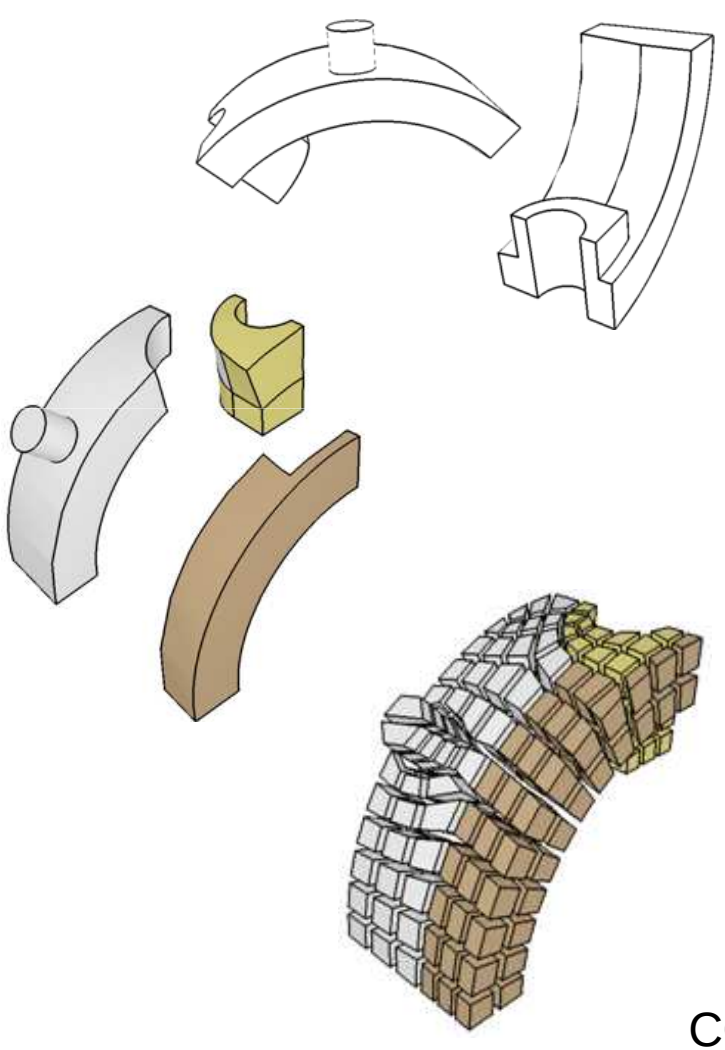


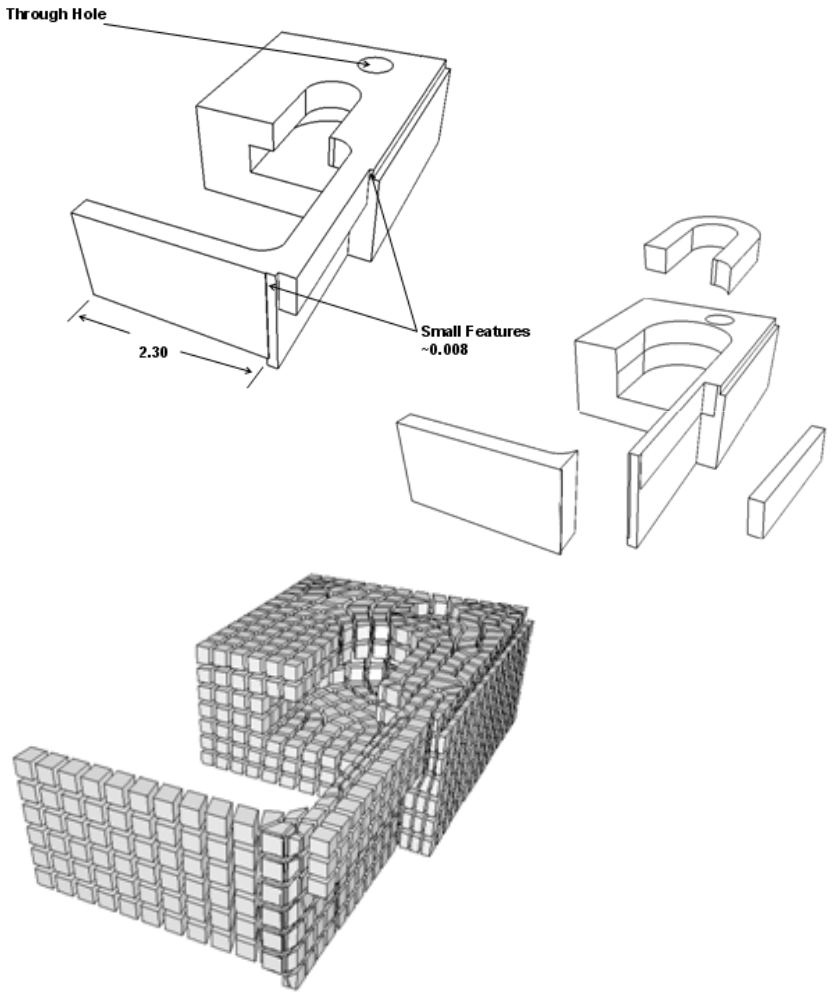


Decomposition / Sweep Overview



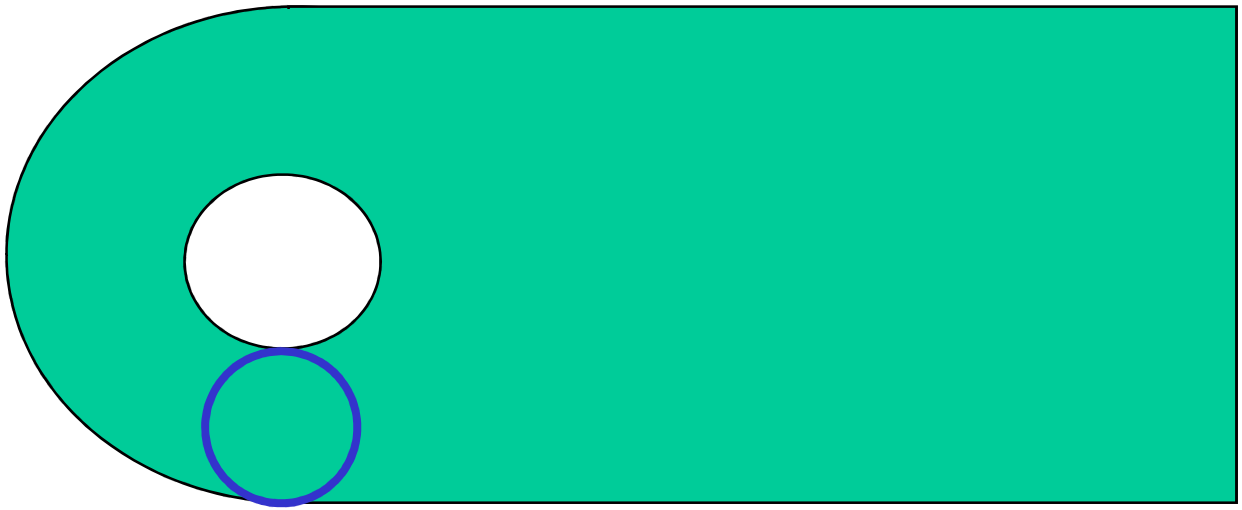
Sweeping





CCSweep (White, 2004)

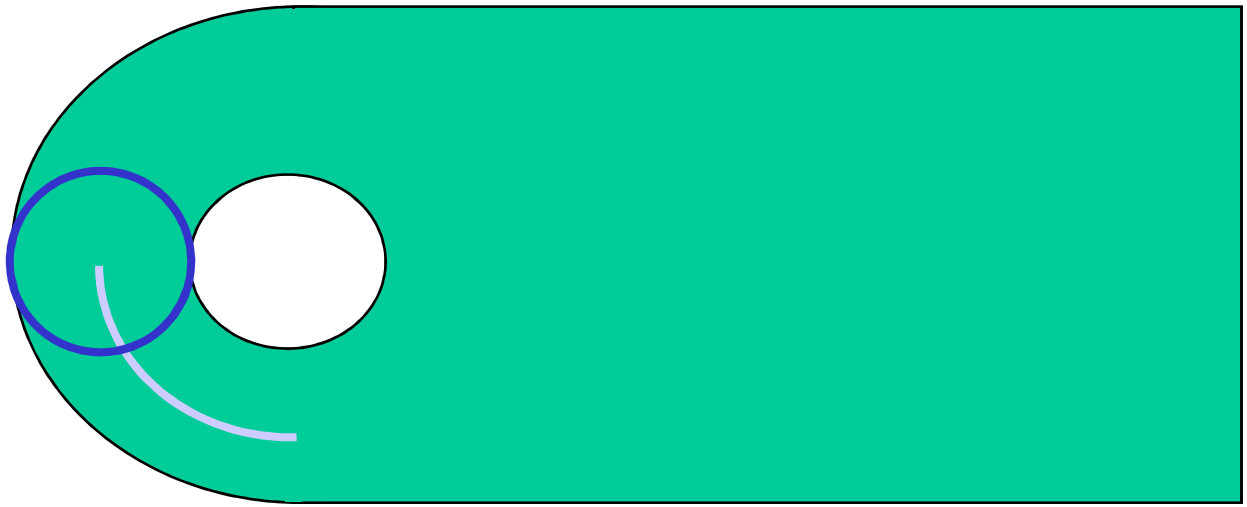
Medial Axis



- Medial Object - Roll a Maximal circle or sphere through the model. The center traces the medial object
- Medial Object used as a tool to automatically decompose model into simpler mapable or sweepable parts

(Price, 95;97)(Tam,91)

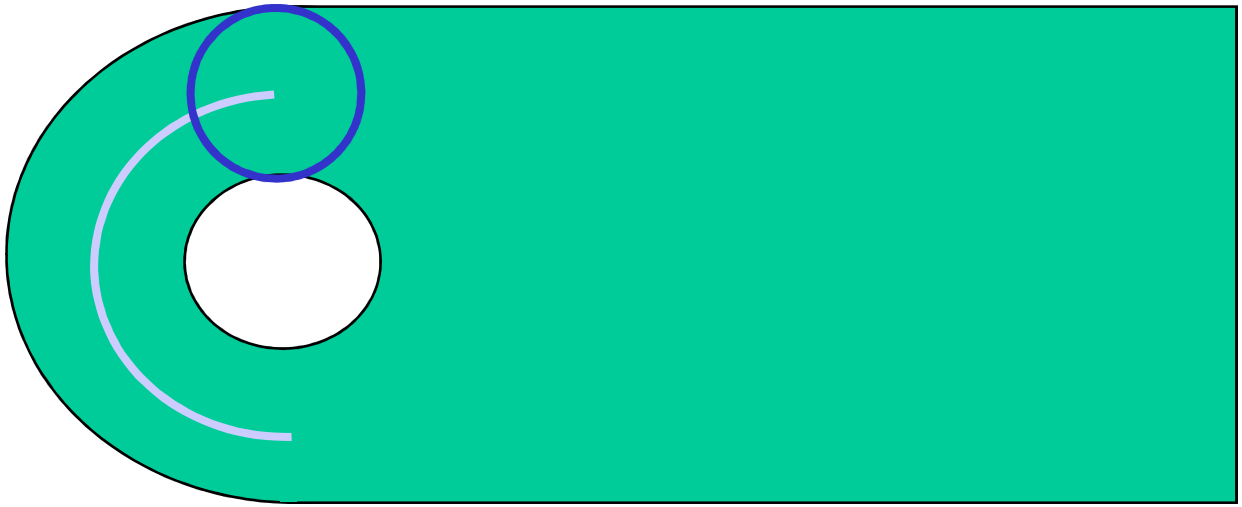
Medial Axis



- Medial Object - Roll a Maximal circle or sphere through the model. The center traces the medial object
- Medial Object used as a tool to automatically decompose model into simpler mapable or sweepable parts

(Price, 95;97)(Tam,91)

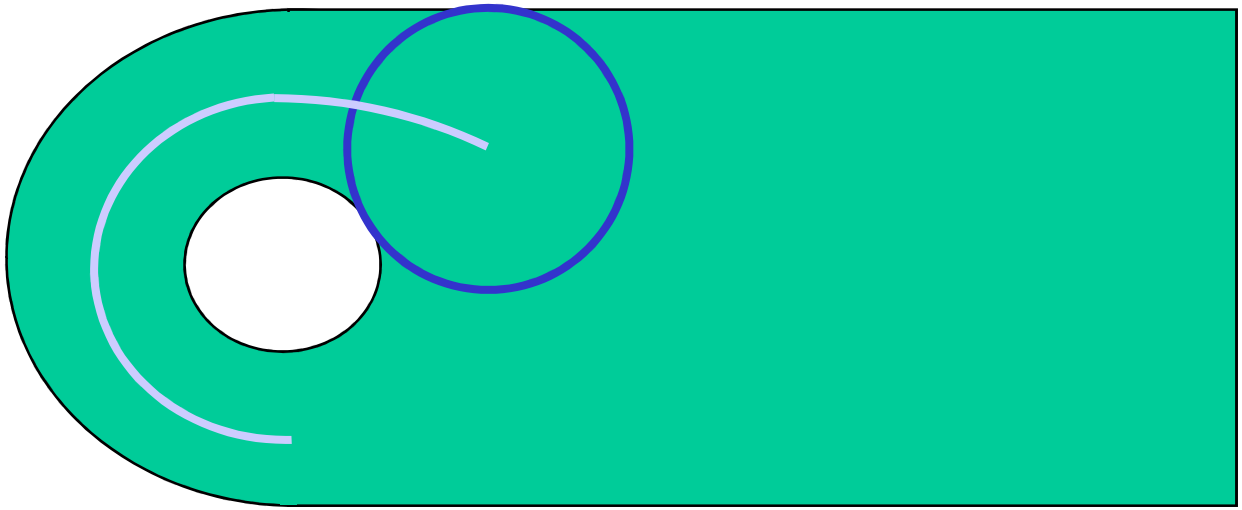
Medial Axis



- Medial Object - Roll a Maximal circle or sphere through the model. The center traces the medial object
- Medial Object used as a tool to automatically decompose model into simpler mapable or sweepable parts

(Price, 95;97)(Tam,91)

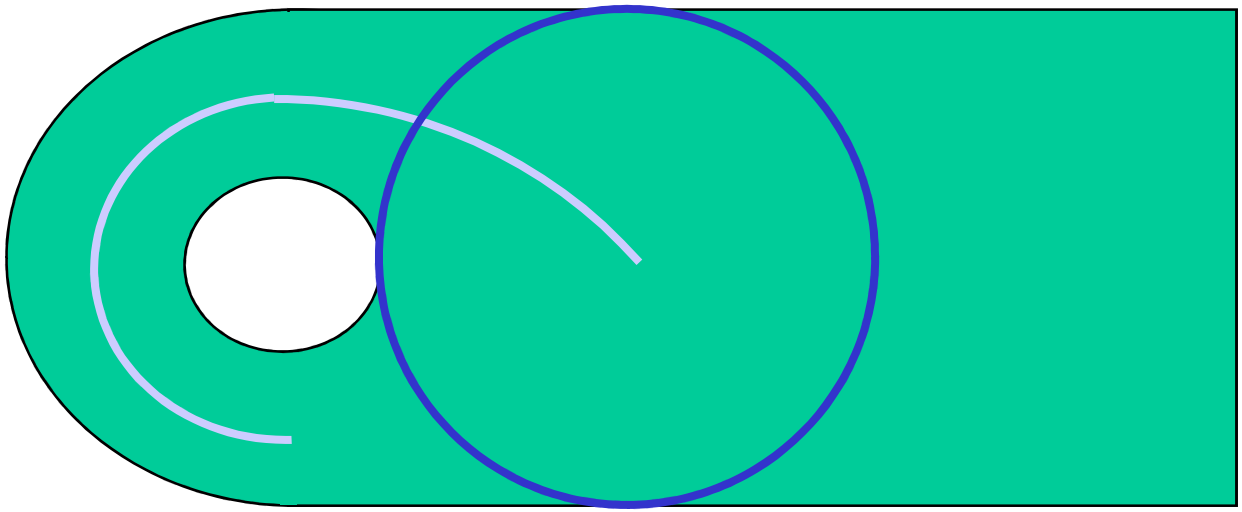
Medial Axis



- Medial Object - Roll a Maximal circle or sphere through the model. The center traces the medial object
- Medial Object used as a tool to automatically decompose model into simpler mapable or sweepable parts

(Price, 95;97)(Tam,91)

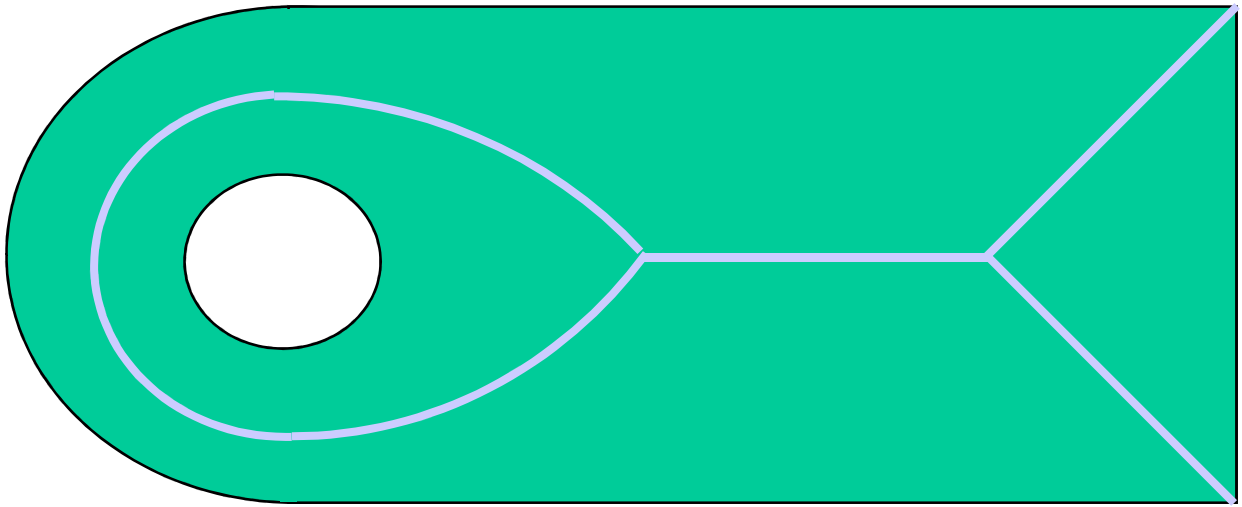
Medial Axis



- Medial Object - Roll a Maximal circle or sphere through the model. The center traces the medial object
- Medial Object used as a tool to automatically decompose model into simpler mapable or sweepable parts

(Price, 95;97)(Tam,91)

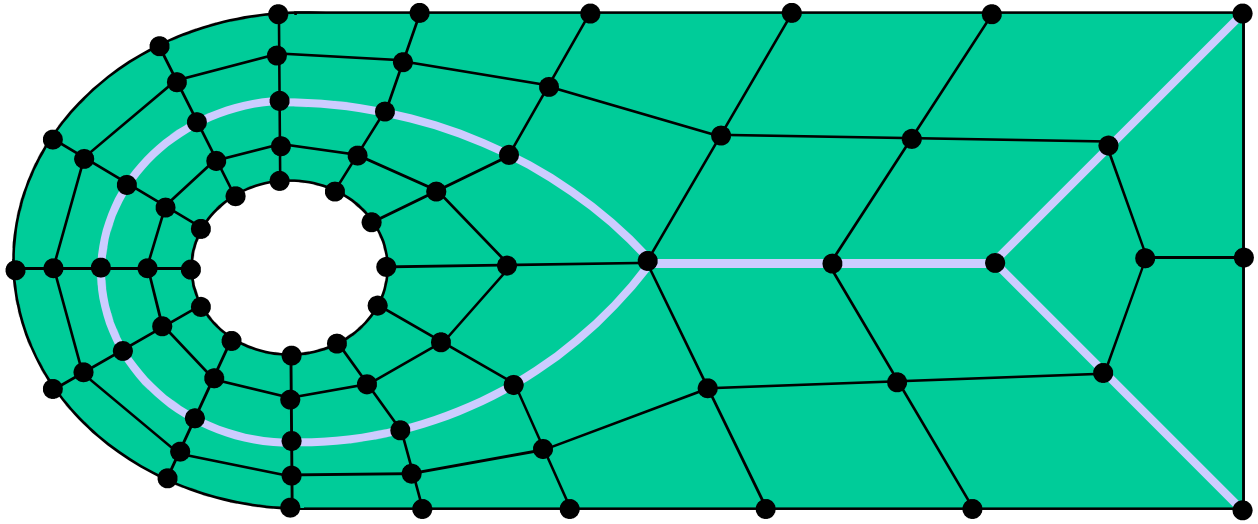
Medial Axis



- Medial Object - Roll a Maximal circle or sphere through the model. The center traces the medial object
- Medial Object used as a tool to automatically decompose model into simpler mapable or sweepable parts

(Price, 95;97)(Tam,91)

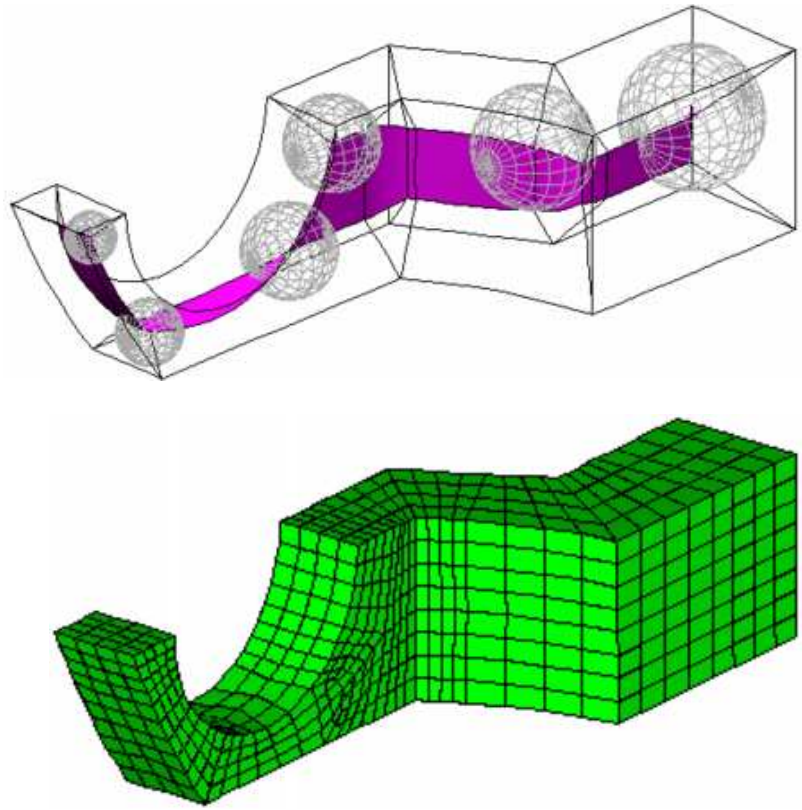
Medial Axis



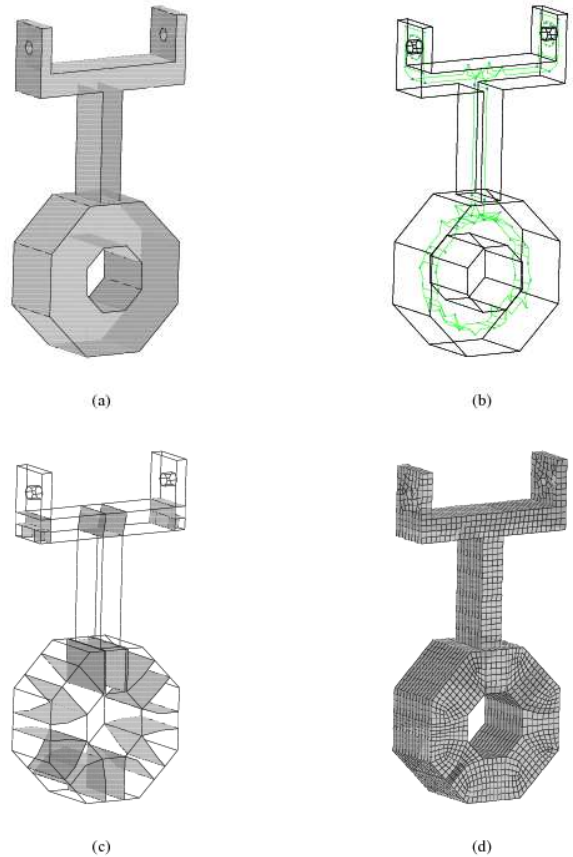
- Medial Object - Roll a Maximal circle or sphere through the model. The center traces the medial object
- Medial Object used as a tool to automatically decompose model into simpler mapable or sweepable parts

(Price, 95;97)(Tam,91)

Medial Axis

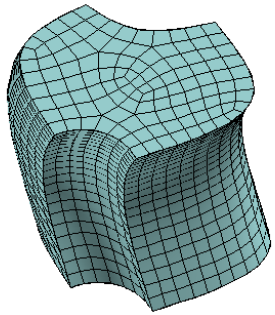


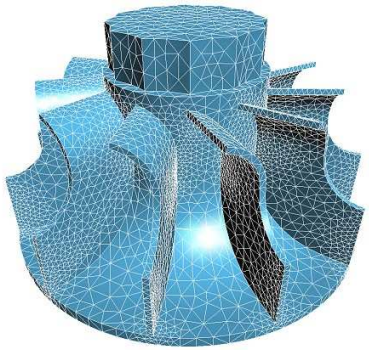
Medial Axis + Midpoint Subdivision
(Price, 95)

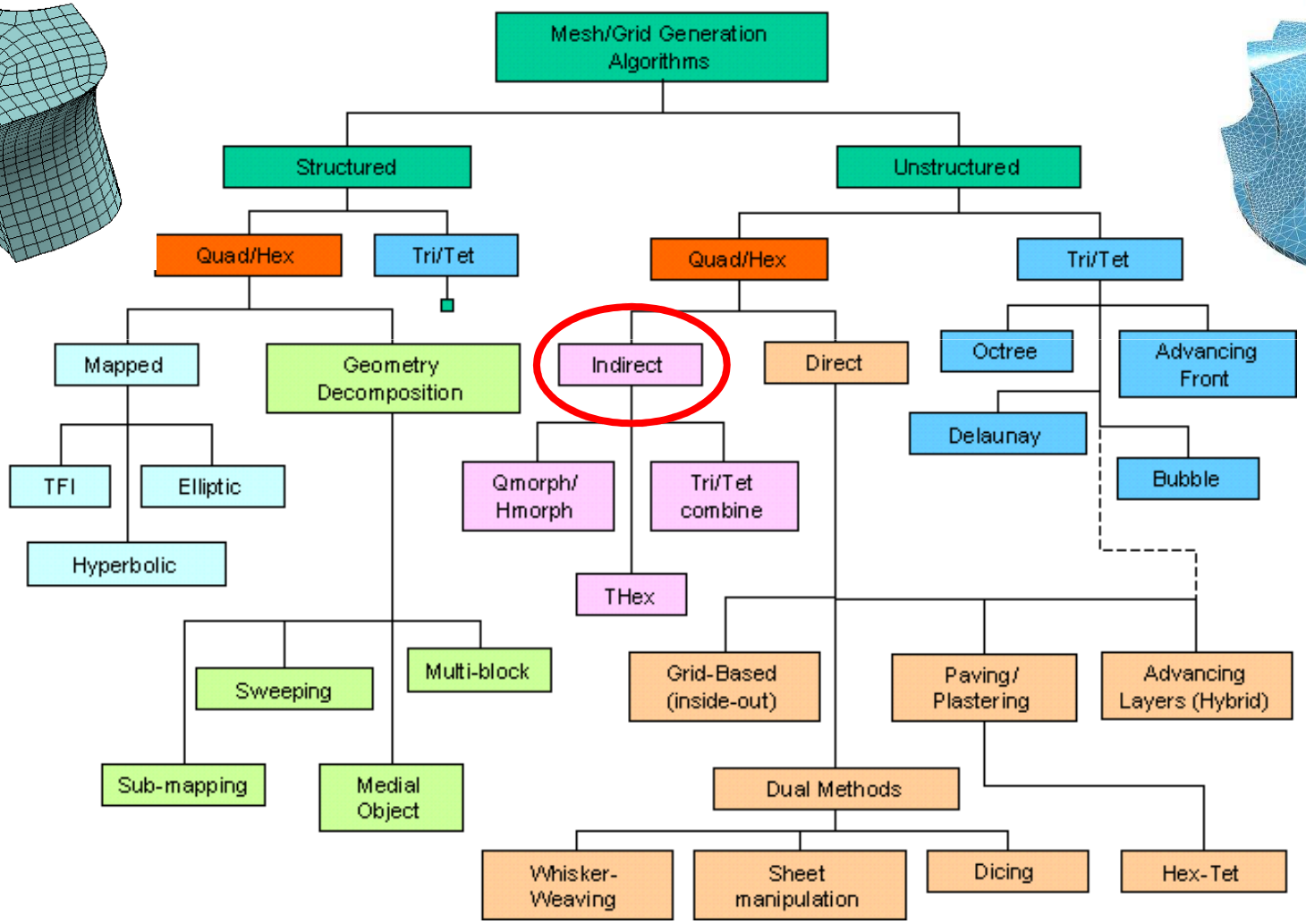


Embedded Voronoï Graph
(Sheffer, 98)

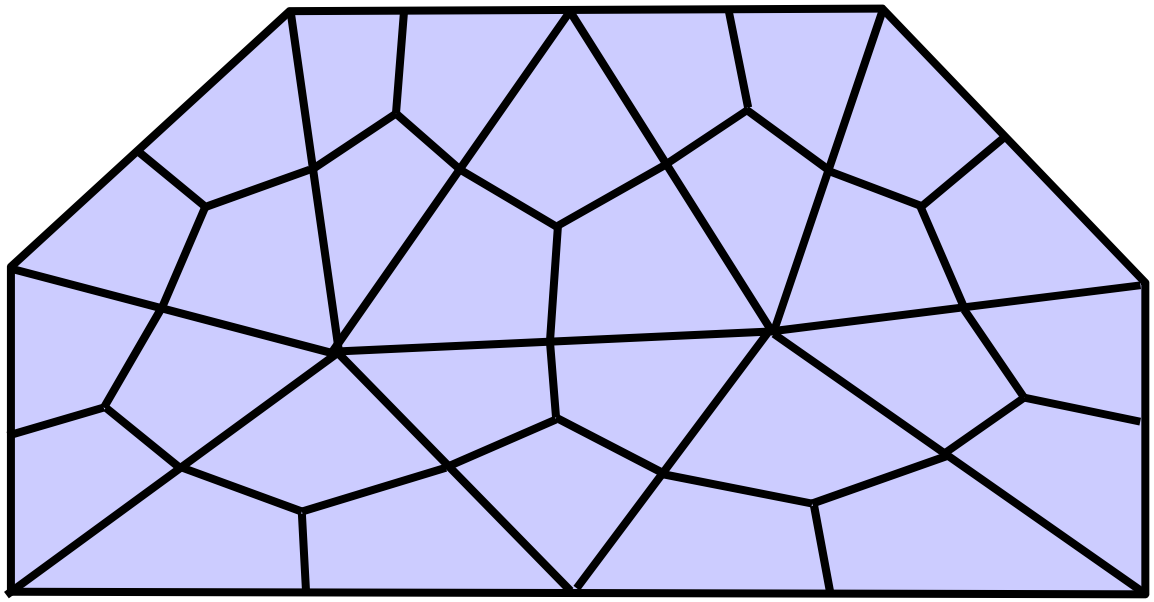
Meshing Algorithms







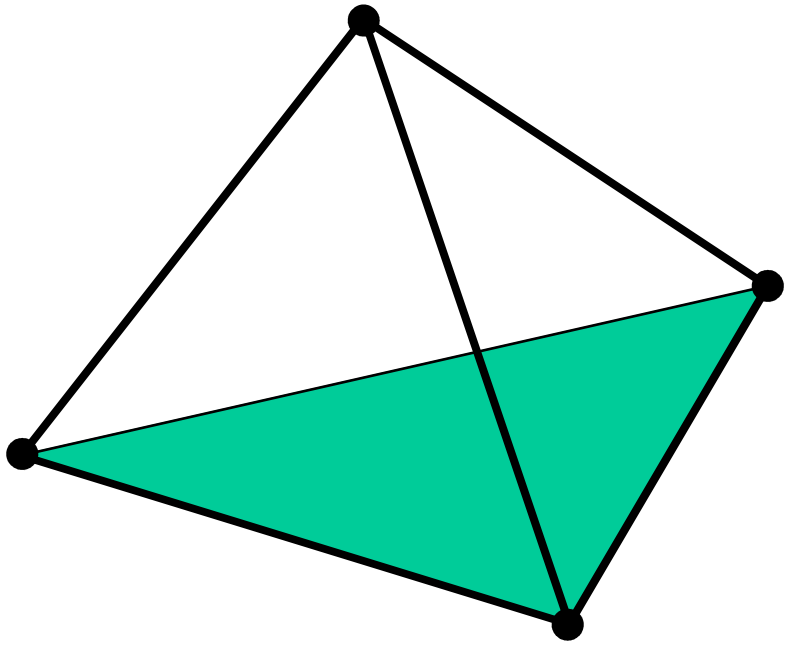
Unstructured: Indirect Quad



Triangle splitting

- Each triangle split into 3 quads
- Typically results in poor angles

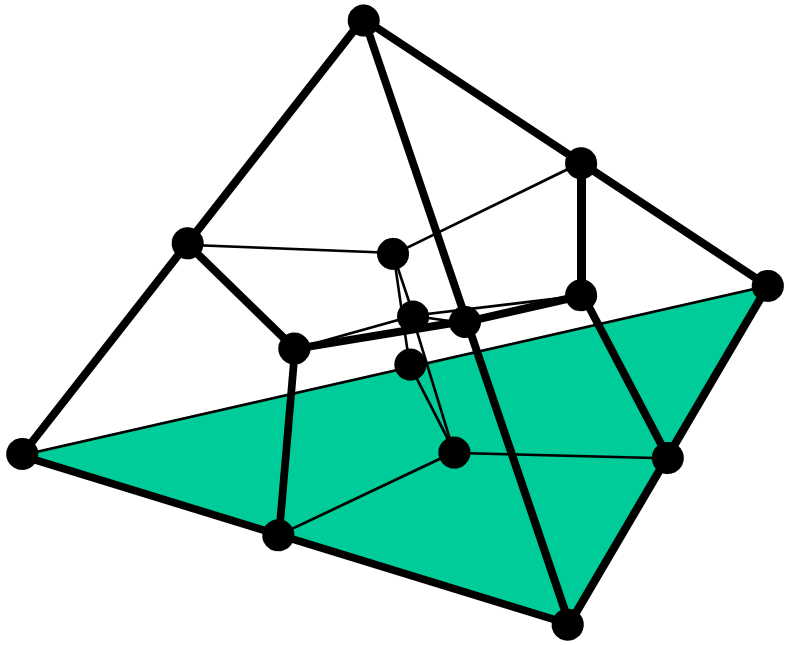
Unstructured: Indirect Hex



Tetrahedron splitting

- Each tetrahedron split into 4 hexahedra
- Typically results in poor angles (Taniguchi, 96)

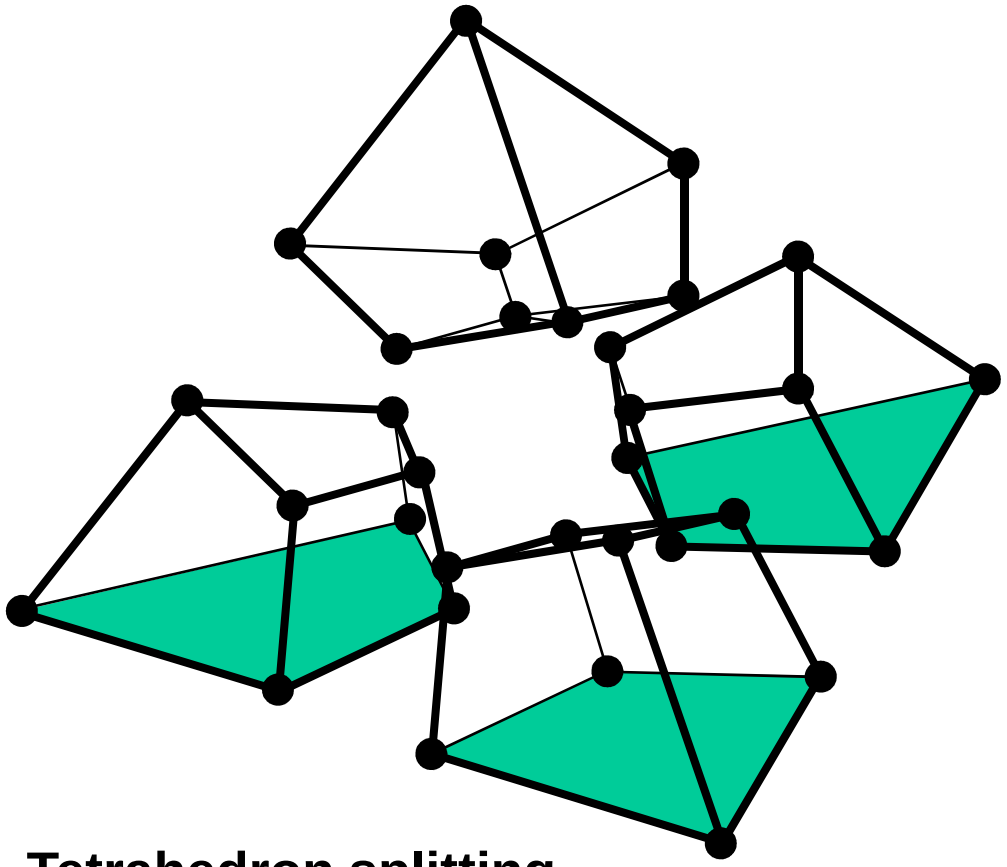
Unstructured: Indirect Hex



Tetrahedron splitting

- Each tetrahedron split into 4 hexahedra
- Typically results in poor angles (Taniguchi, 96)

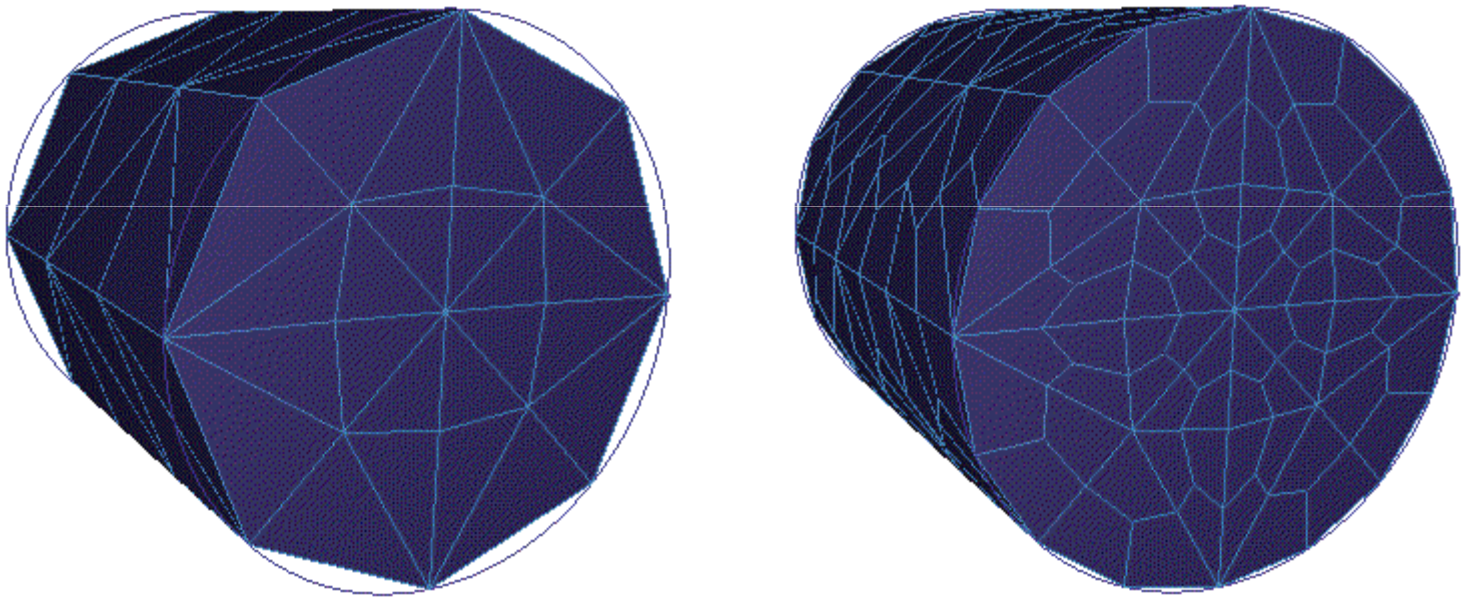
Unstructured: Indirect Hex



Tetrahedron splitting

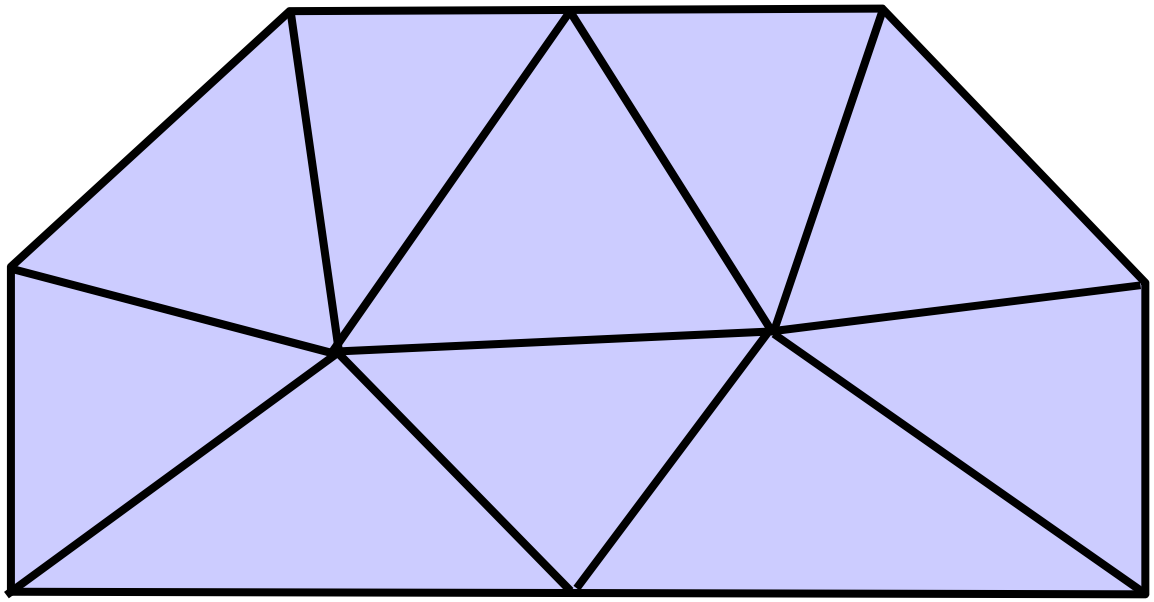
- Each tetrahedron split into 4 hexahedra
- Typically results in poor angles (Taniguchi, 96)

Unstructured: Indirect Hex



Example of geometry meshed by tetrahedron splitting

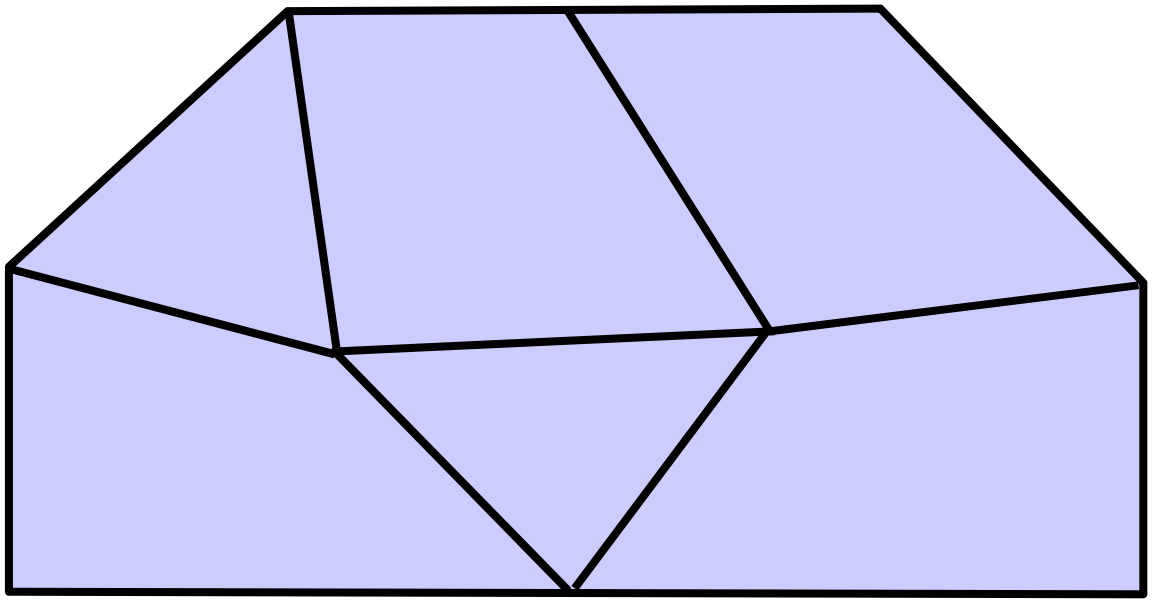
Unstructured: Indirect Quad



Triangle Merge

- Two adjacent triangles combined into a single quad
- Test for best local choice for combination
- Triangles can remain if attention is not paid to order of combination

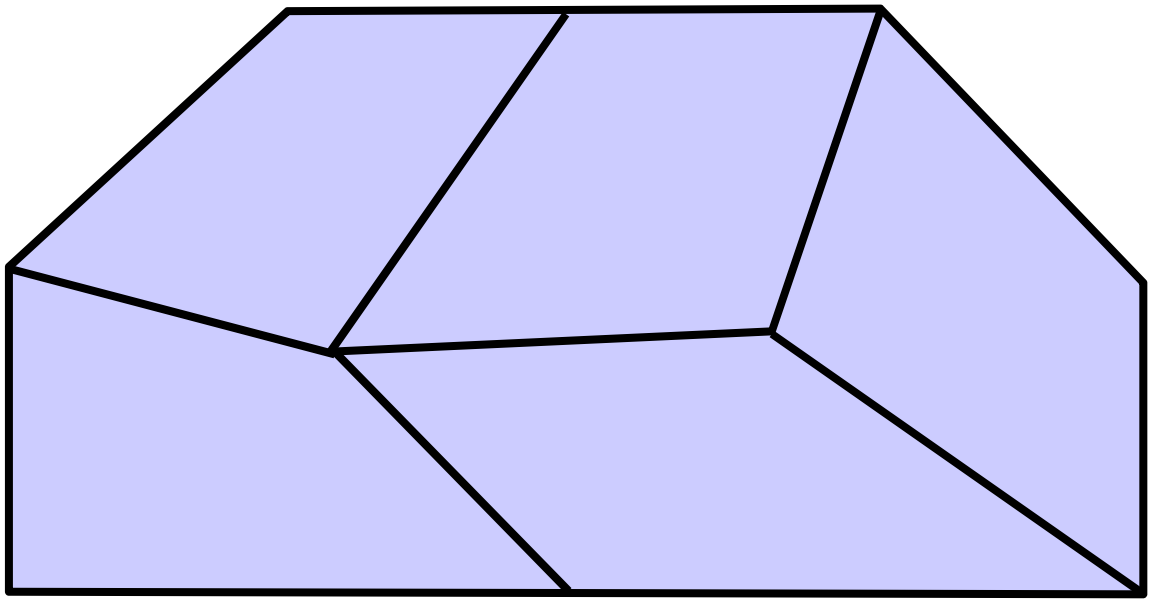
Unstructured: Indirect Quad



Triangle Merge

- Two adjacent triangles combined into a single quad
- Test for best local choice for combination
- Triangles can remain if attention is not paid to order of combination

Unstructured: Indirect Quad



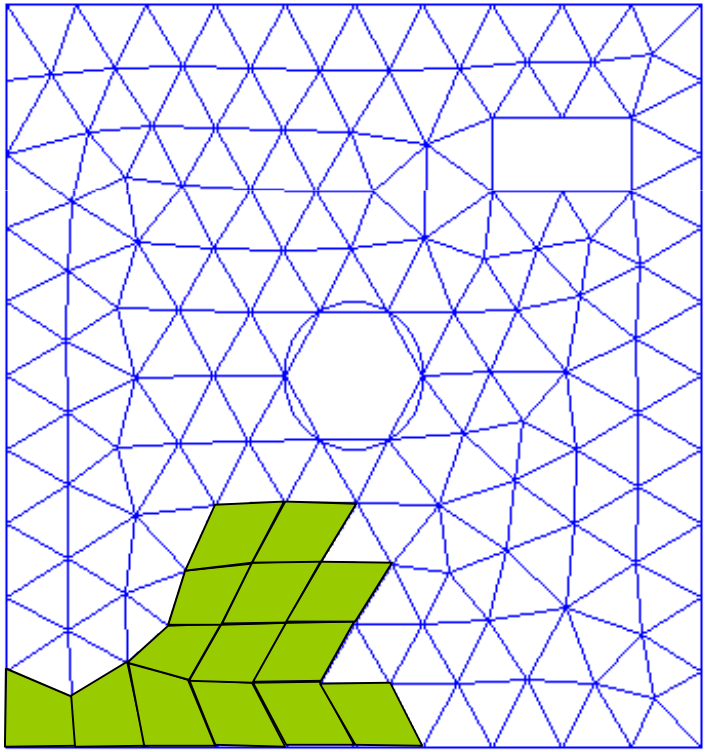
Triangle Merge

- Two adjacent triangles combined into a single quad
- Test for best local choice for combination
- Triangles can remain if attention is not paid to order of combination

Unstructured: Indirect Quad

Directed Triangle Merge

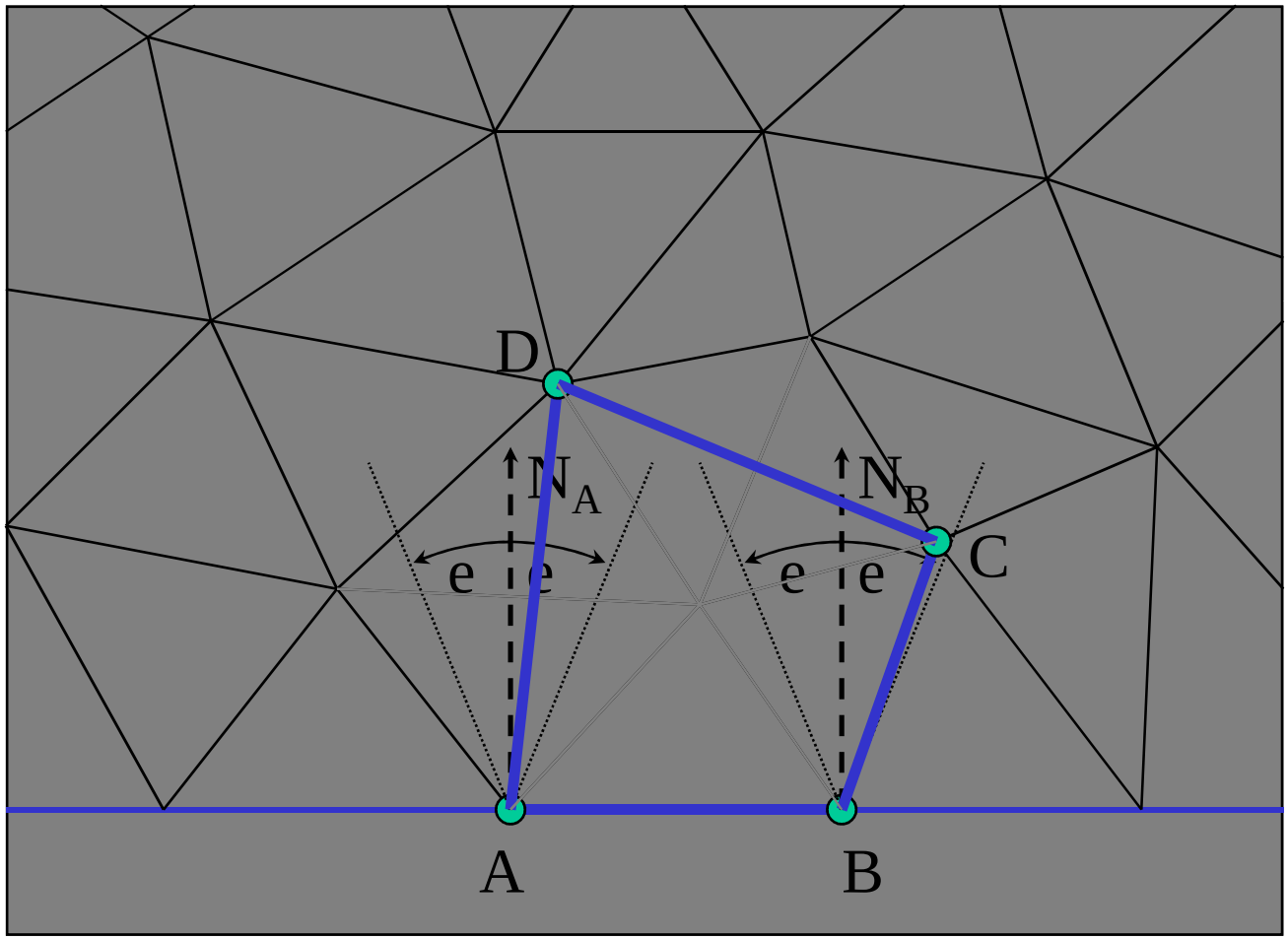
- Merging begins at a boundary
- Advances from one set of triangles to the next
- Attempts to maintain even number of intervals on any loop
- Can produce all-quad mesh
- Can also incorporate triangle splitting (Lee and Lo, 94)



Unstructured: Indirect Quad

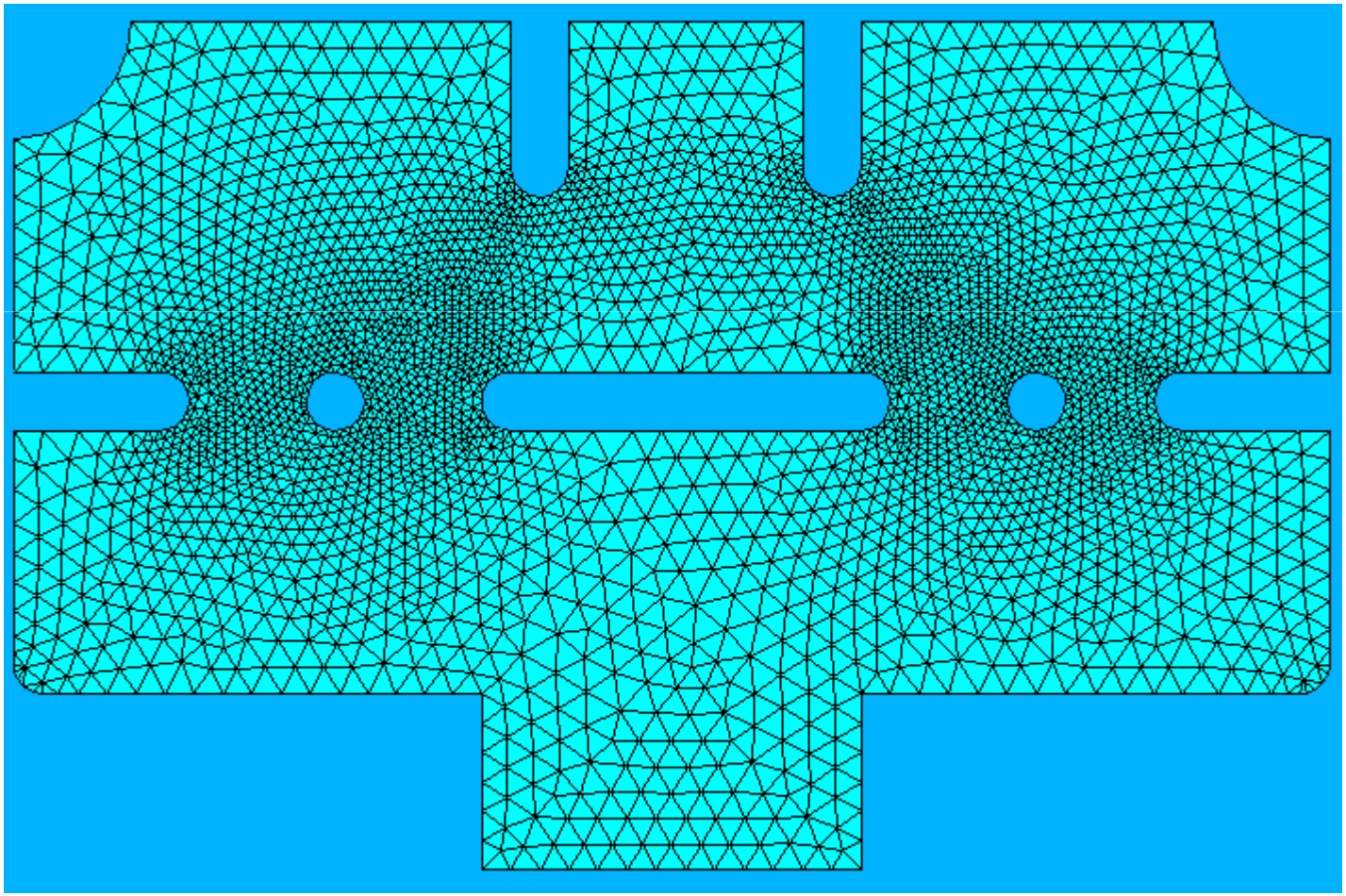
Triangle Merge w/ local transformations (“Q-Morph”)

- Uses an advancing front approach
- Local swaps applied to improve resulting quad
- Any number of triangles merged to create a quad
- Attempts to maintain even number of intervals on any loop
- Produces all-quad mesh from even intervals (Owen, 99)



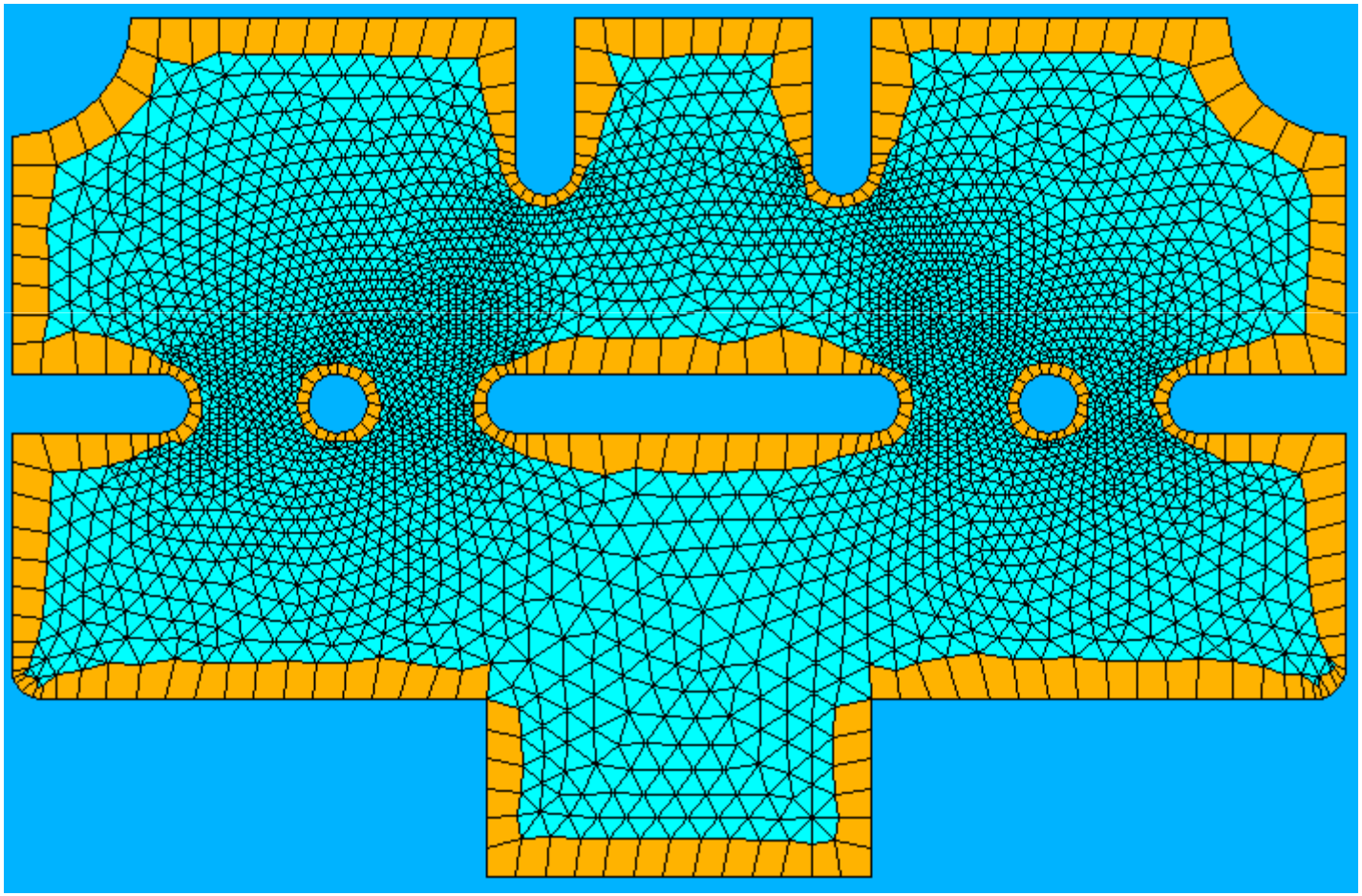
Unstructured: Indirect Quad

Q-Morph



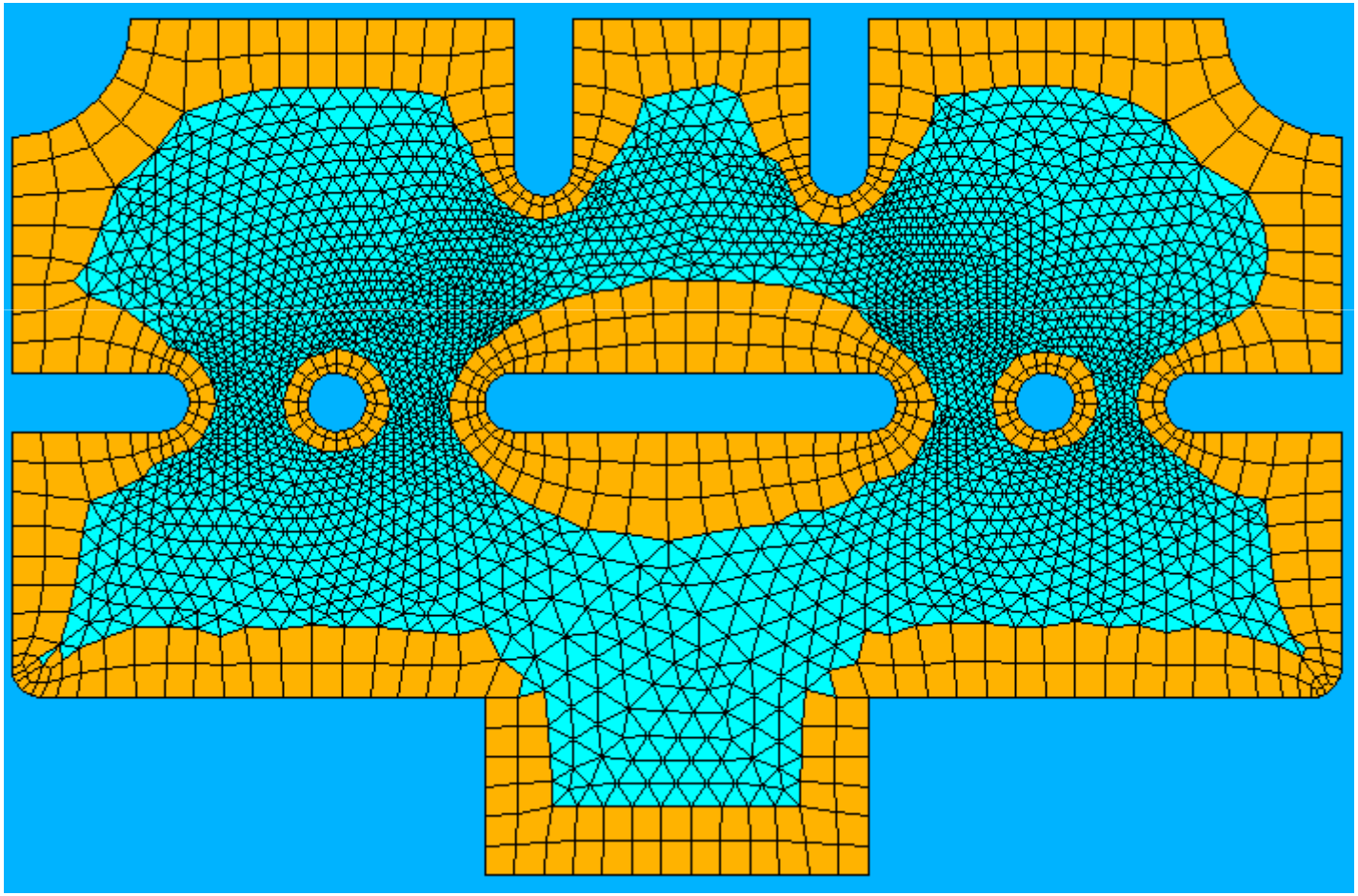
Unstructured: Indirect Quad

Q-Morph



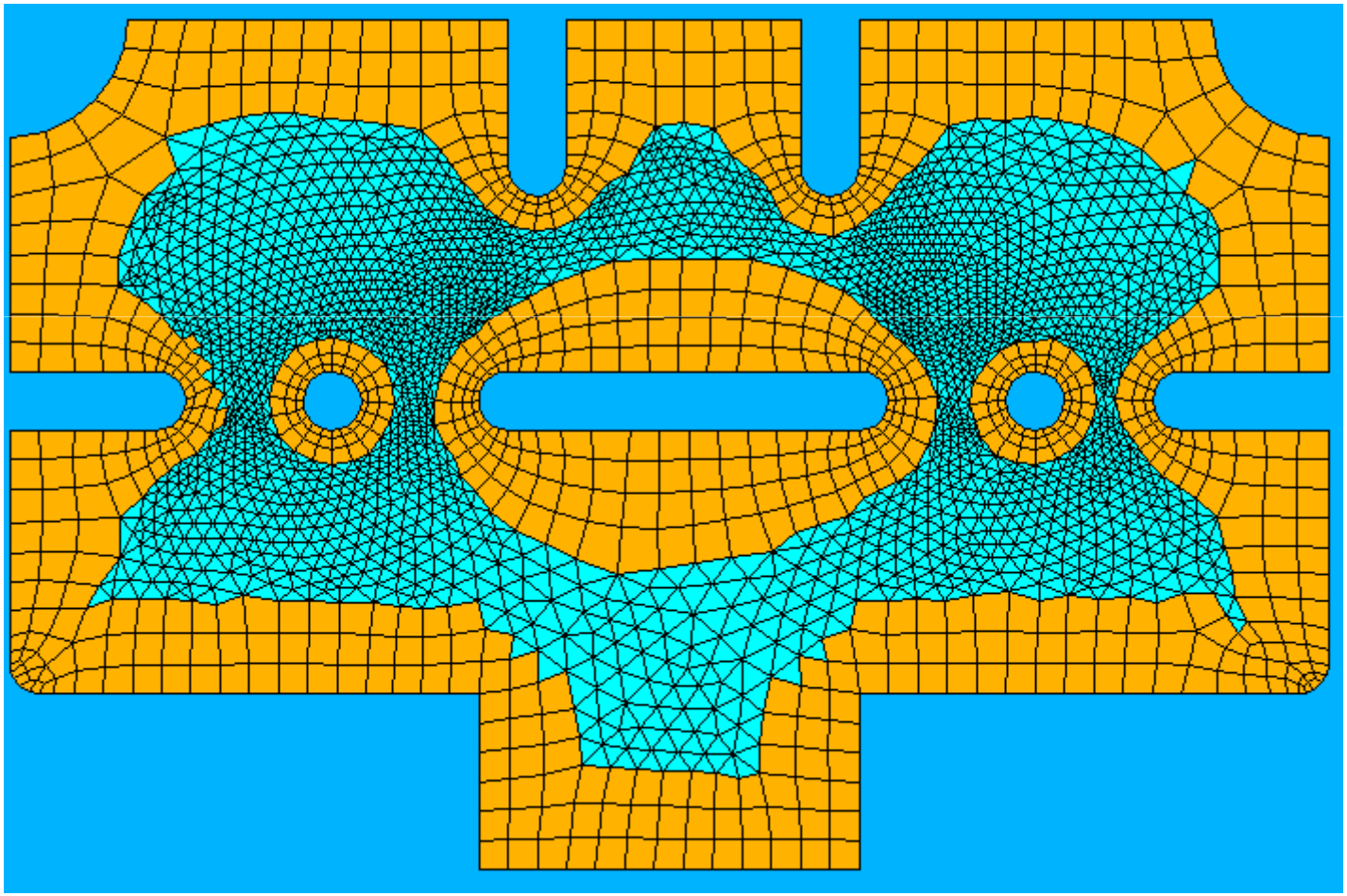
Unstructured: Indirect Quad

Q-Morph



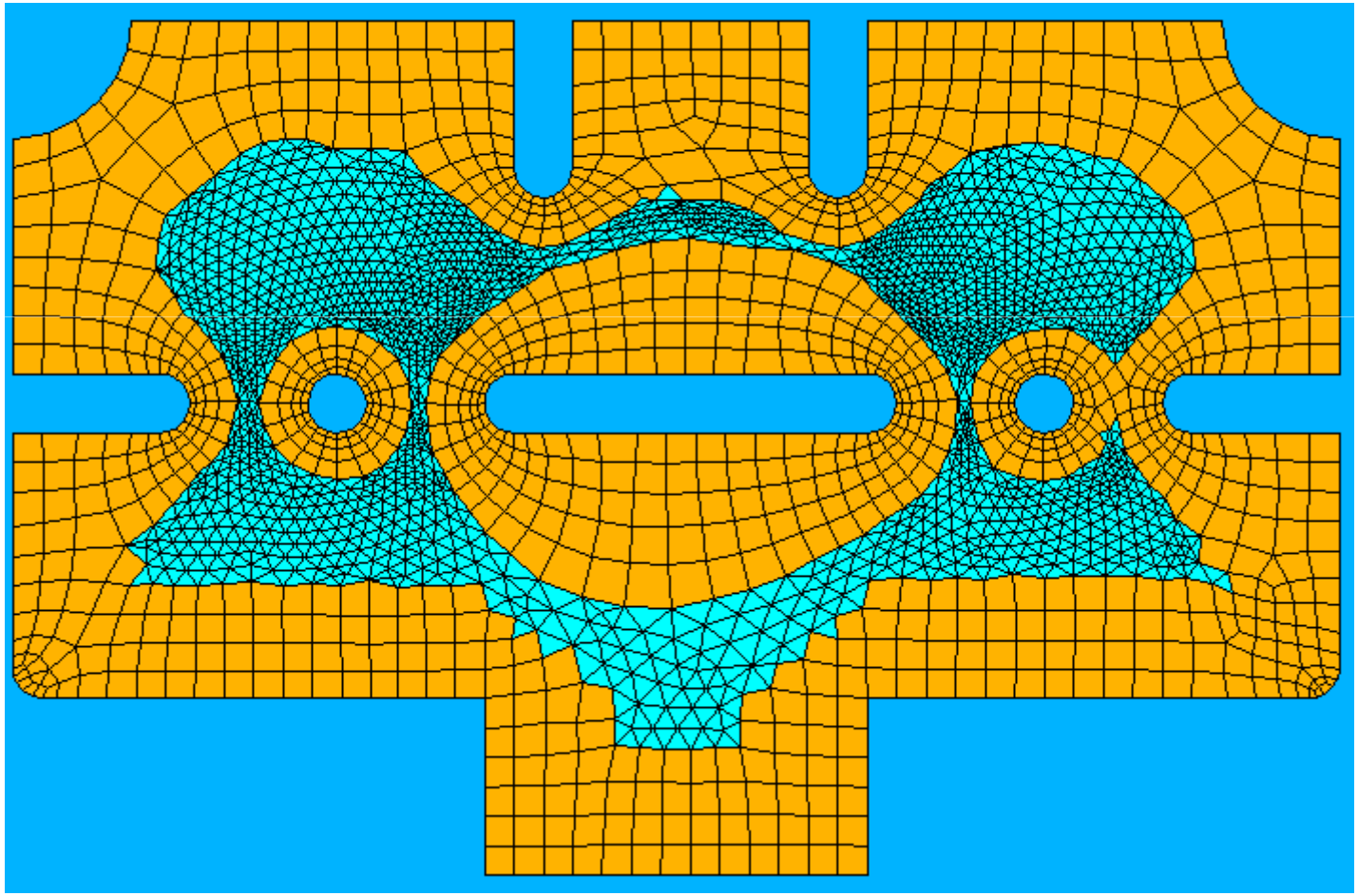
Unstructured: Indirect Quad

Q-Morph



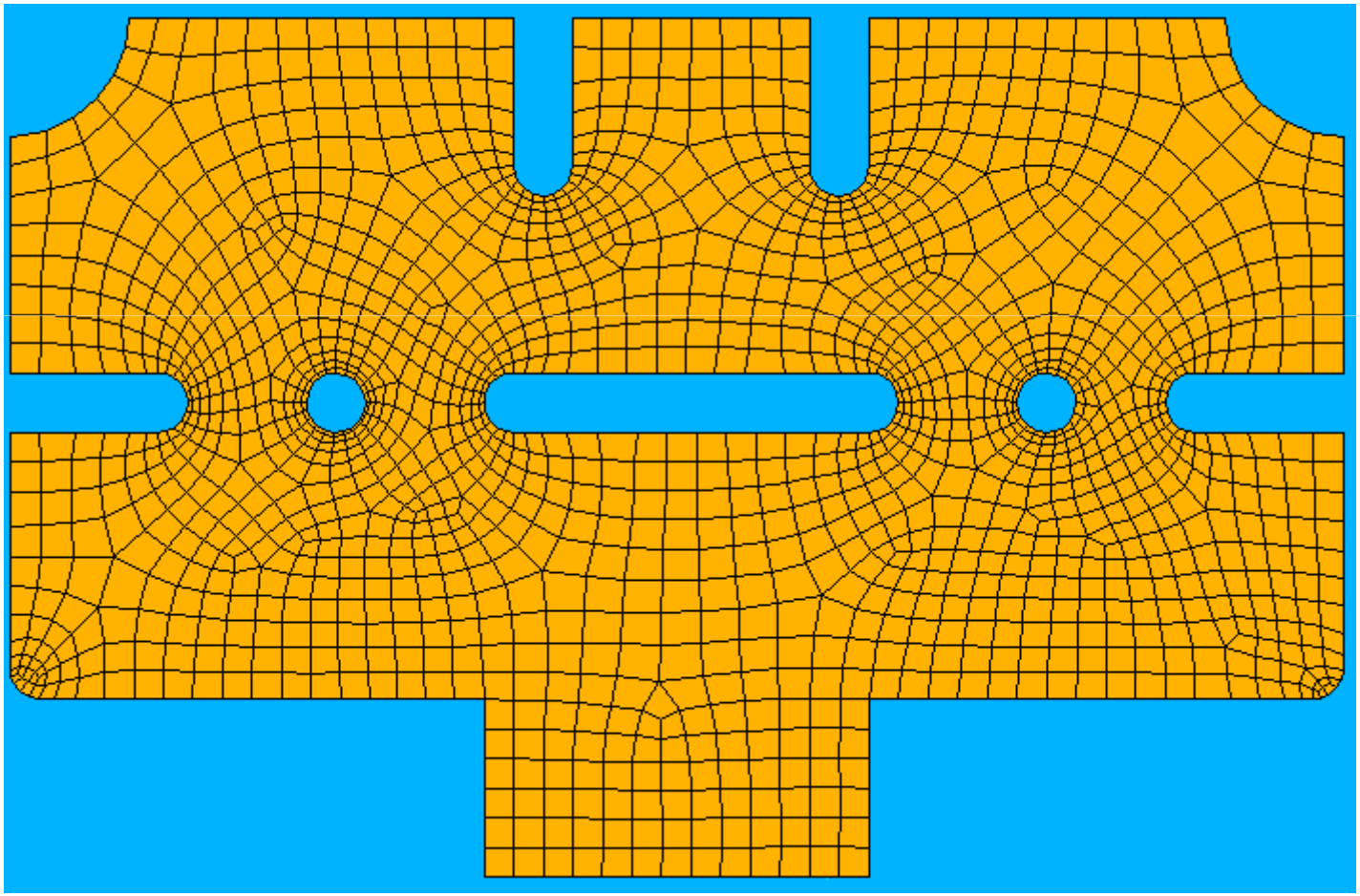
Unstructured: Indirect Quad

Q-Morph

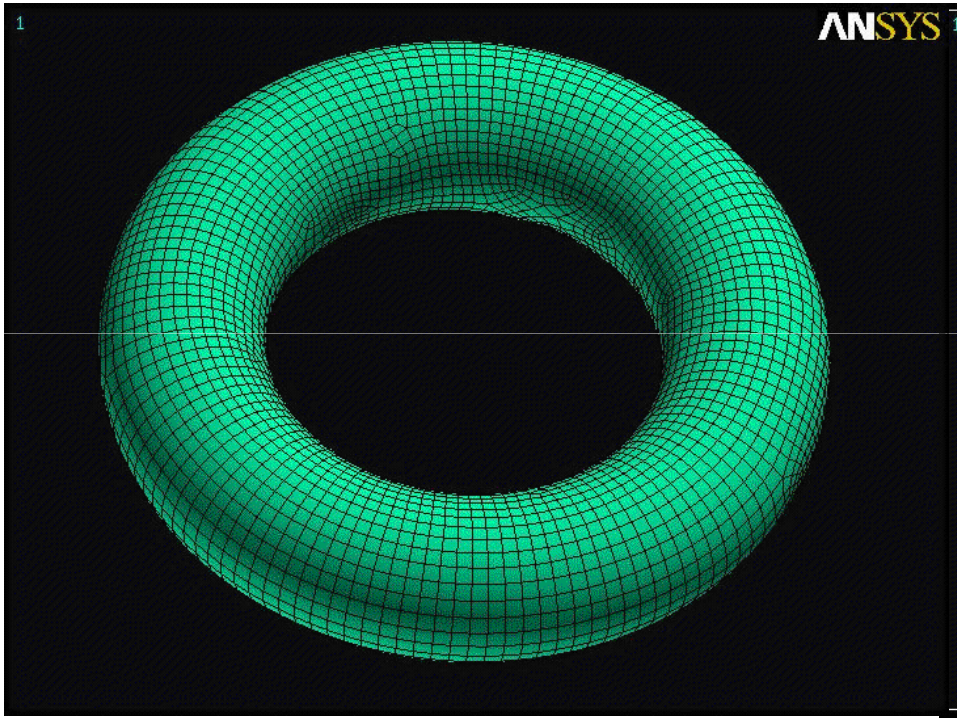


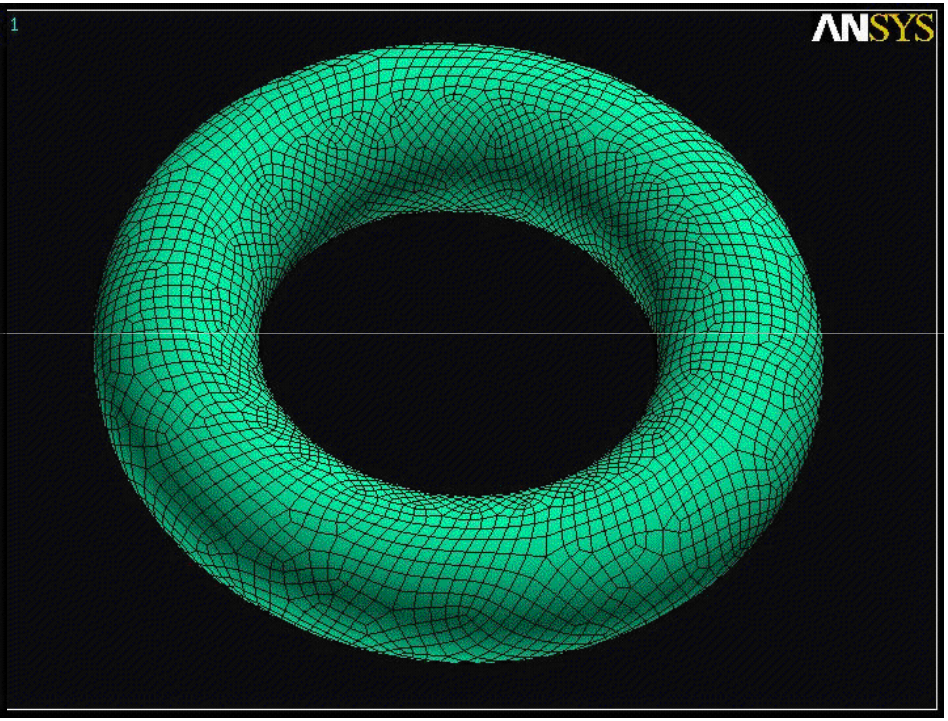
Unstructured: Indirect Quad

Q-Morph



Unstructured: Indirect Quad

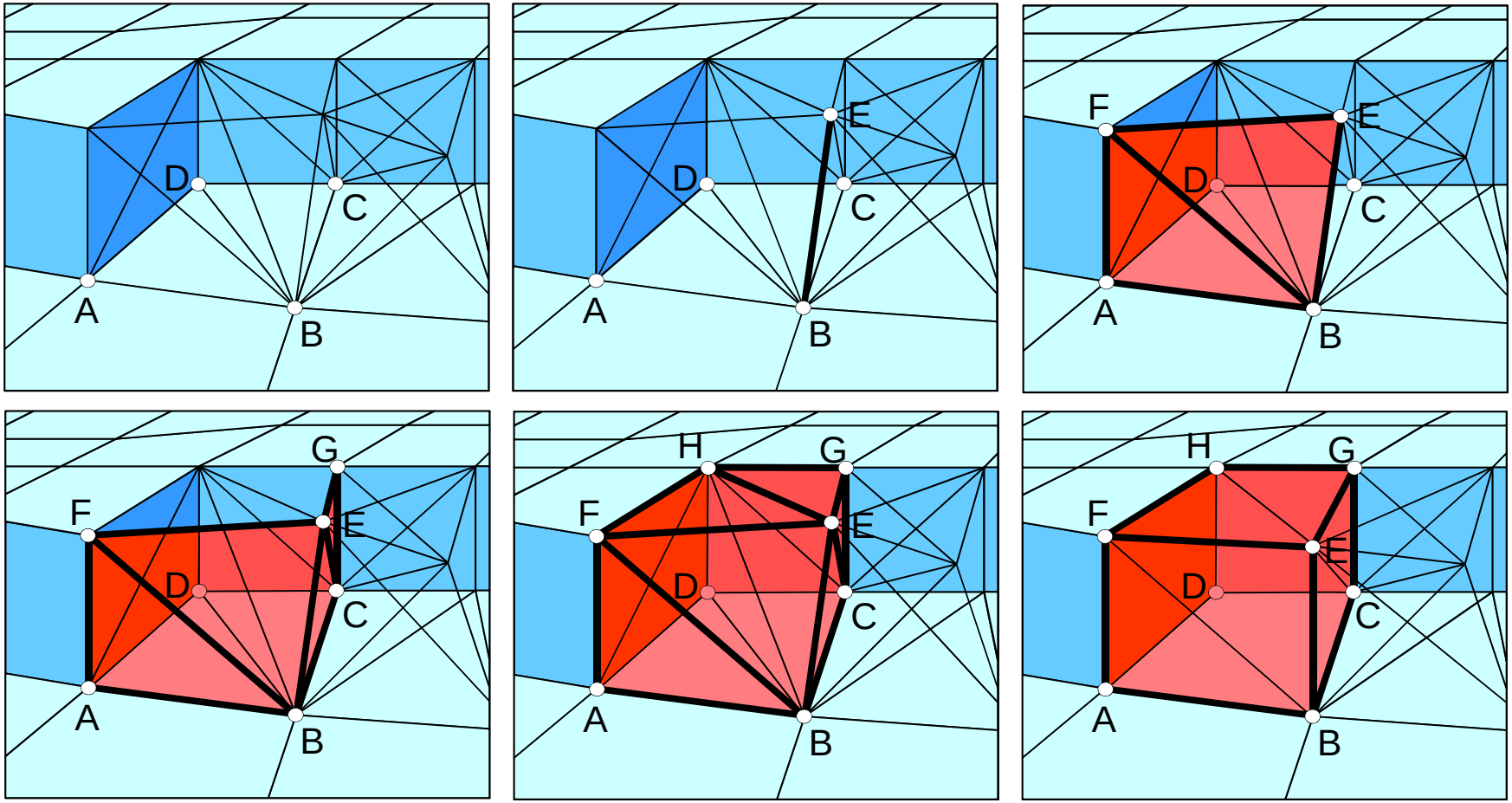




Q-Morph

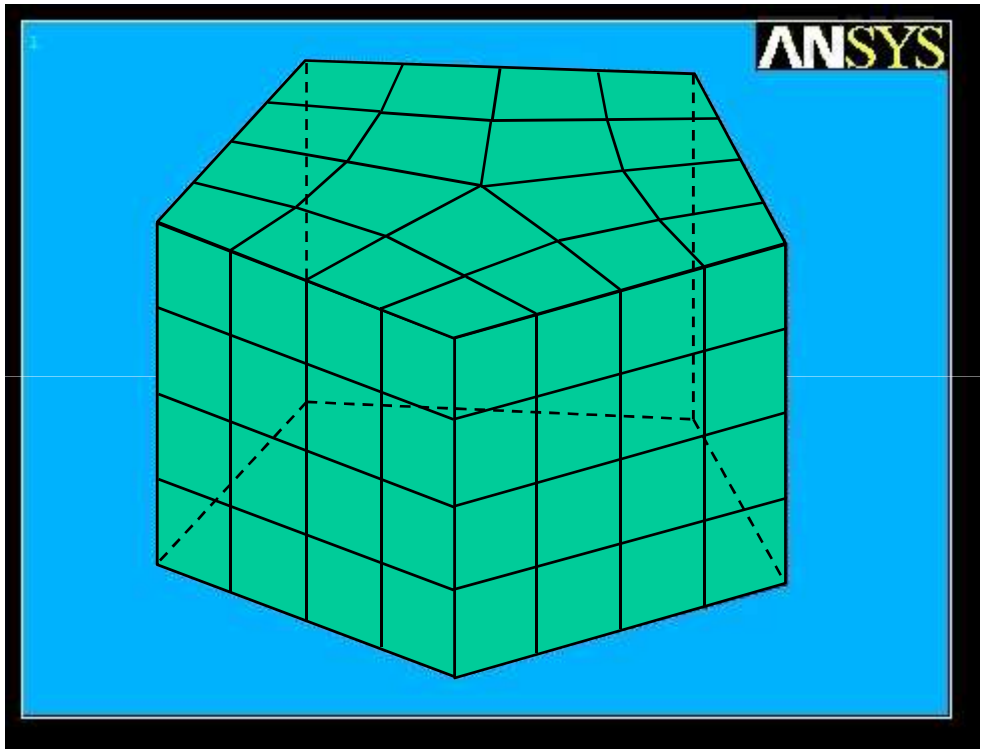
Lee,Lo Method

Indirect



Tetrahedral Merge w/ local transformations (“H-Morph”)

Unstructured-Hex

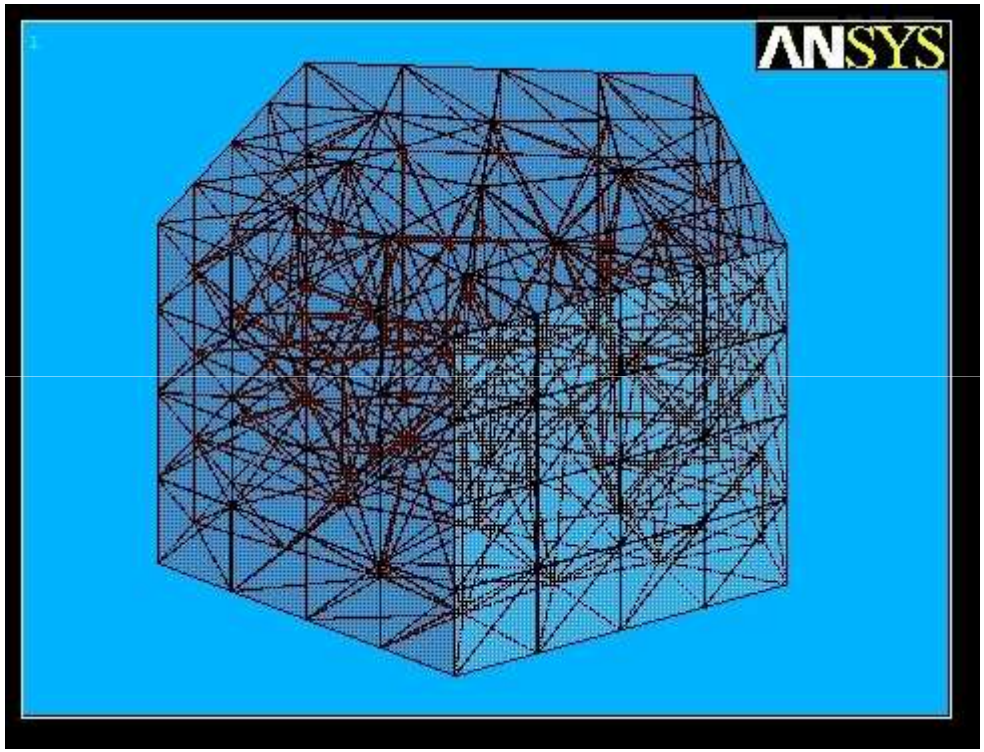


H-Morph

“Hex-Dominant Meshing”

(Owen and Saigal, 00)

Unstructured-Hex

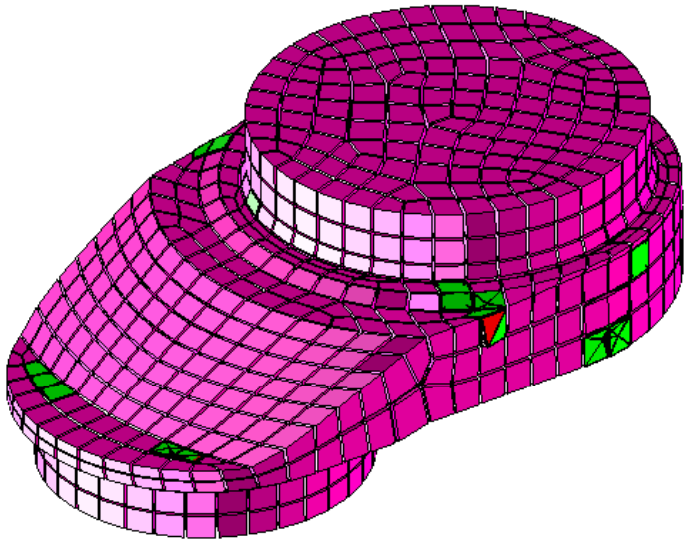


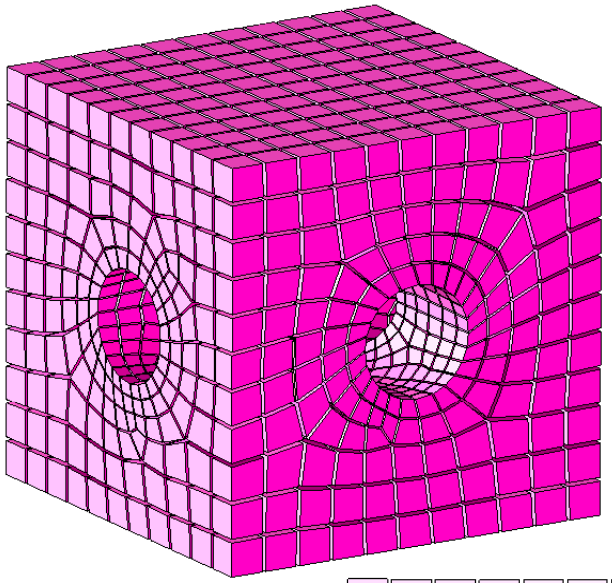
H-Morph

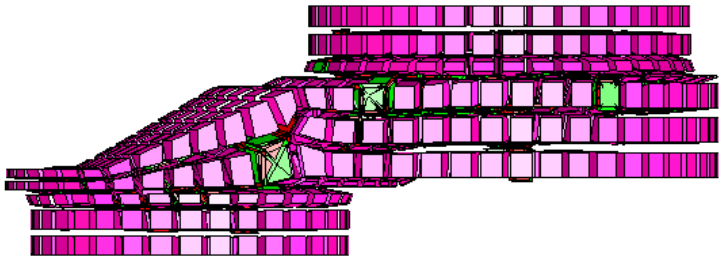
“Hex-Dominant Meshing”

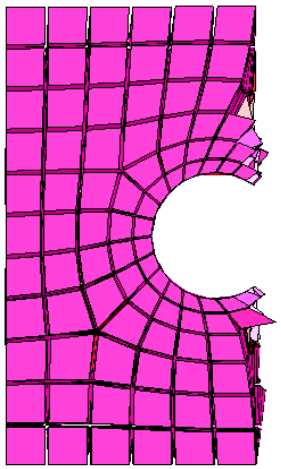
(Owen and Saigal, 00)

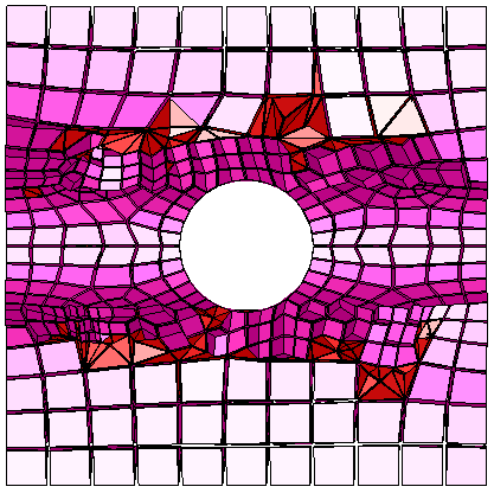
Unstructured-Hex



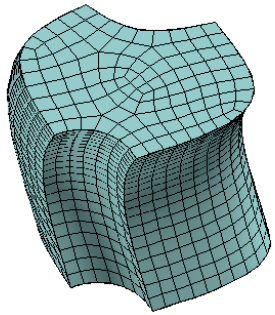


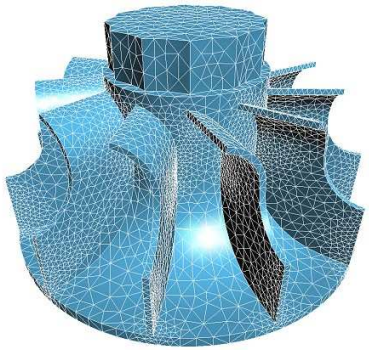


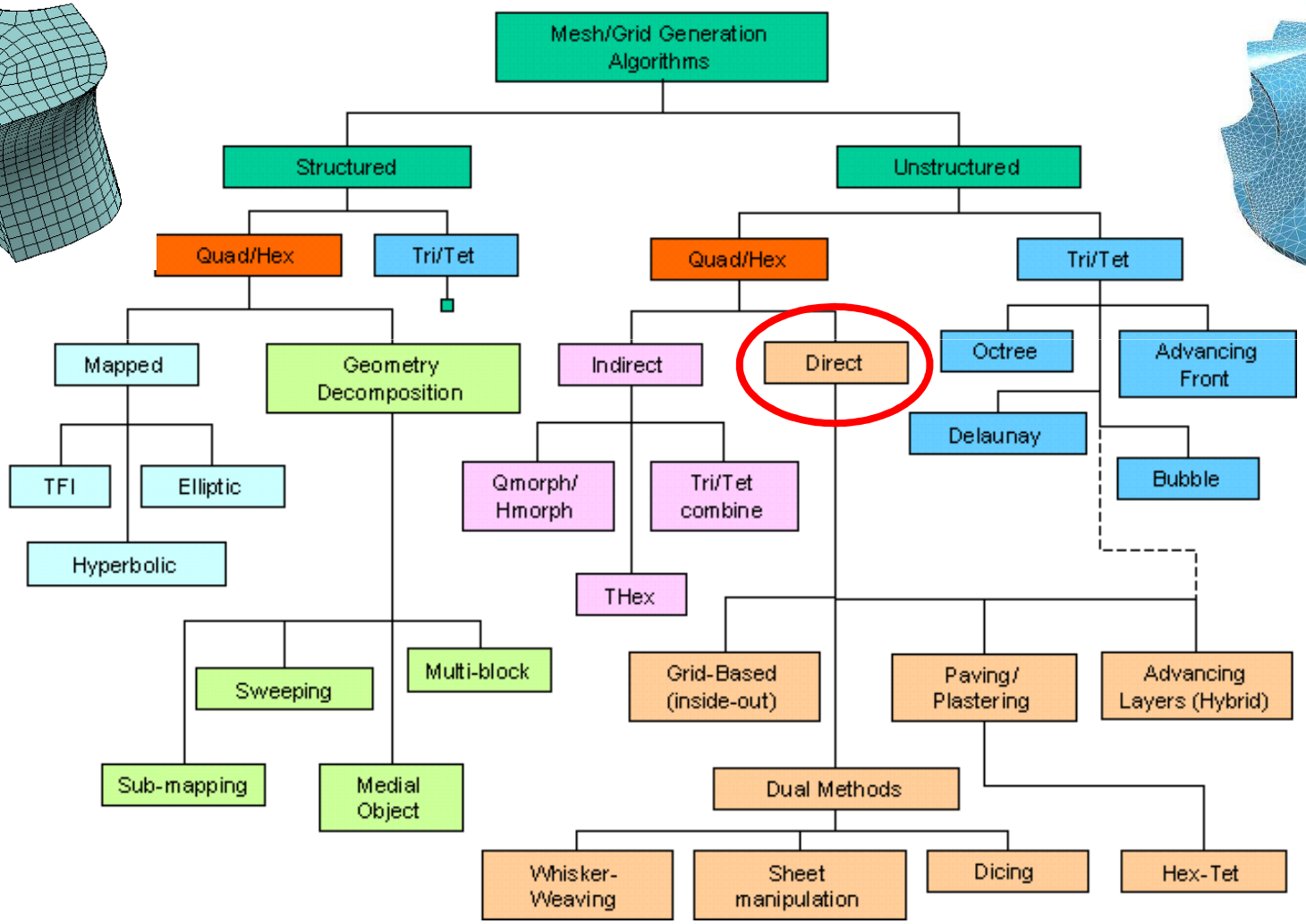




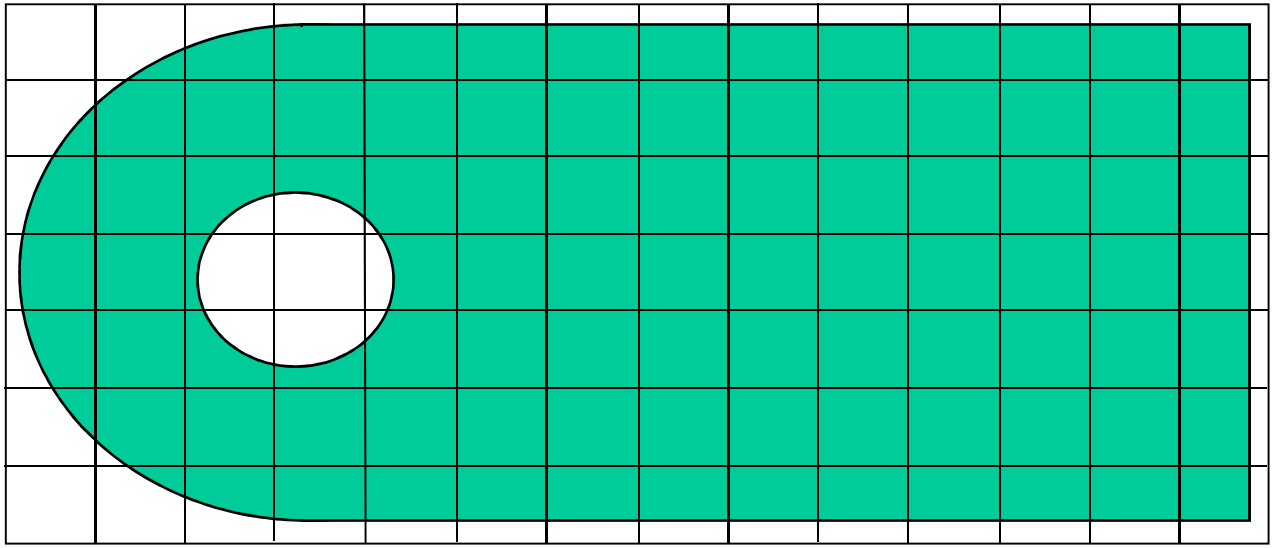
Meshing Algorithms







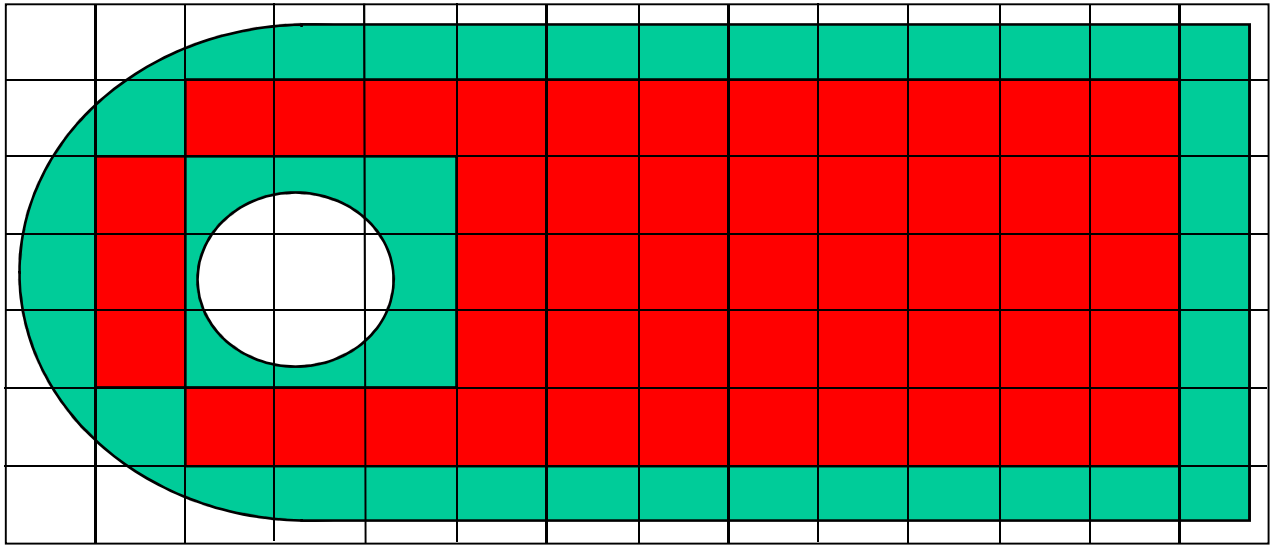
Unstructured-Hex



Grid-Based

- Generate regular grid of quads/hexes on the interior of model
- Fit elements to the boundary by projecting interior faces towards the surfaces
- Lower quality elements near boundary
- Non-boundary conforming

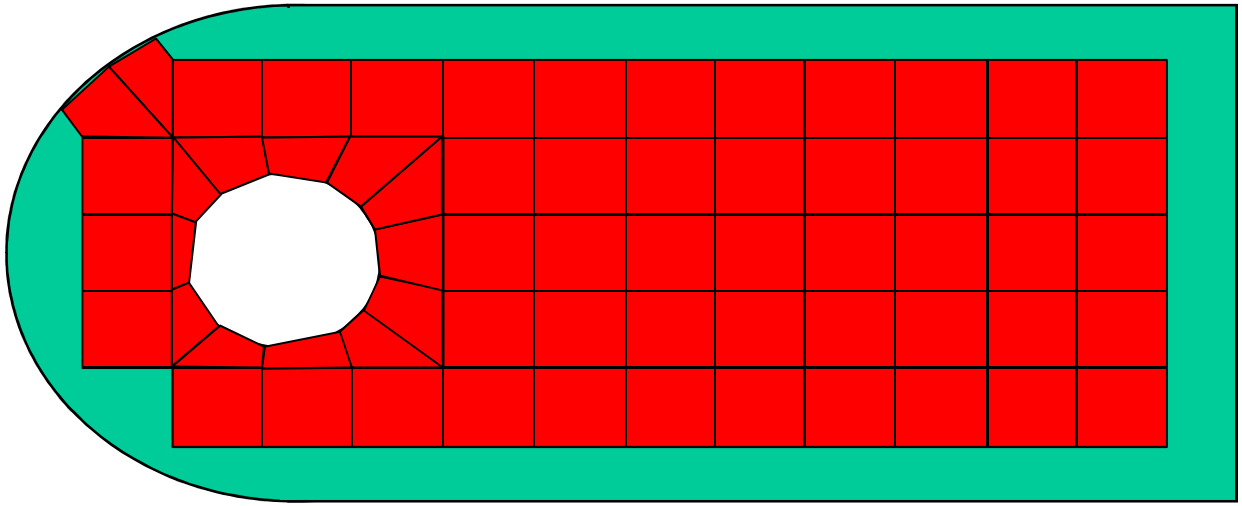
Unstructured-Hex



Grid-Based

- Generate regular grid of quads/hexes on the interior of model
- Fit elements to the boundary by projecting interior faces towards the surfaces
- Lower quality elements near boundary
- Non-boundary conforming

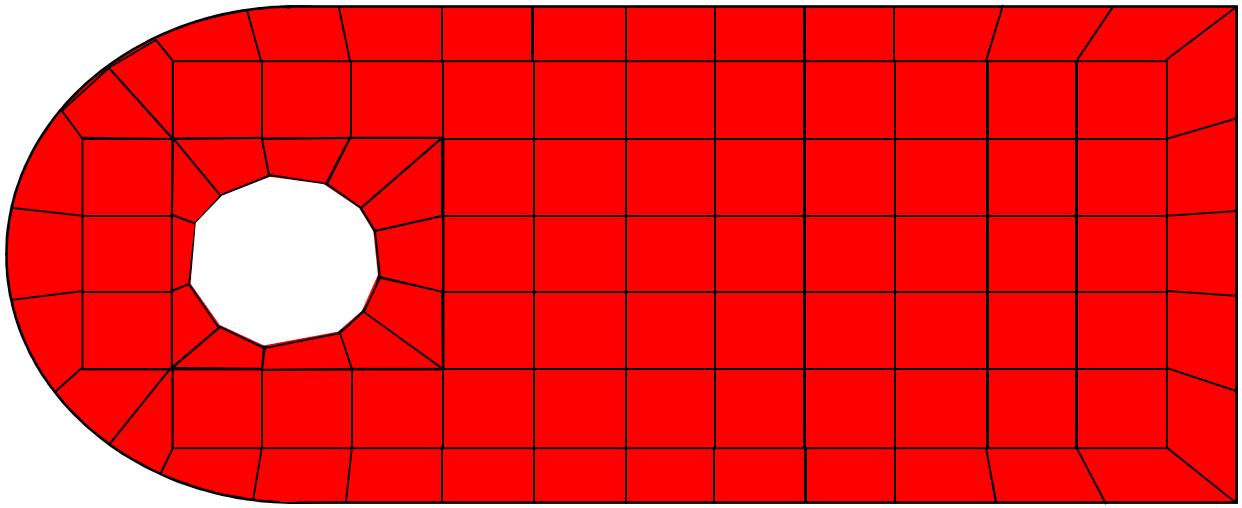
Unstructured-Hex



Grid-Based

- Generate regular grid of quads/hexes on the interior of model
- Fit elements to the boundary by projecting interior faces towards the surfaces
- Lower quality elements near boundary
- Non-boundary conforming

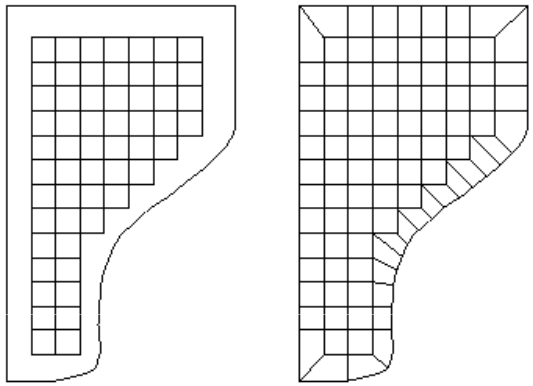
Unstructured-Hex



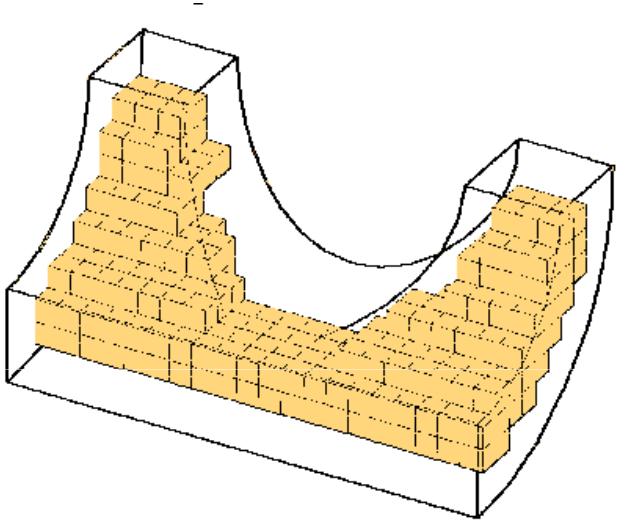
Grid-Based

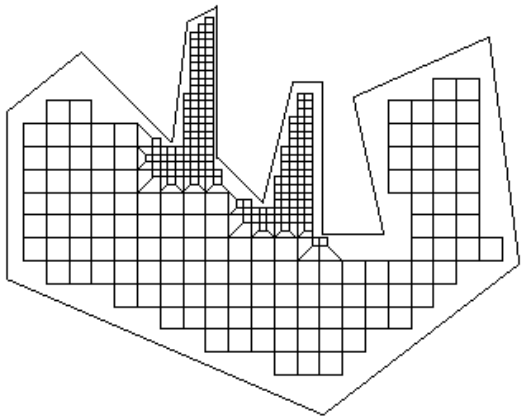
- Generate regular grid of quads/hexes on the interior of model
- Fit elements to the boundary by projecting interior faces towards the surfaces
- Lower quality elements near boundary
- Non-boundary conforming

Unstructured-Hex

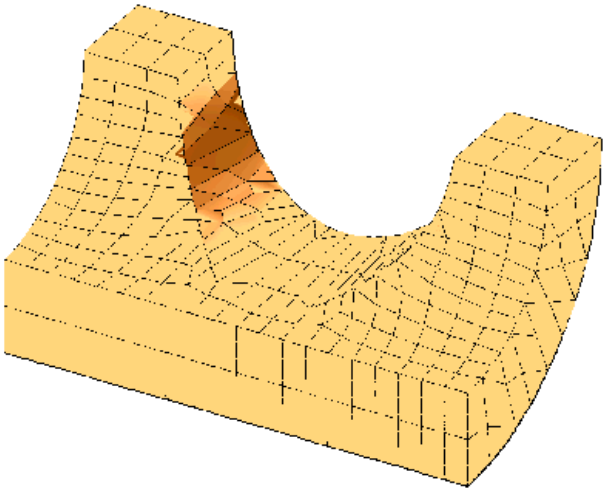


(Schneiders,96)

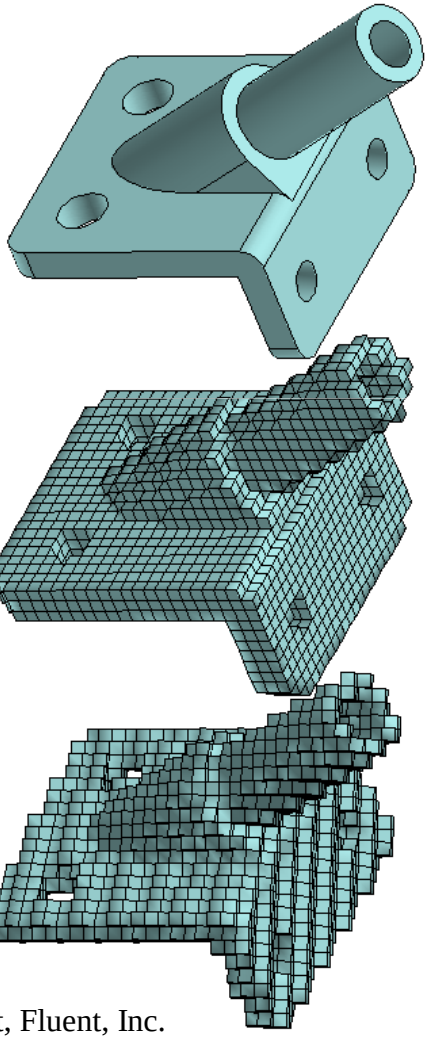




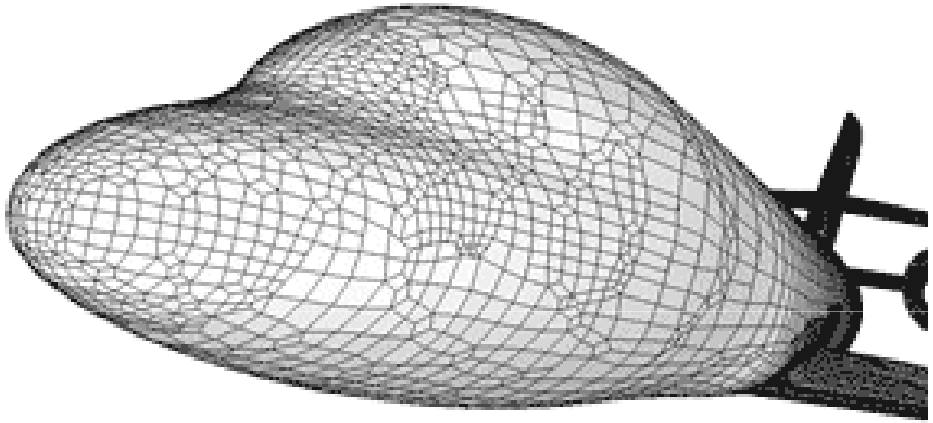
Grid-Based



Unstructured-Hex



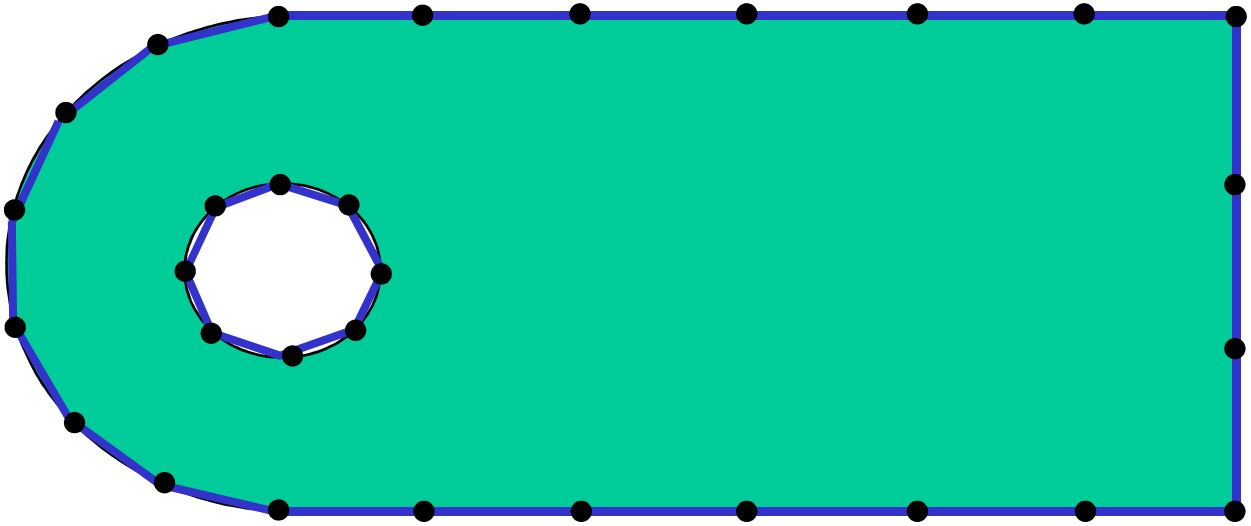
Gambit, Fluent, Inc.



http://www.numeca.be/hexpress_home.html

Grid-Based

Direct Quad

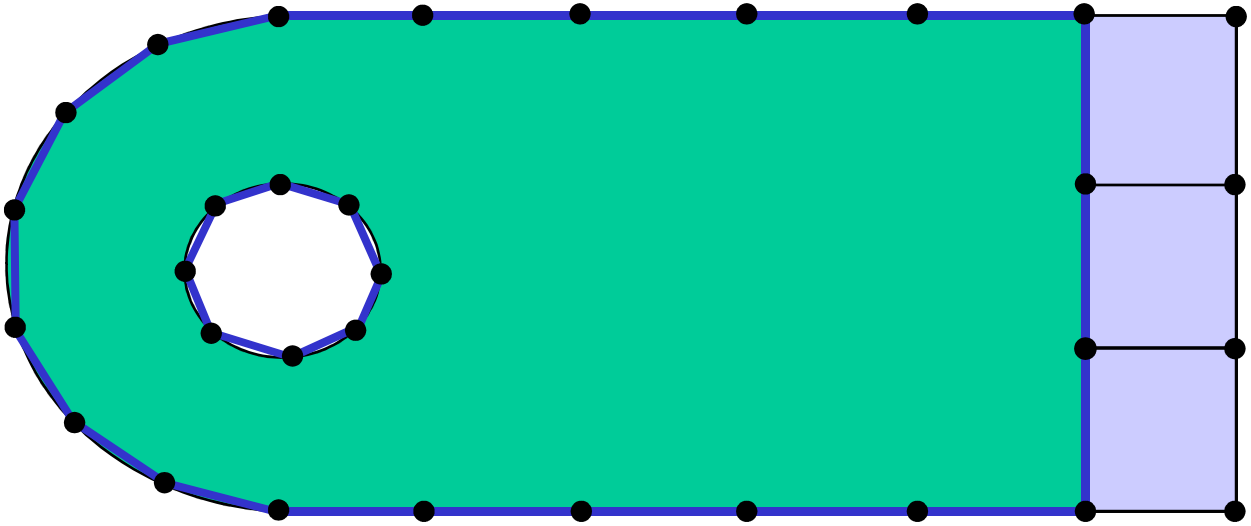


Paving

- Advancing Front: Begins with front at boundary
- Forms rows of elements based on front angles
- Must have even number of intervals for all-quad mesh

(Blacker,92)(Cass,96)

Unstructured-Quad

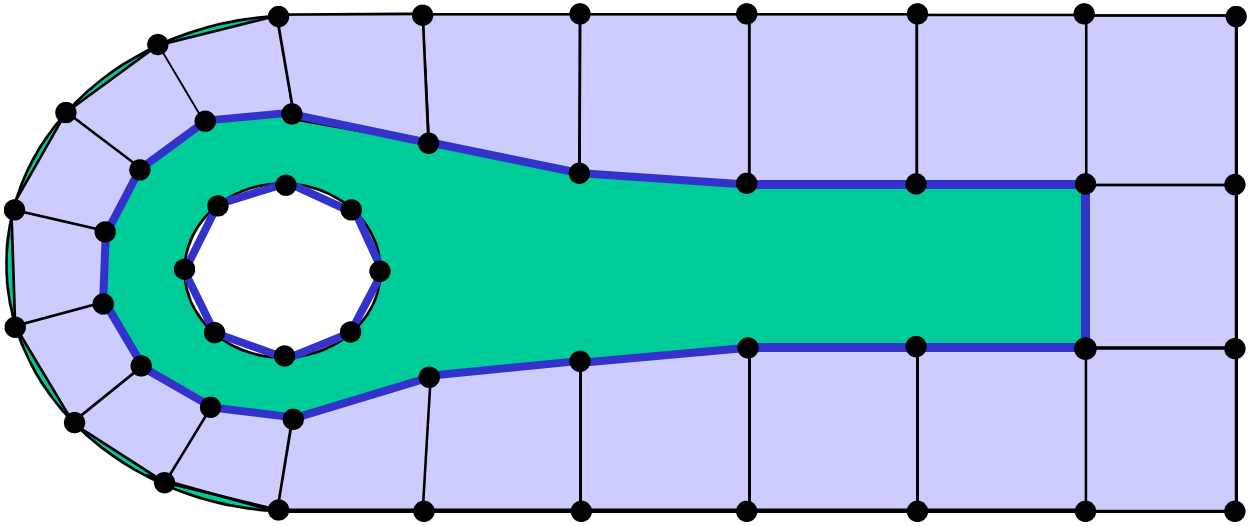


Paving

- Advancing Front: Begins with front at boundary
- Forms rows of elements based on front angles
- Must have even number of intervals for all-quad mesh

(Blacker,92)(Cass,96)

Unstructured-Quad

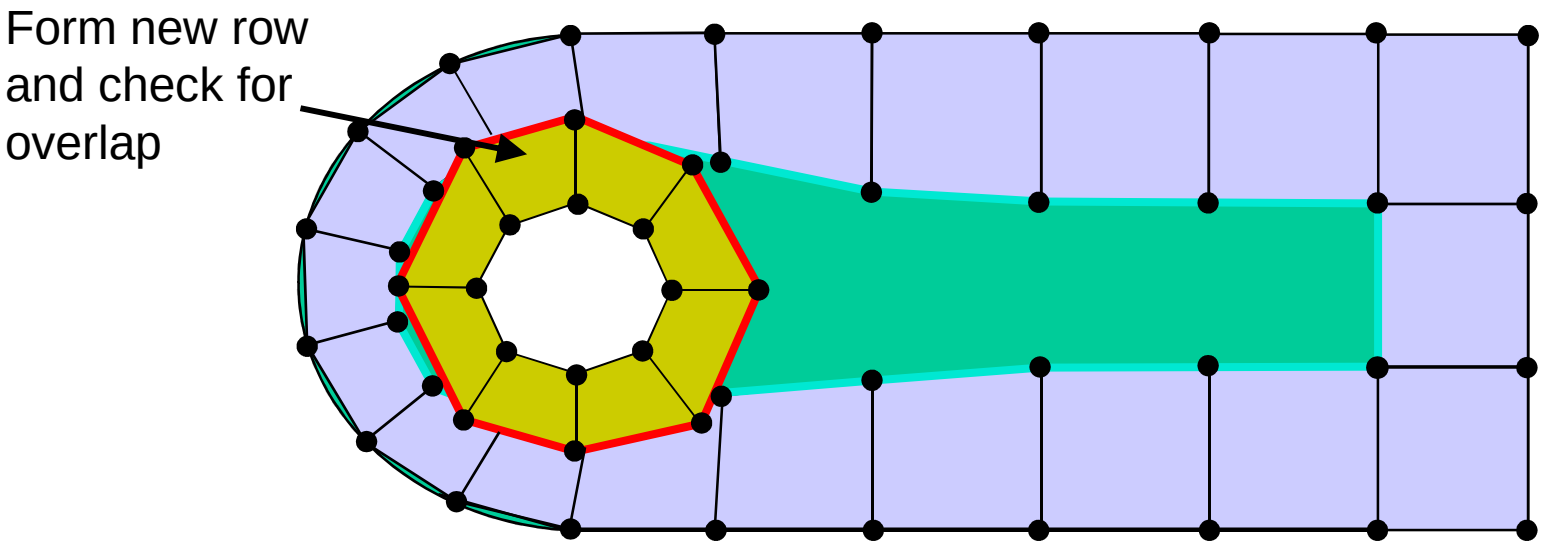


Paving

- Advancing Front: Begins with front at boundary
- Forms rows of elements based on front angles
- Must have even number of intervals for all-quad mesh

(Blacker,92)(Cass,96)

Unstructured-Quad

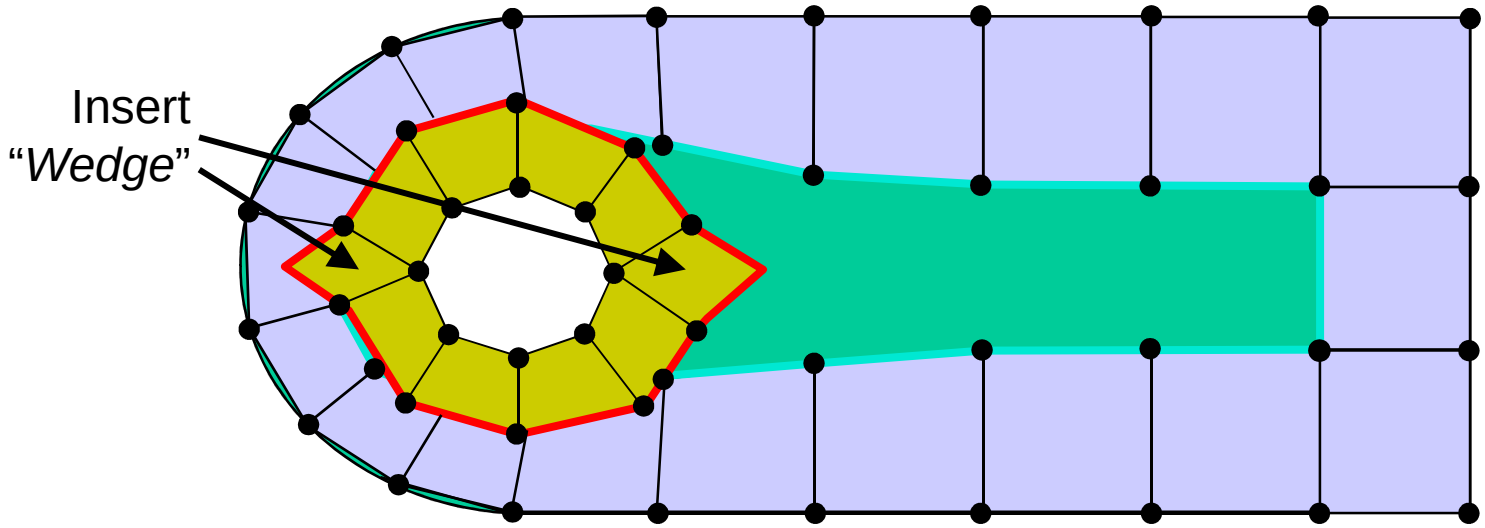


Paving

- Advancing Front: Begins with front at boundary
- Forms rows of elements based on front angles
- Must have even number of intervals for all-quad mesh

(Blacker,92)(Cass,96)

Unstructured-Quad

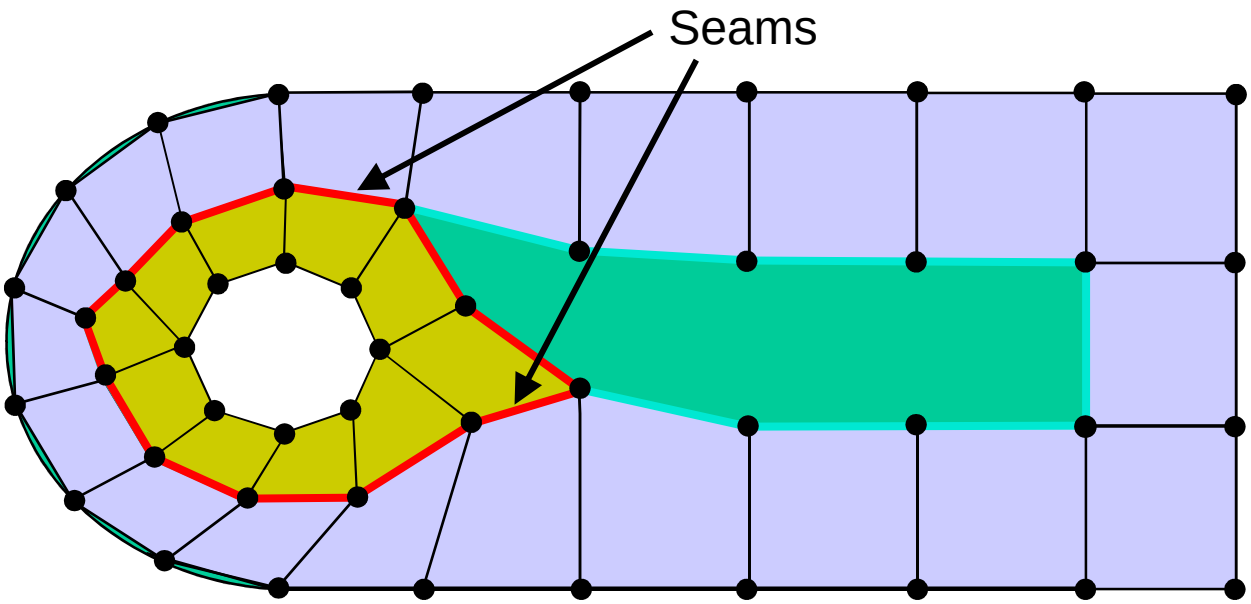


Paving

- Advancing Front: Begins with front at boundary
- Forms rows of elements based on front angles
- Must have even number of intervals for all-quad mesh

(Blacker,92)(Cass,96)

Unstructured-Quad



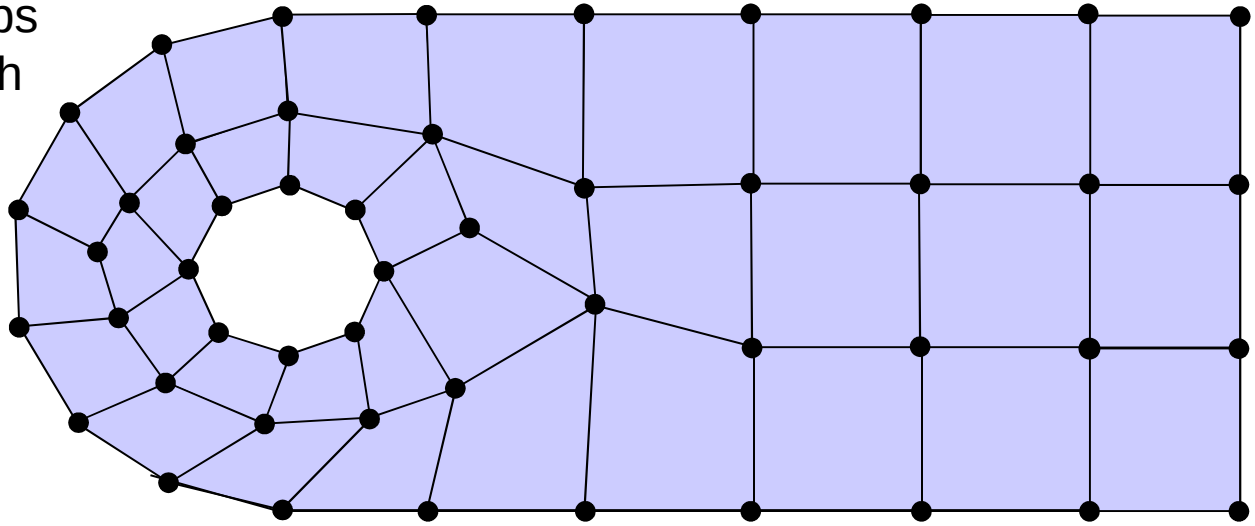
Paving

- Advancing Front: Begins with front at boundary
- Forms rows of elements based on front angles
- Must have even number of intervals for all-quad mesh

(Blacker,92)(Cass,96)

Unstructured-Quad

Close Loops
and smooth

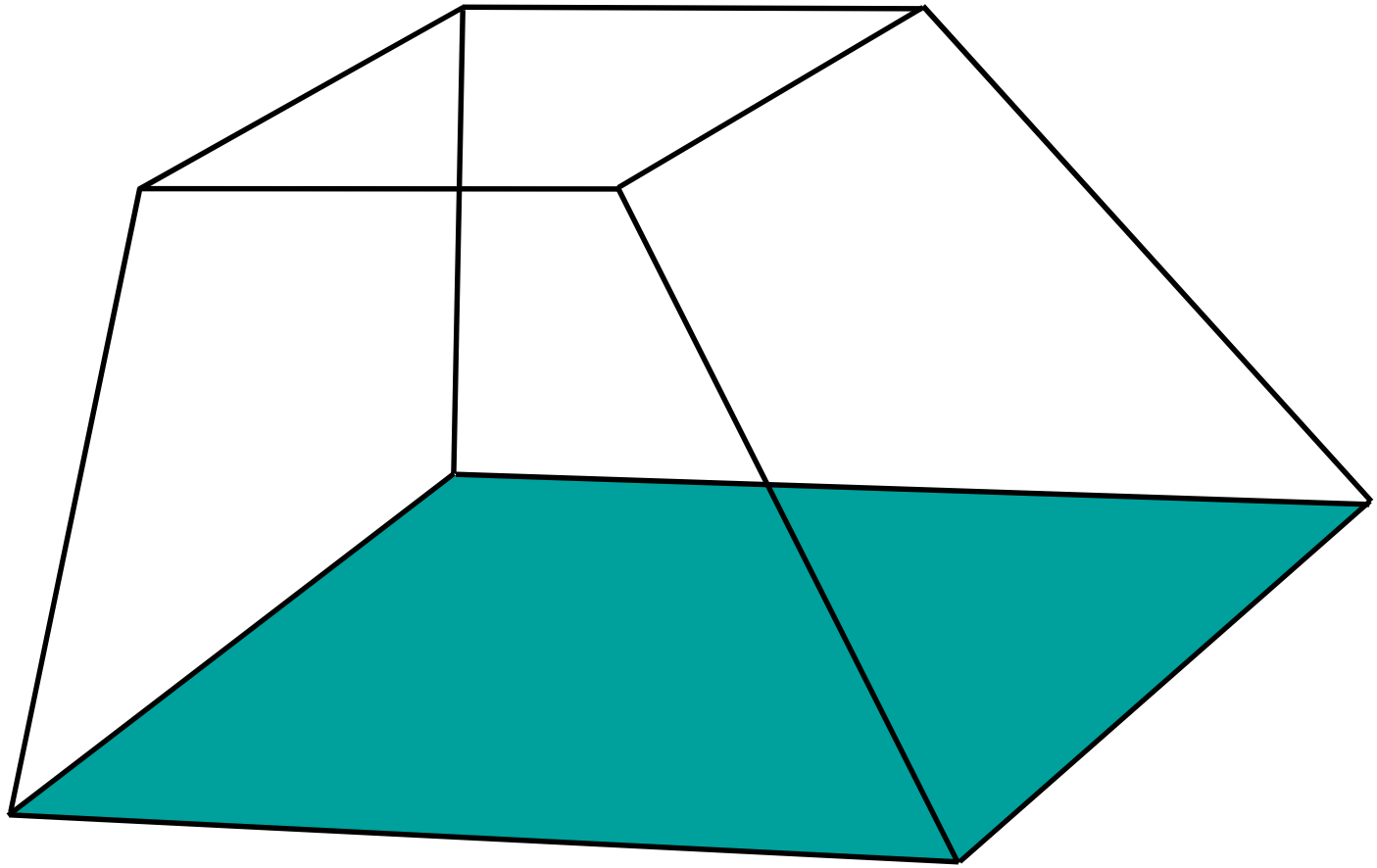


Paving

- Advancing Front: Begins with front at boundary
- Forms rows of elements based on front angles
- Must have even number of intervals for all-quad mesh

(Blacker,92)(Cass,96)

Unstructured-Hex

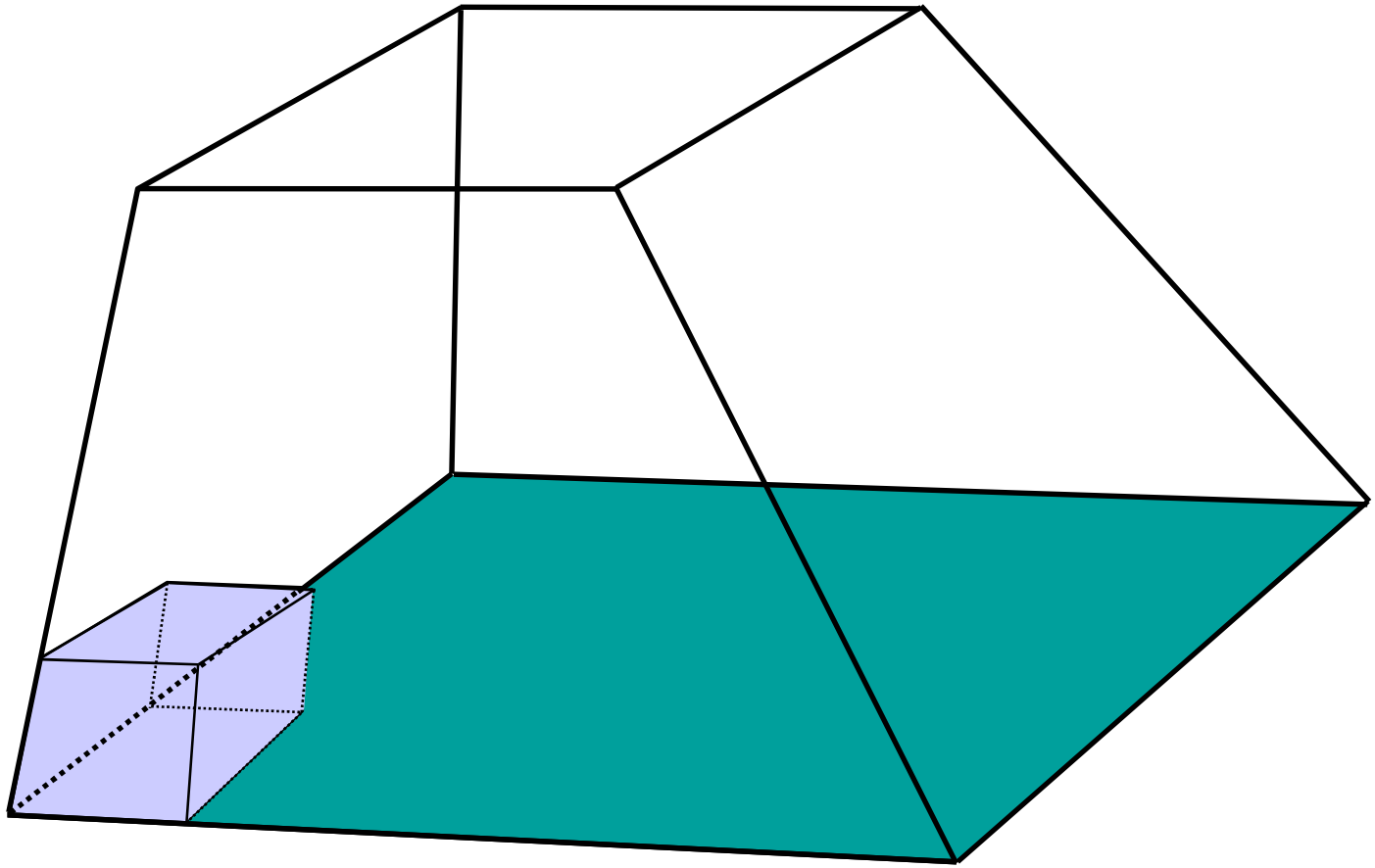


Plastering

- 3D extension of “paving”
- Row-by row or element-by-element

(Blacker, 93)

Unstructured-Hex

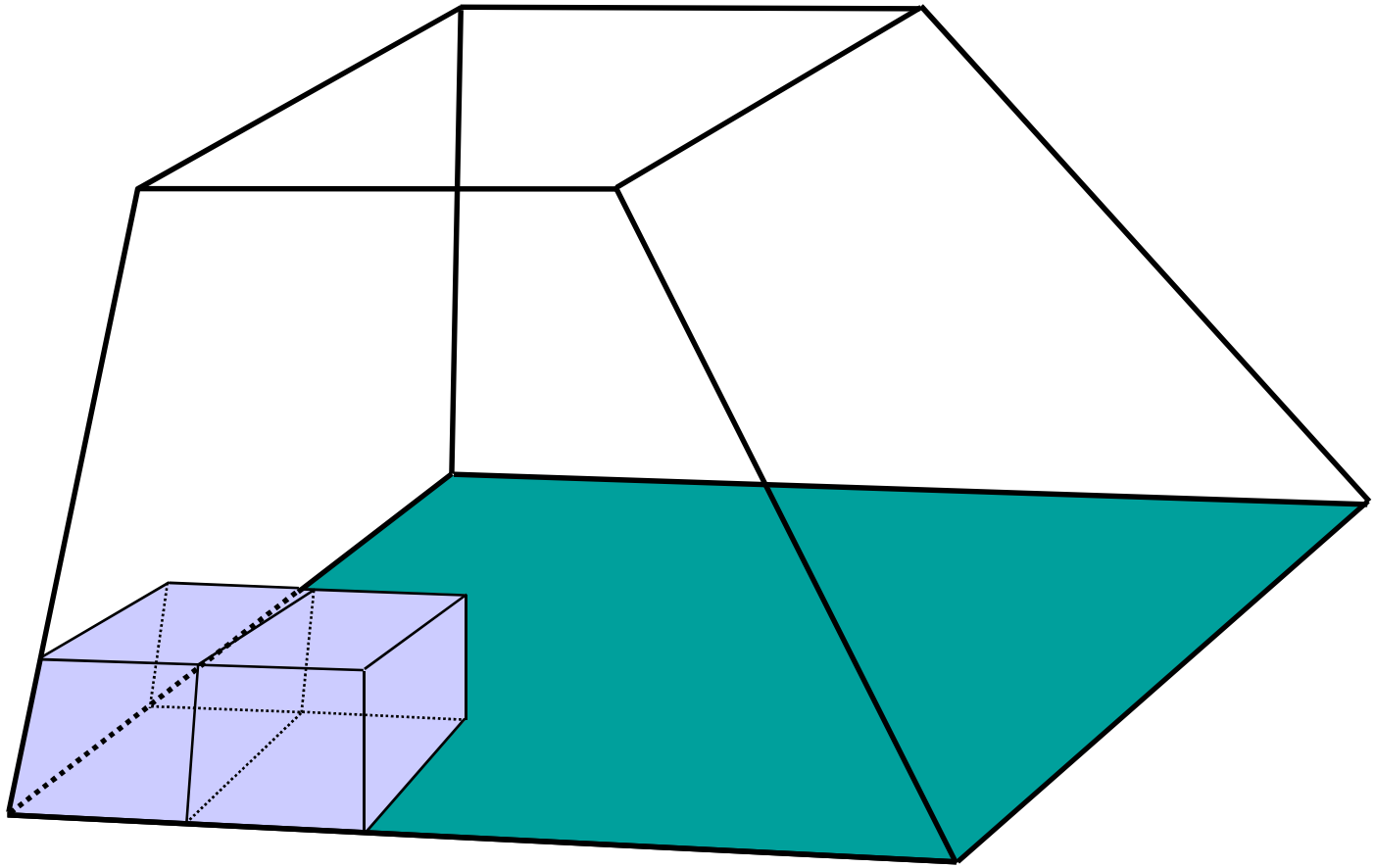


Plastering

- 3D extension of “paving”
- Row-by row or element-by-element

(Blacker, 93)

Unstructured-Hex

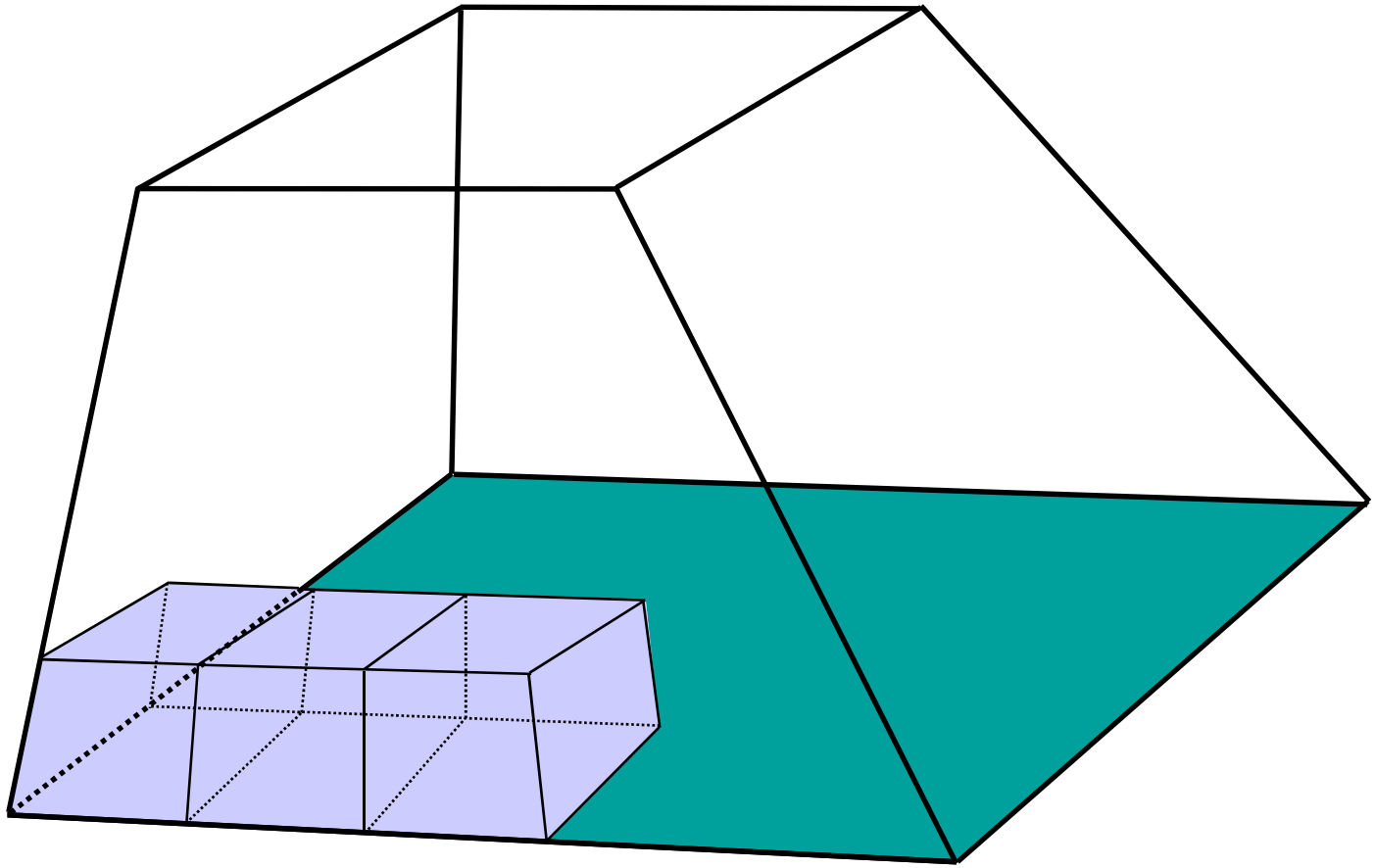


Plastering

- 3D extension of “paving”
- Row-by row or element-by-element

(Blacker, 93)

Unstructured-Hex

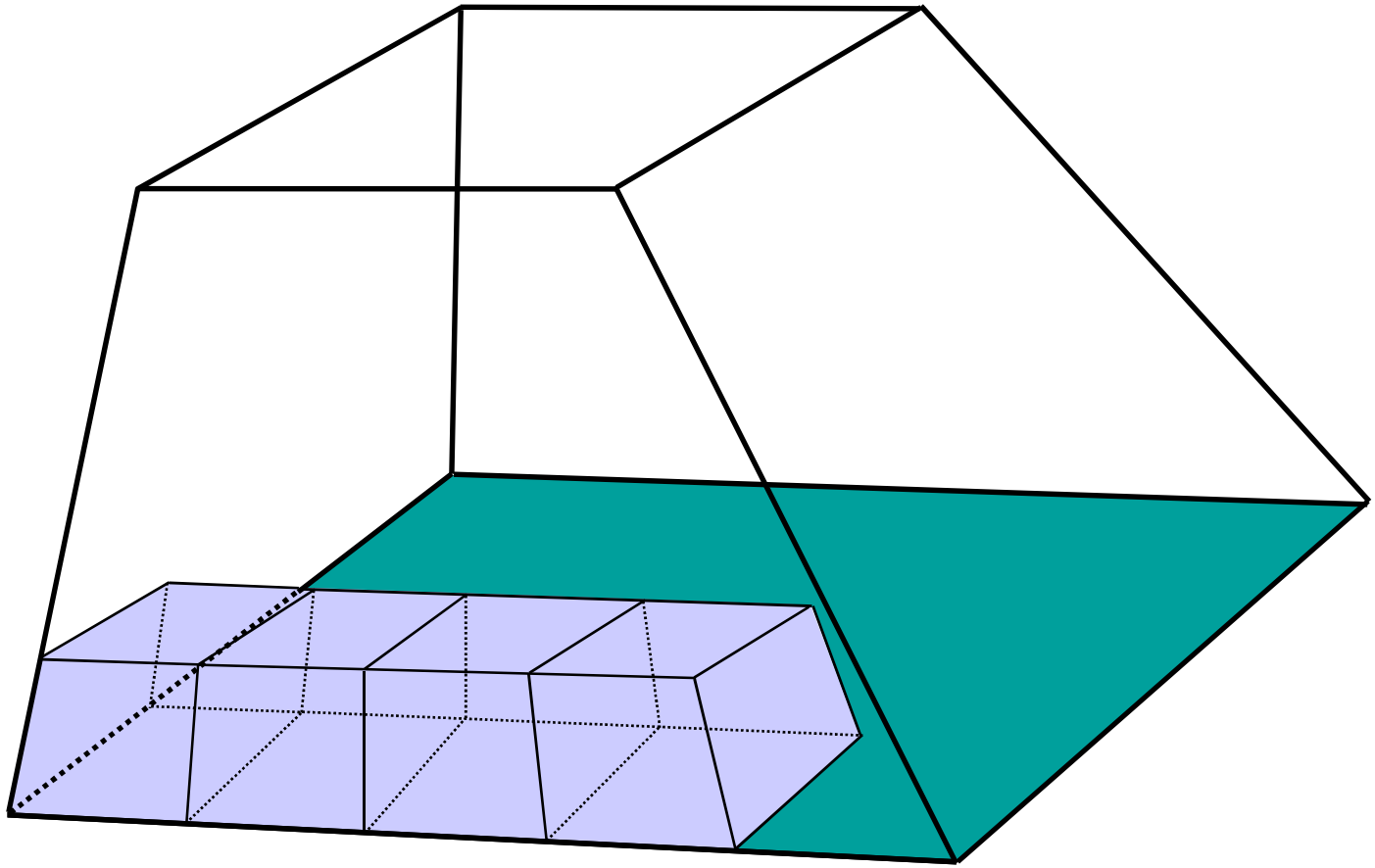


Plastering

- 3D extension of “paving”
- Row-by row or element-by-element

(Blacker, 93)

Unstructured-Hex

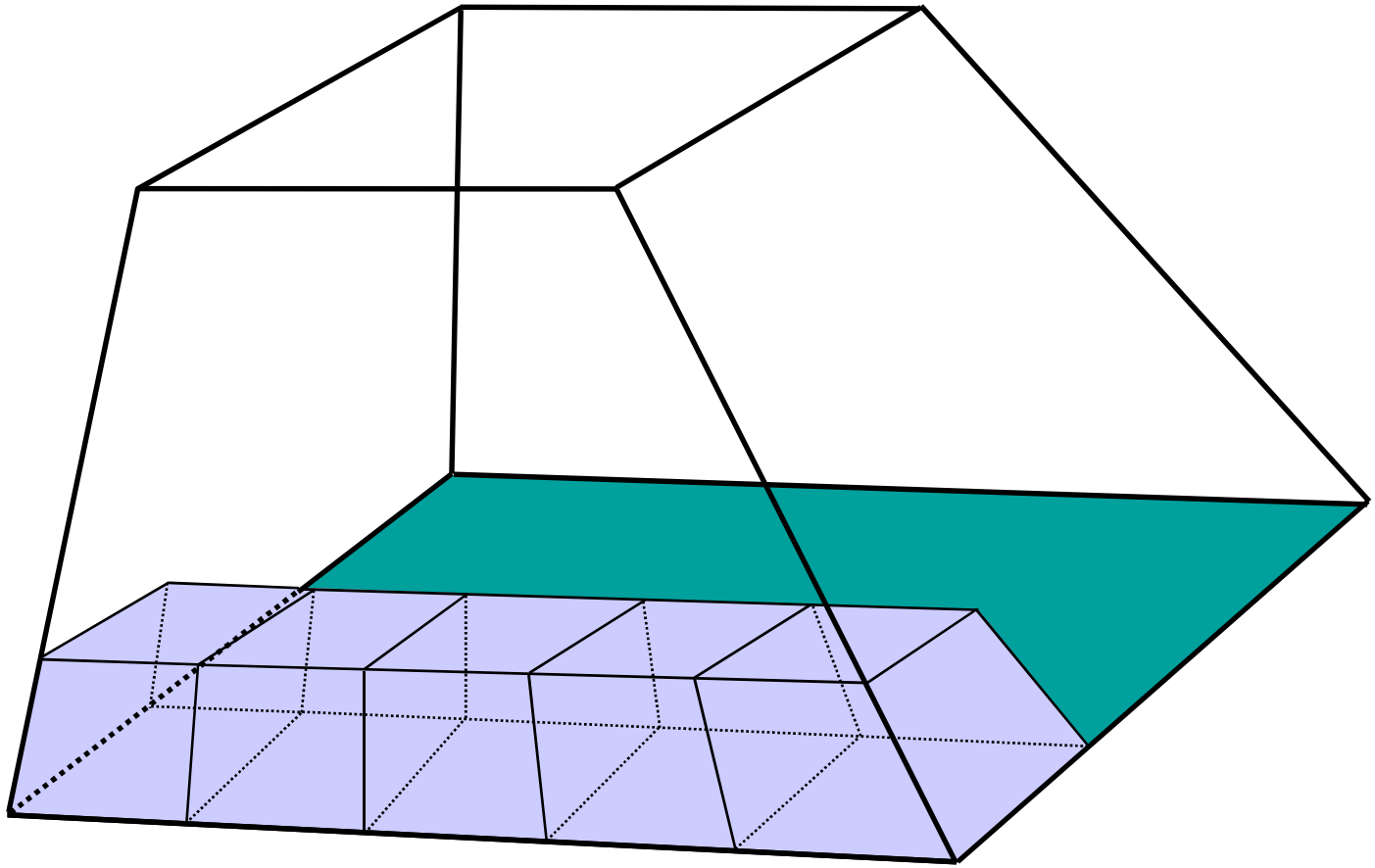


Plastering

- 3D extension of “paving”
- Row-by row or element-by-element

(Blacker, 93)

Unstructured-Hex

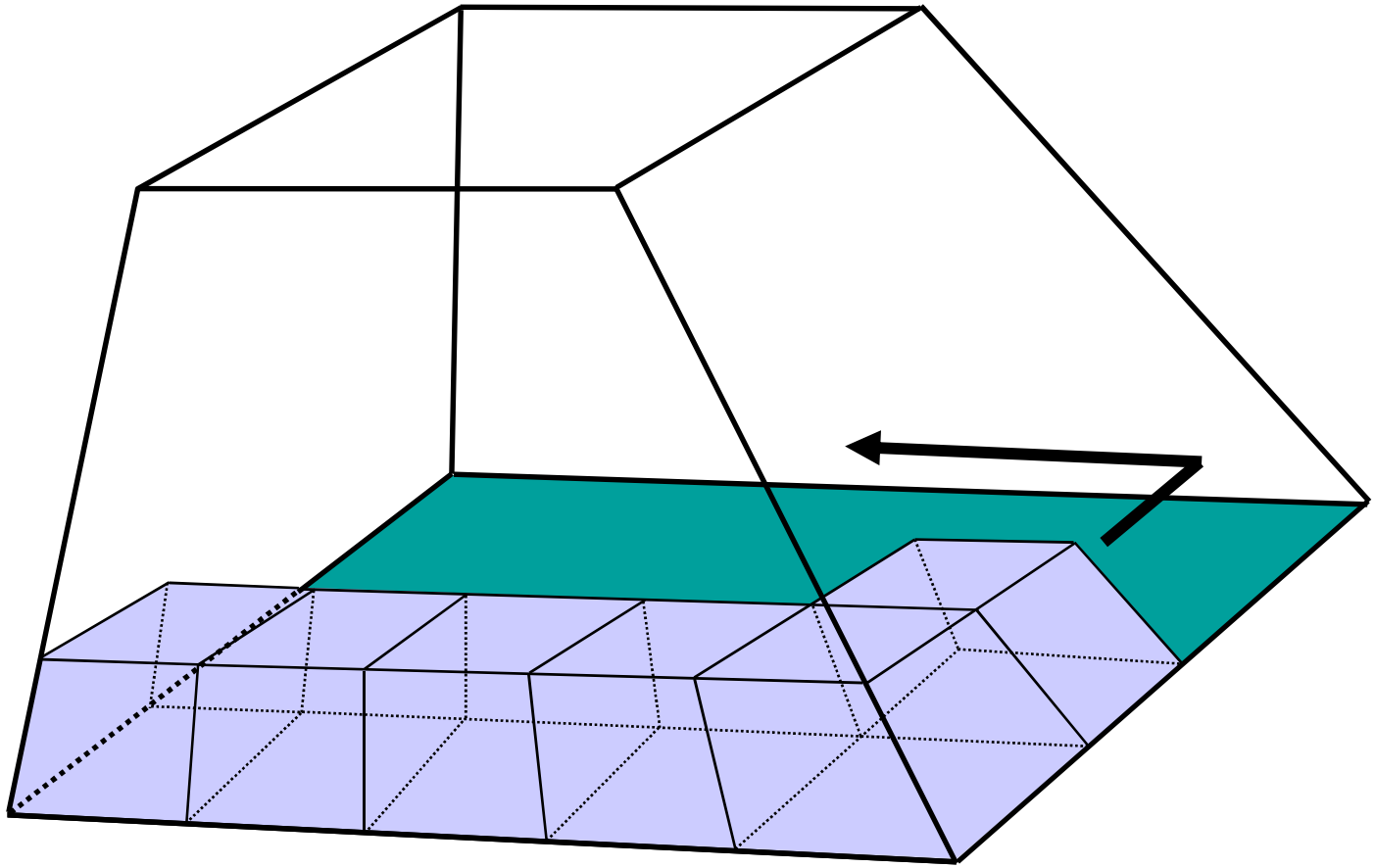


Plastering

- 3D extension of “paving”
- Row-by row or element-by-element

(Blacker, 93)

Unstructured-Hex

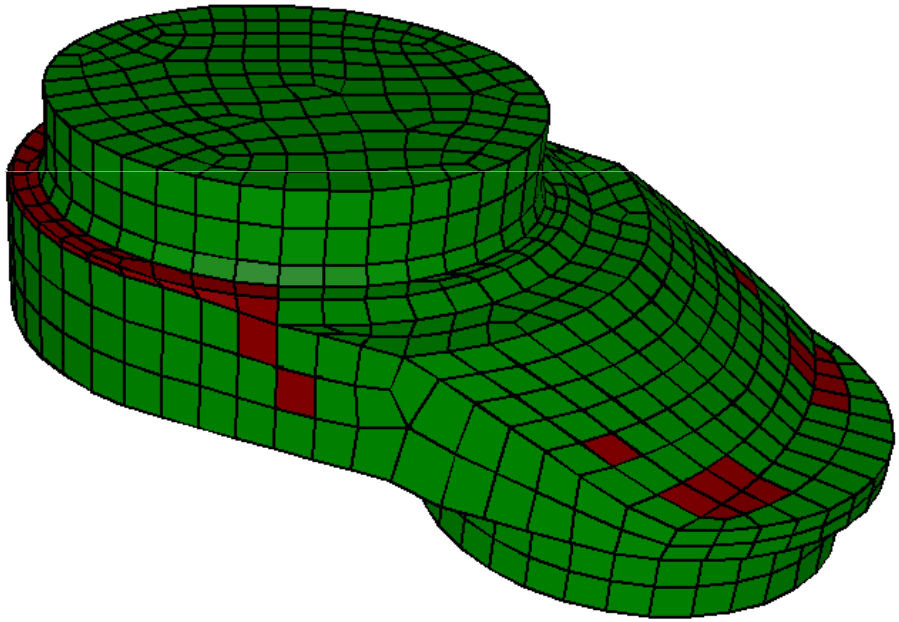


Plastering

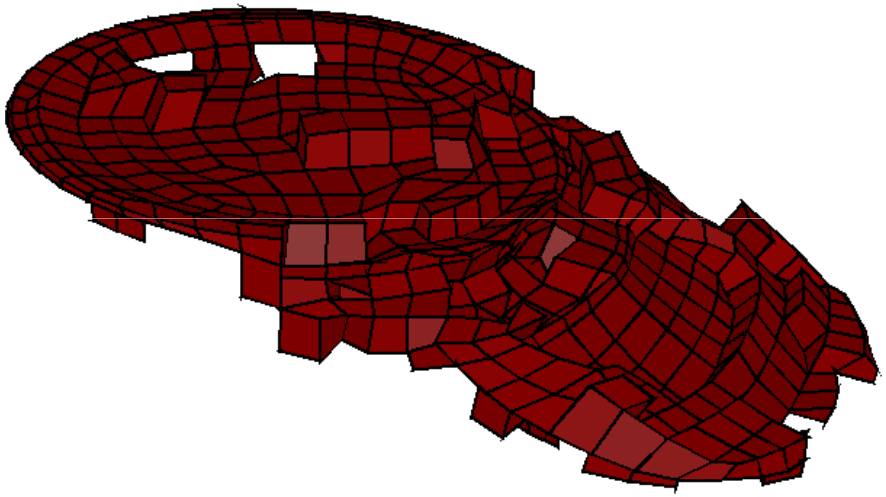
- 3D extension of “paving”
- Row-by row or element-by-element (Blacker, 93)

Unstructured-Hex

Exterior Hex mesh



Remaining Void

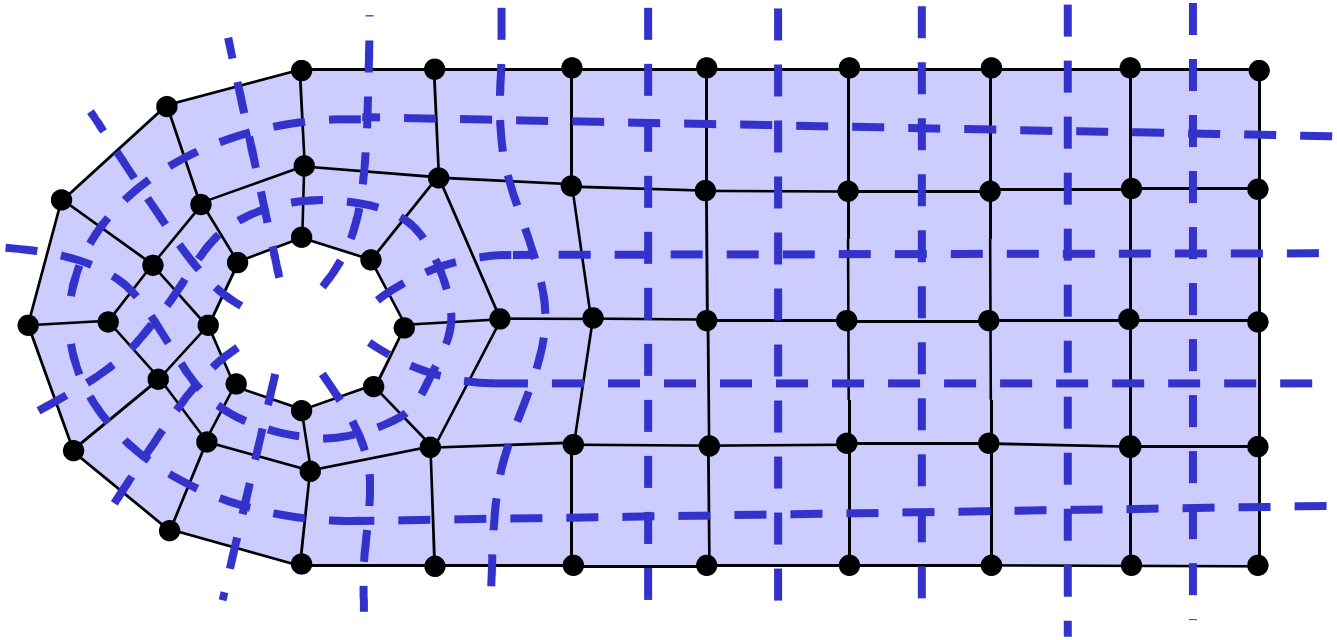


Ford Crankshaft

Plastering + Tet Meshing

“Hex-Dominant Meshing”

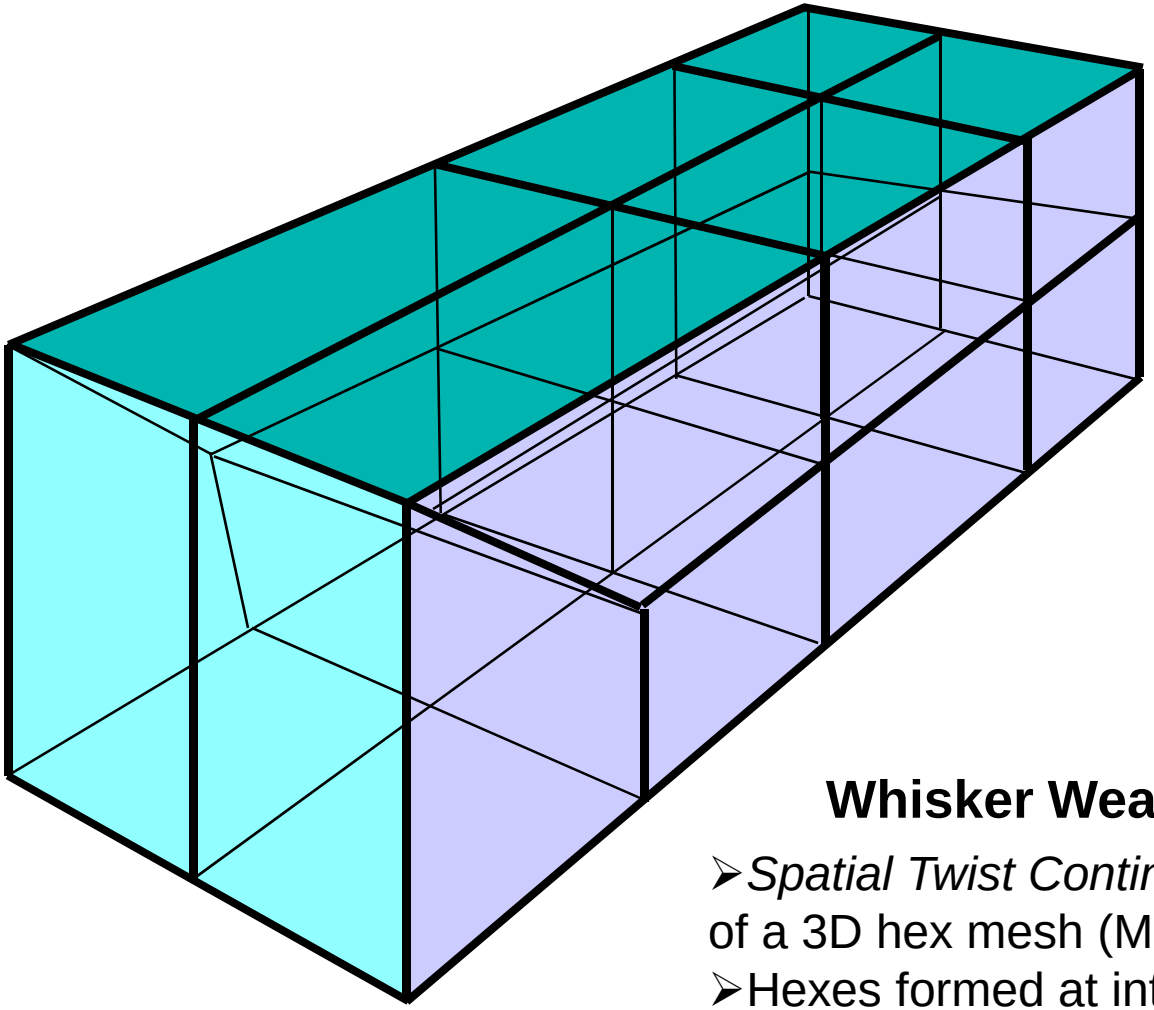
Direct



Whisker Weaving

- First constructs *dual* of the quad/hex mesh
- Inserts quad/hex at the intersections of the dual chords

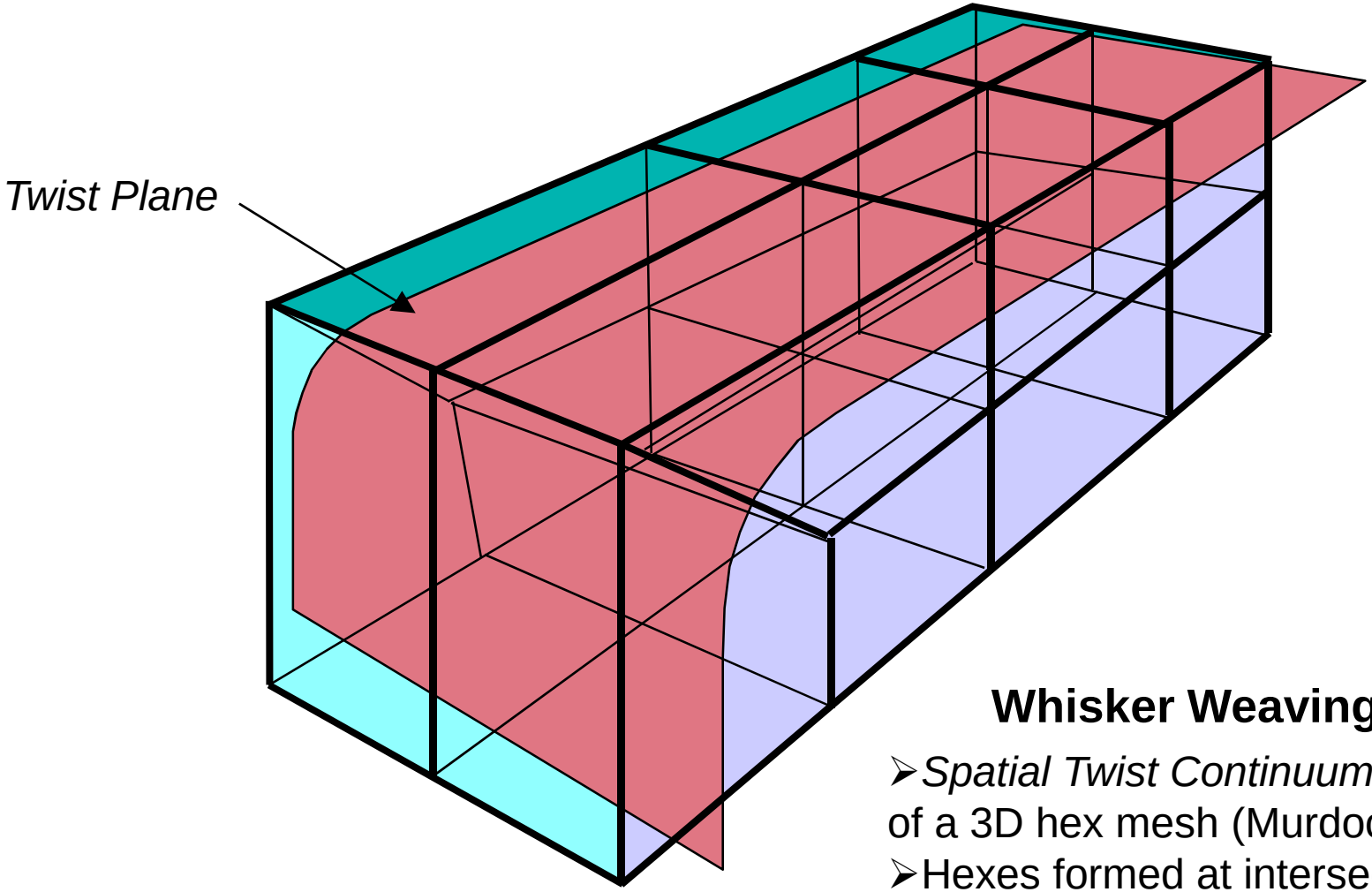
Direct



Whisker Weaving

- *Spatial Twist Continuum* - Dual of a 3D hex mesh (Murdoch, 96)
- Hexes formed at intersection of twist planes

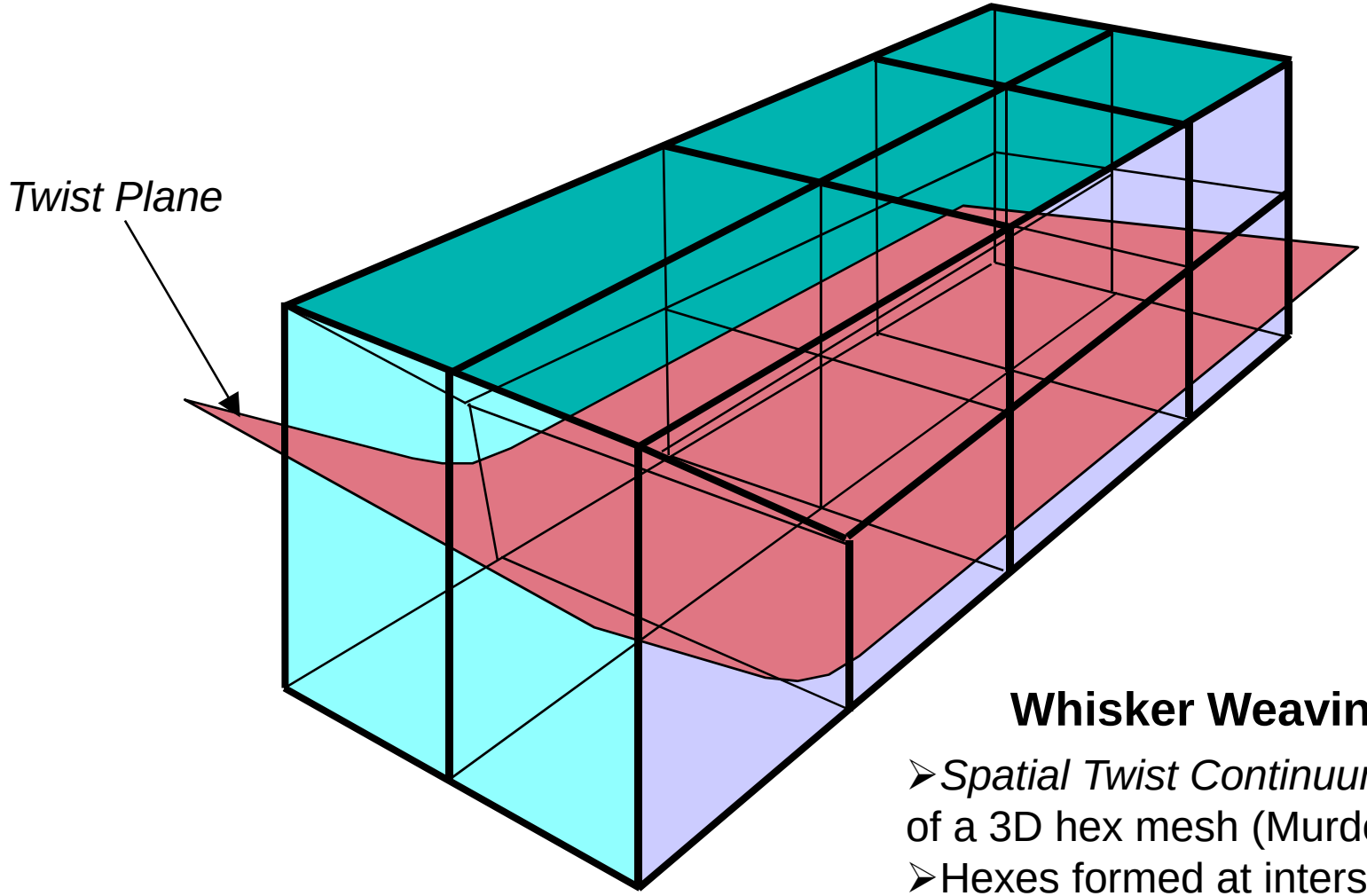
Direct



Whisker Weaving

- *Spatial Twist Continuum* - Dual of a 3D hex mesh (Murdoch, 96)
- Hexes formed at intersection of twist planes

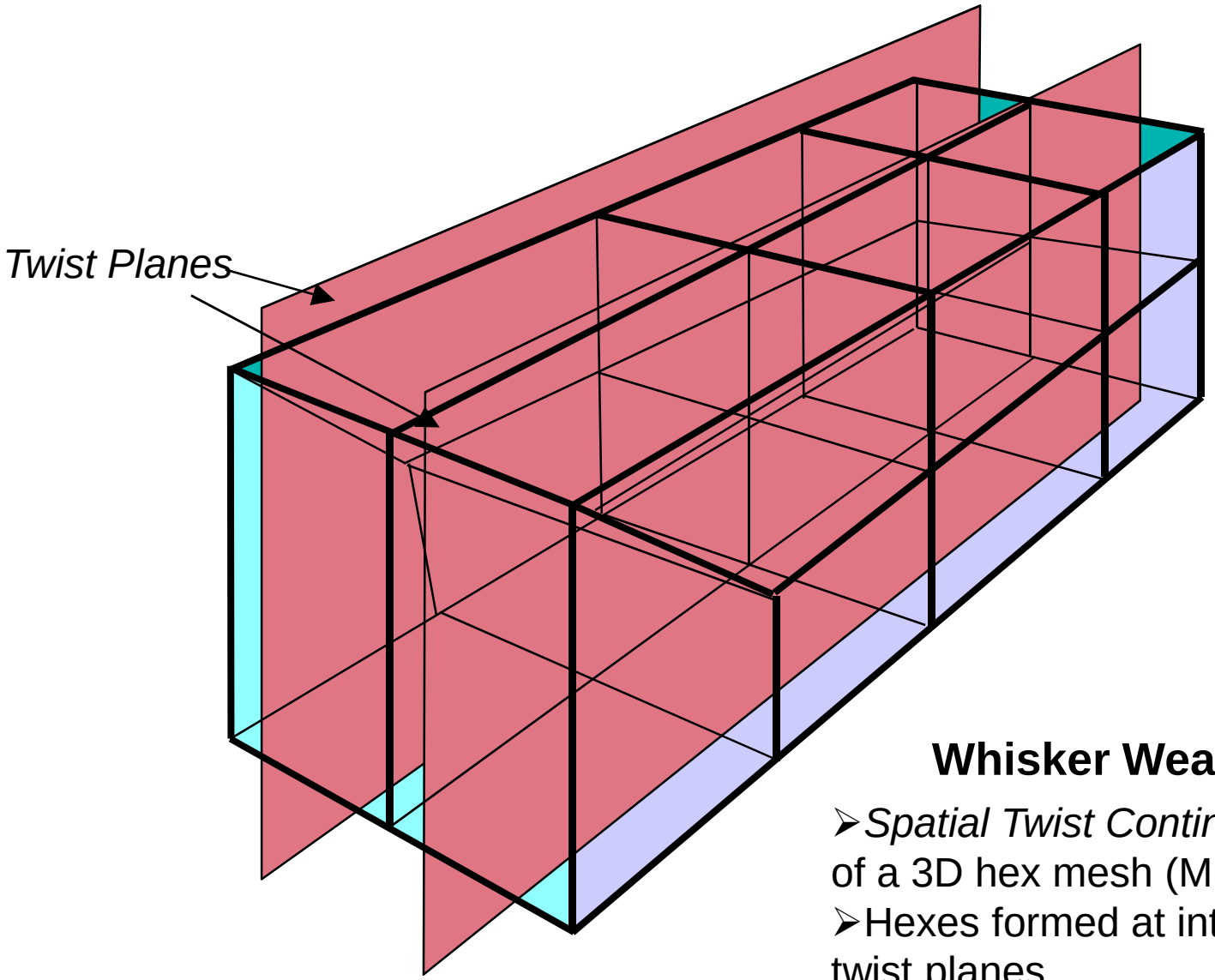
Direct



Whisker Weaving

- *Spatial Twist Continuum* - Dual of a 3D hex mesh (Murdoch, 96)
- Hexes formed at intersection of twist planes

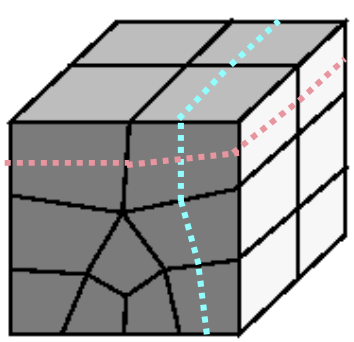
Direct

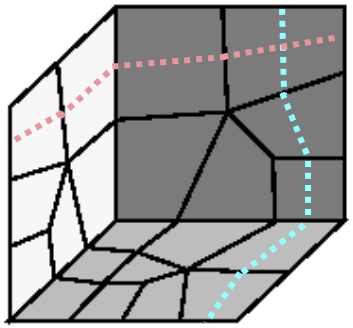


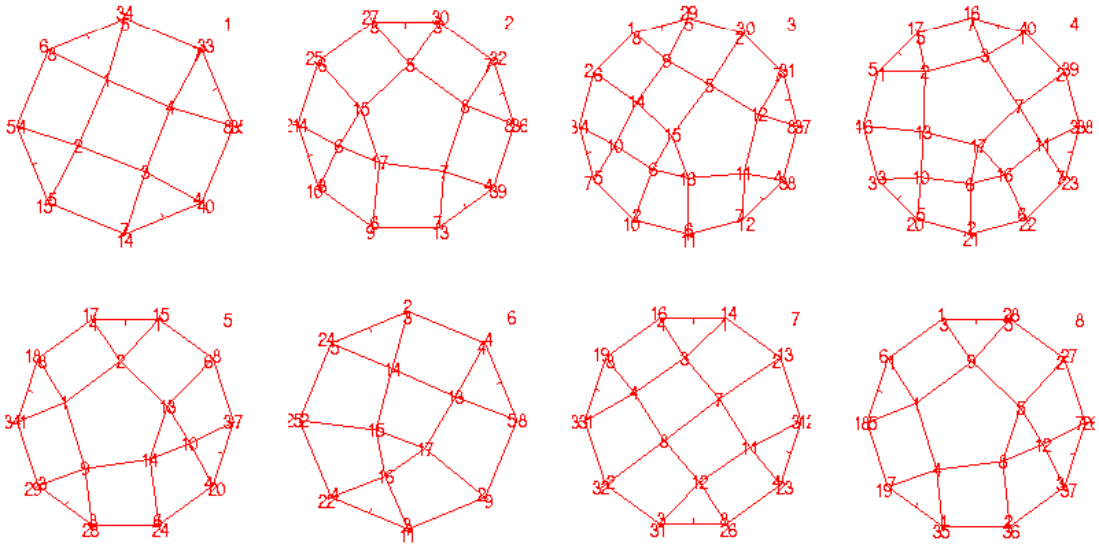
Whisker Weaving

- *Spatial Twist Continuum* - Dual of a 3D hex mesh (Murdoch, 96)
- Hexes formed at intersection of twist planes

Direct







Whisker diagrams used to resolve hex mesh above

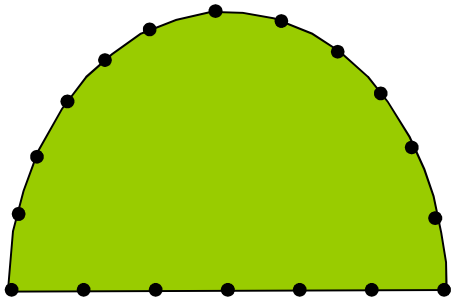
Whisker Weaving

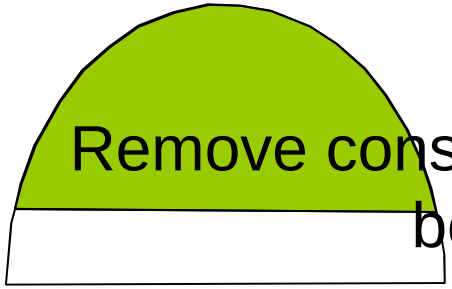
- Define the topology of the twist planes using whisker diagrams
- Each whisker diagram represents a closed loop of the surface dual
- Each boundary vertex on the diagram represents a quad face on the surface
- Objective is to resolve internal connectivity by "weaving" the chords following a set of basic rules

(Tautges,95;96)

Hex Meshing Research

Unconstrained Paving





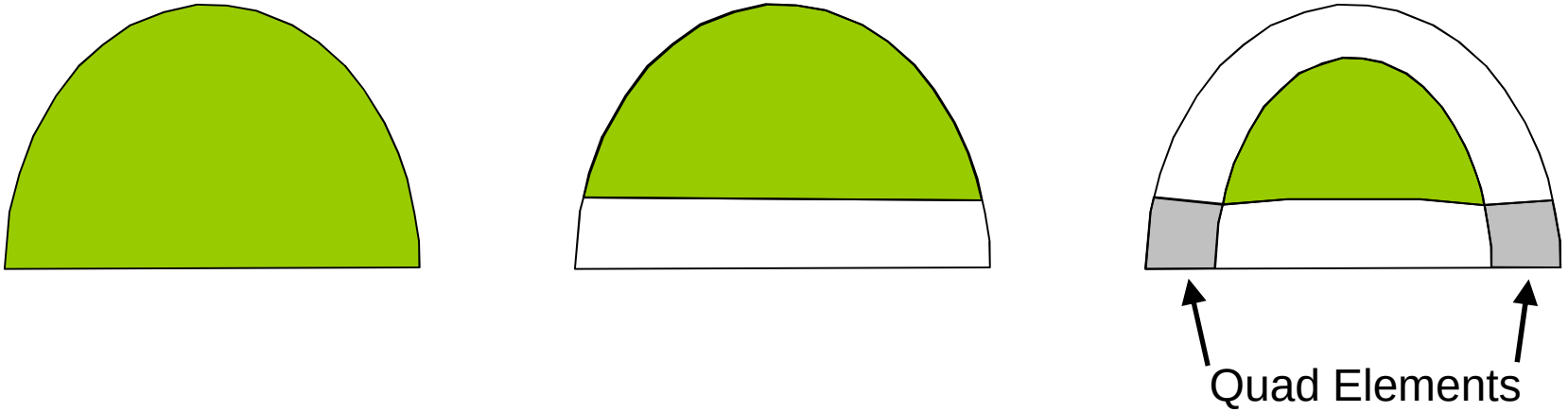
Remove constraint of pre-meshed boundary.

Remove constraint that we must define number of quad when row is advanced. This constrains only 1 DOF.

Hex Meshing Research

Unconstrained Paving

Each Row Advancement Constrains Only 1 DOF

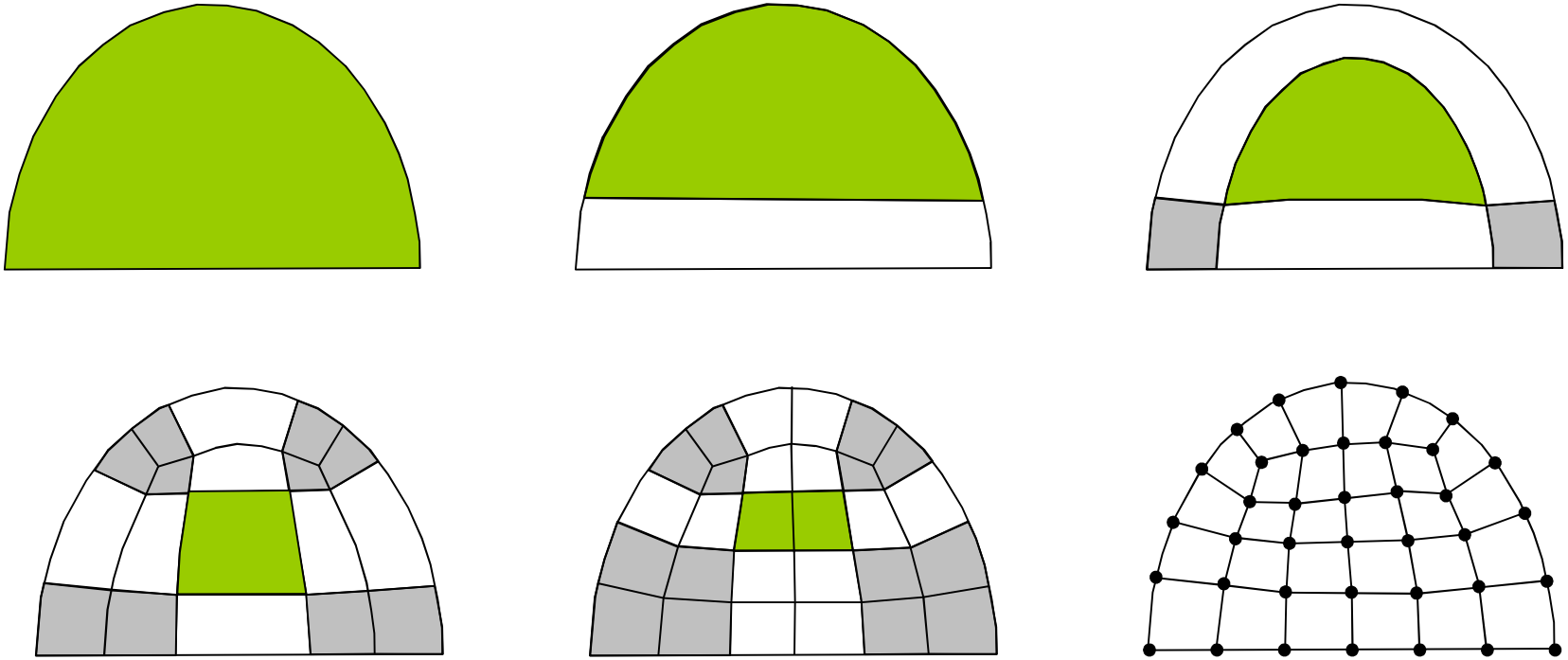


Quads are only completely defined when 2 unconstrained rows cross

Hex Meshing Research

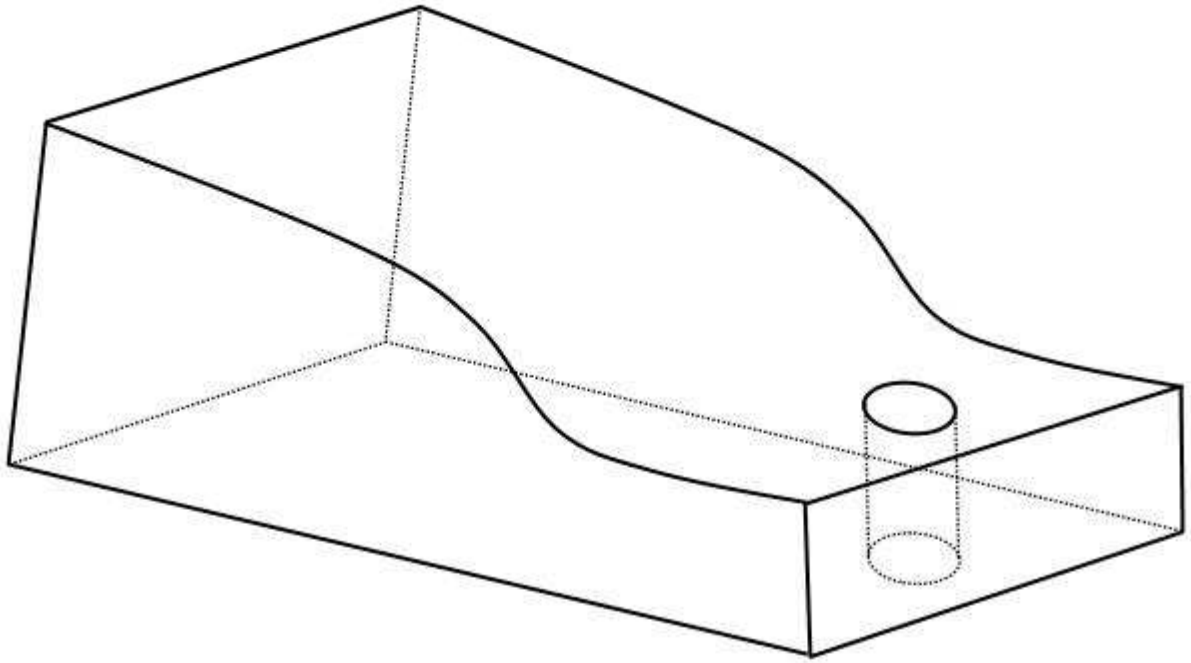
Unconstrained Paving

Each Row Advancement Constrains Only 1 DOF



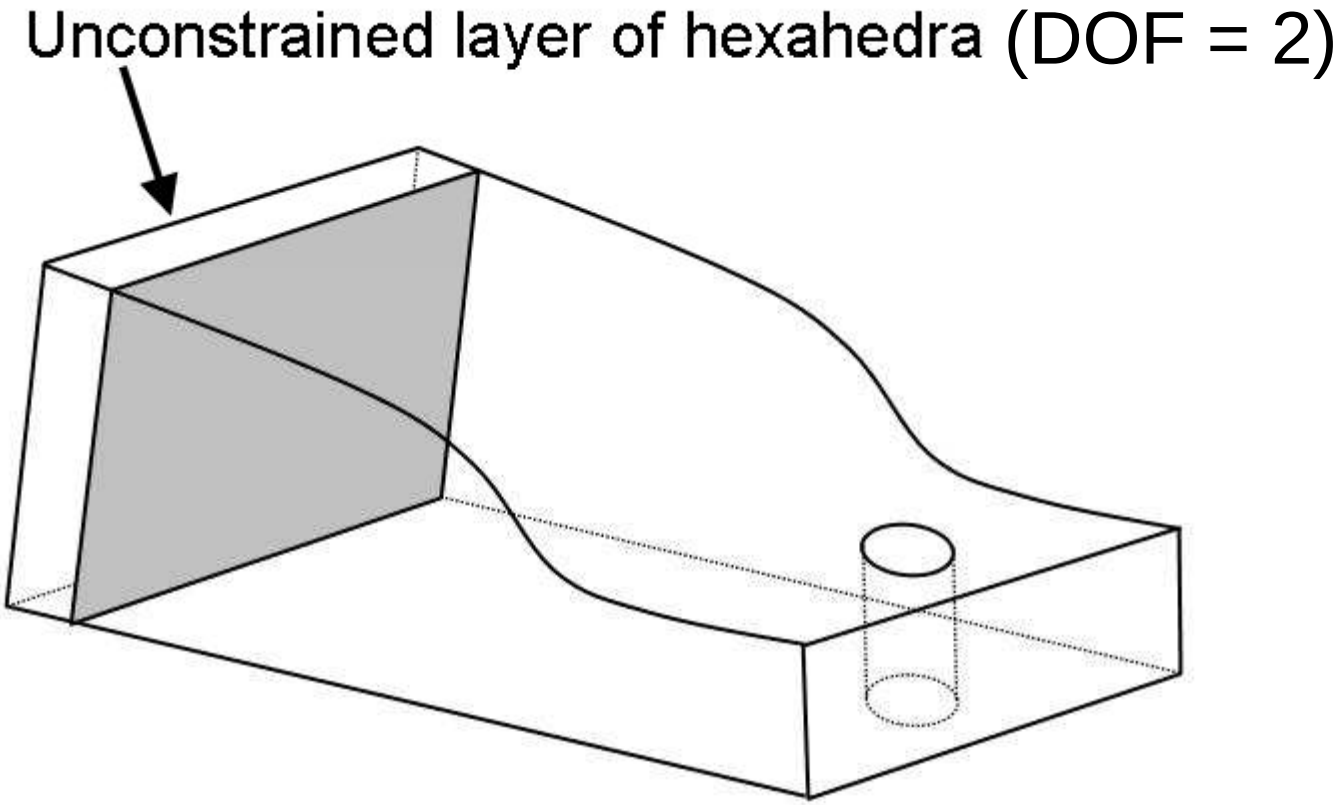
Hex Meshing Research

Unconstrained Plastering



Hex Meshing Research

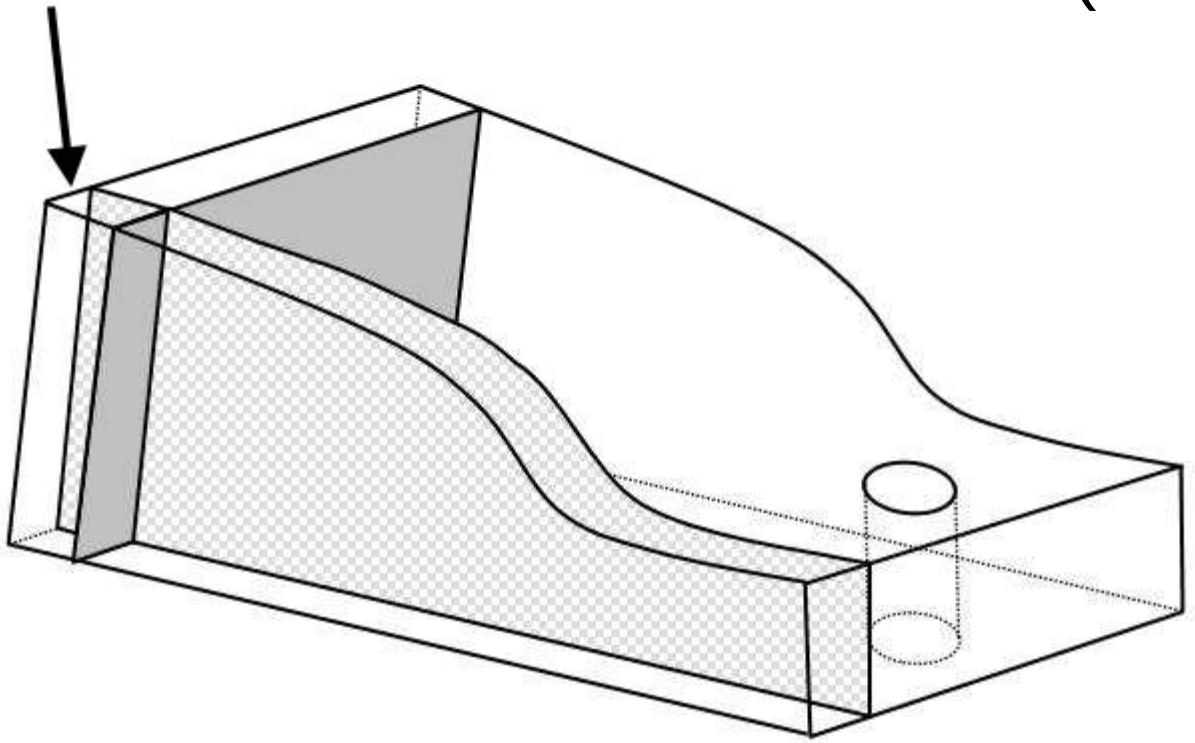
Unconstrained Plastering



Hex Meshing Research

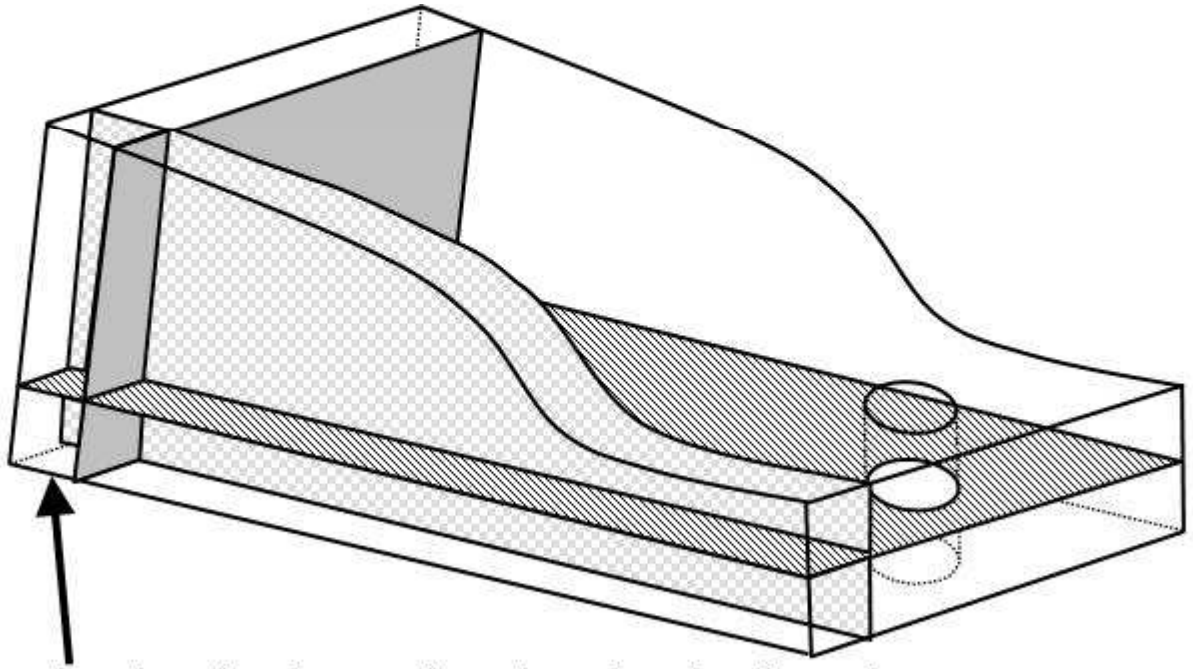
Unconstrained Plastering

Unconstrained column of hexahedra (DOF = 1)



Hex Meshing Research

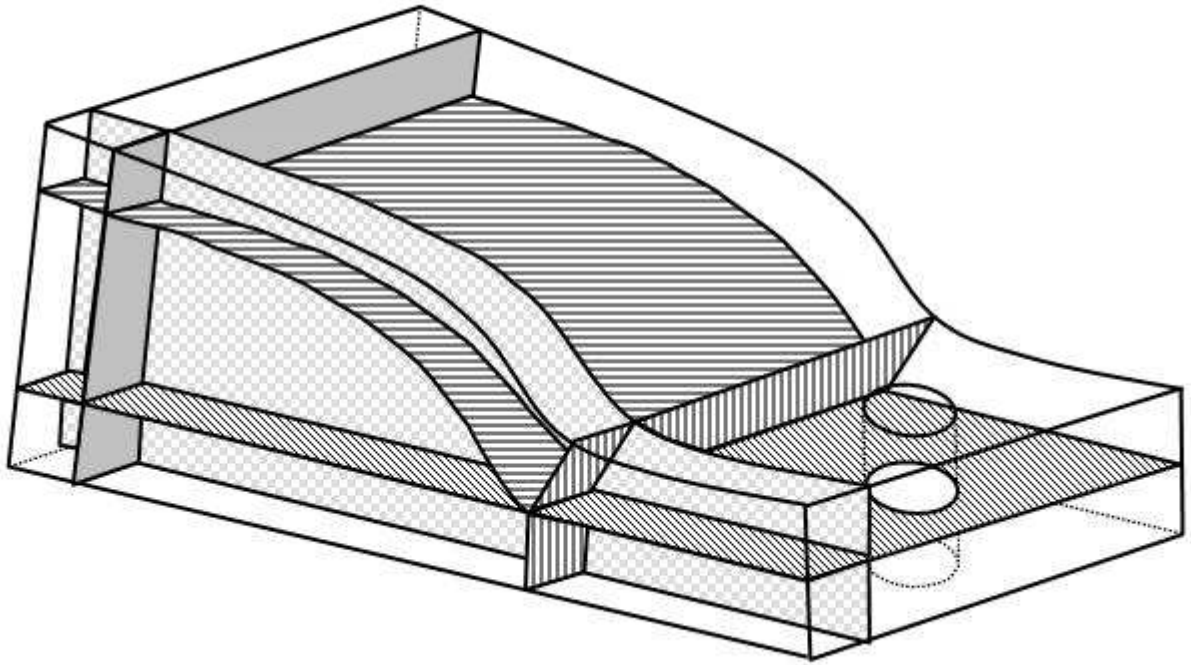
Unconstrained Plastering



A single hexahedra is defined (DOF = 0)

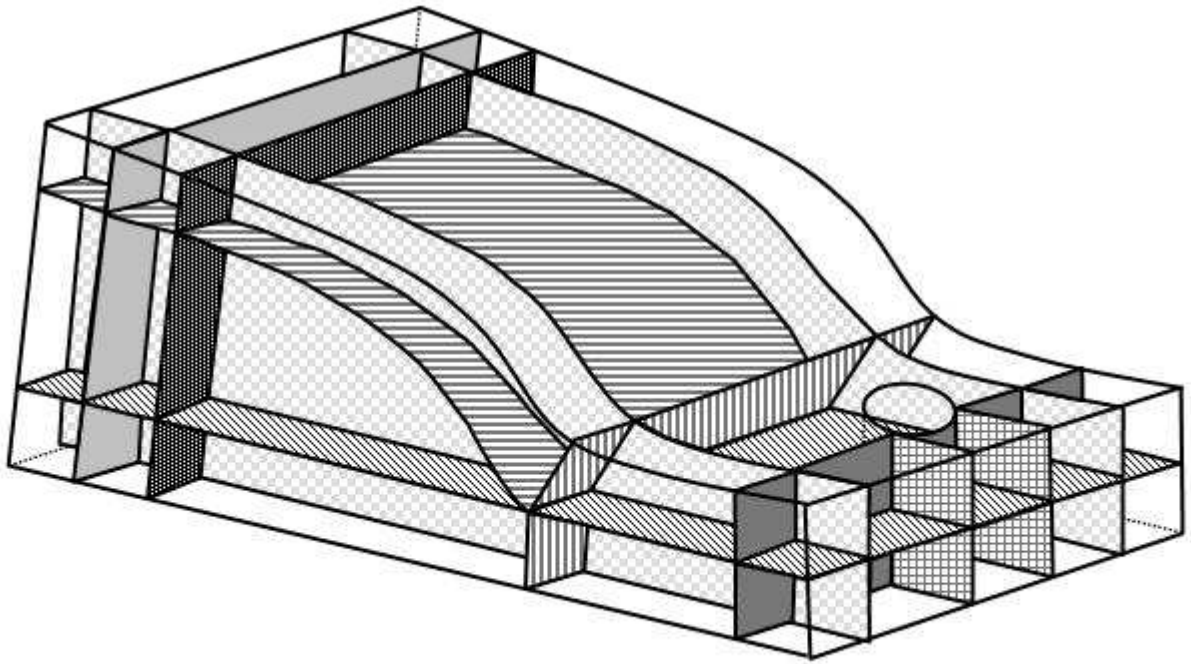
Hex Meshing Research

Unconstrained Plastering



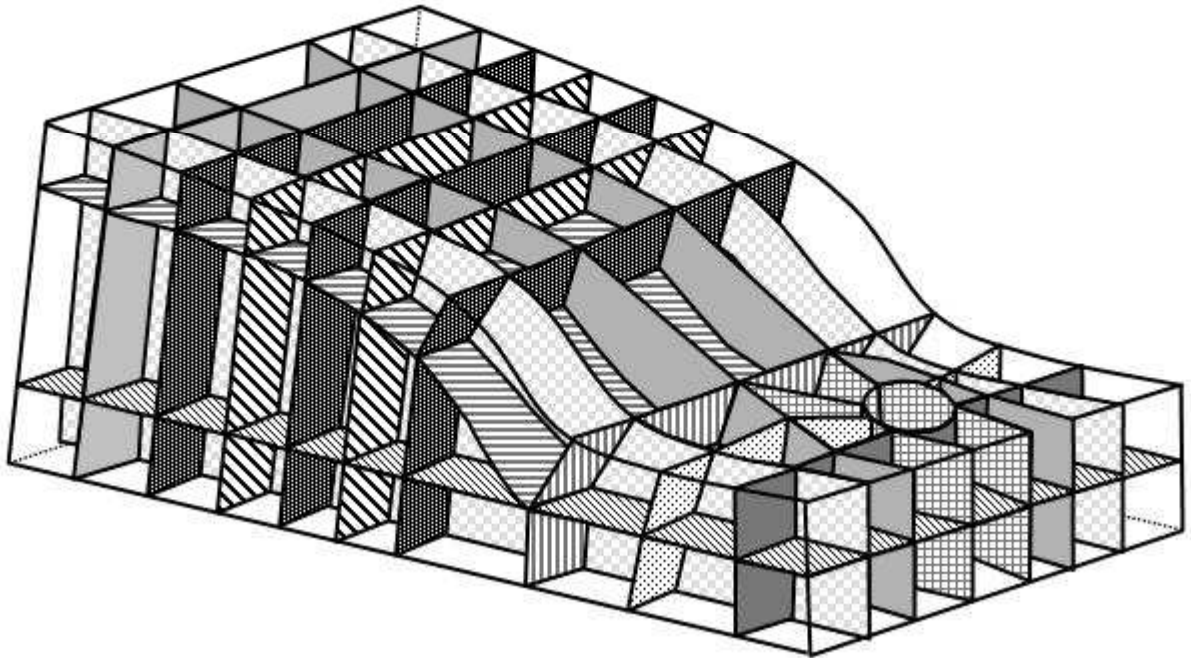
Hex Meshing Research

Unconstrained Plastering



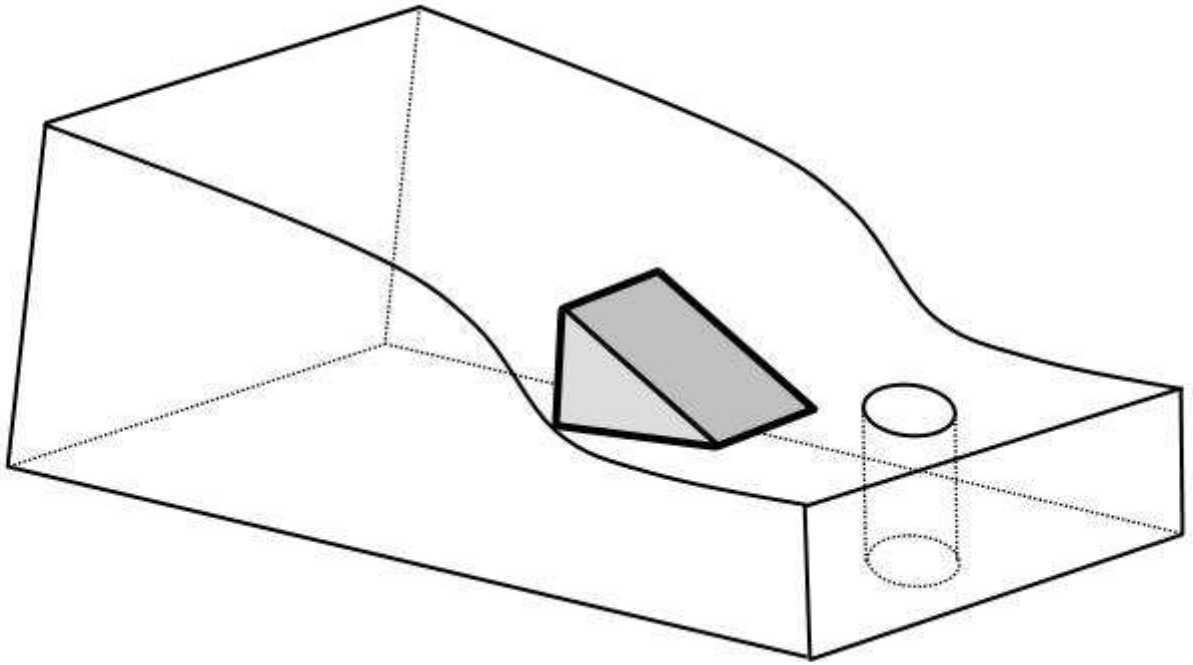
Hex Meshing Research

Unconstrained Plastering



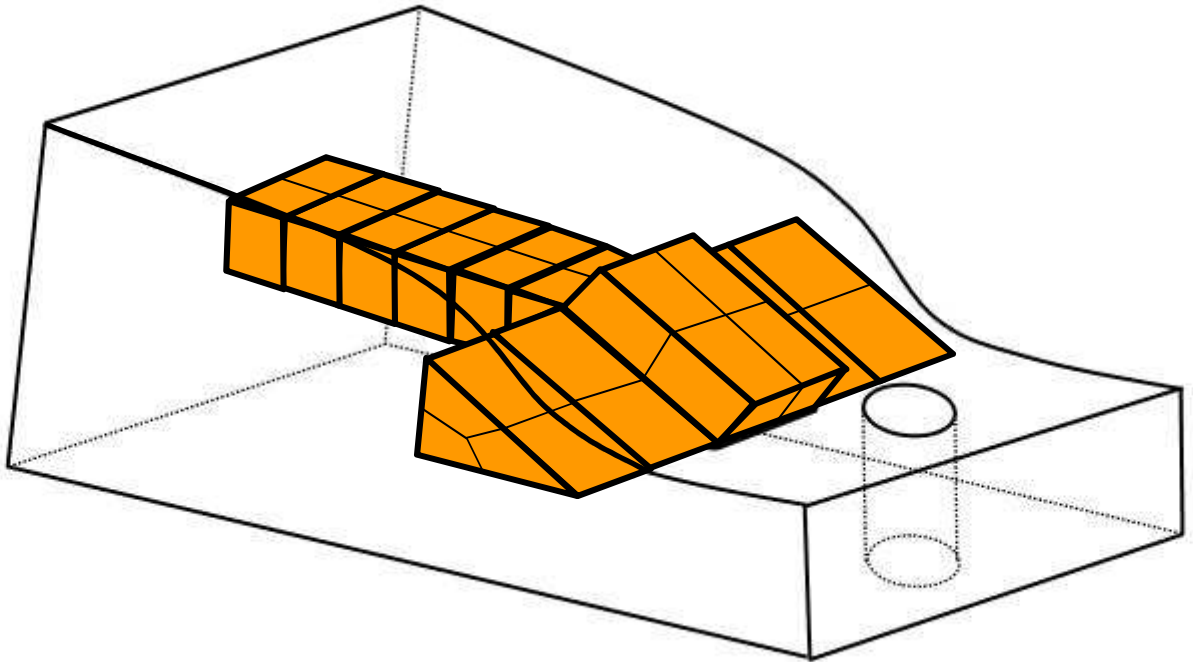
Hex Meshing Research

Unconstrained Plastering



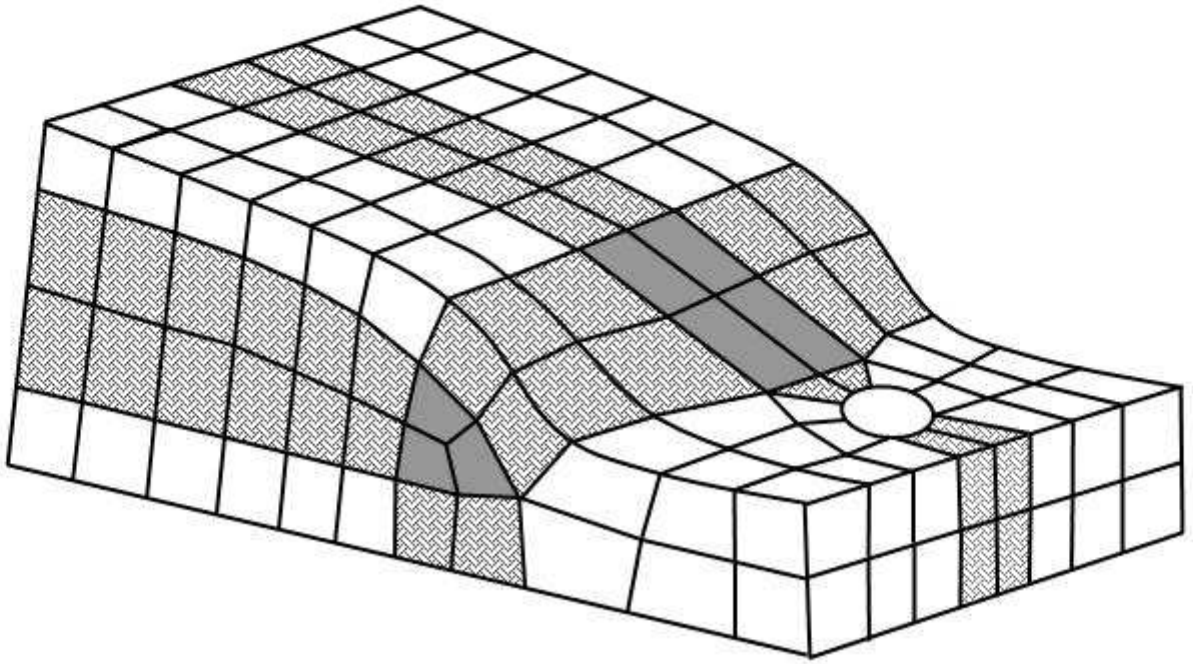
Hex Meshing Research

Unconstrained Plastering



Hex Meshing Research

Unconstrained Plastering



Hybrid Methods

CFD Meshing

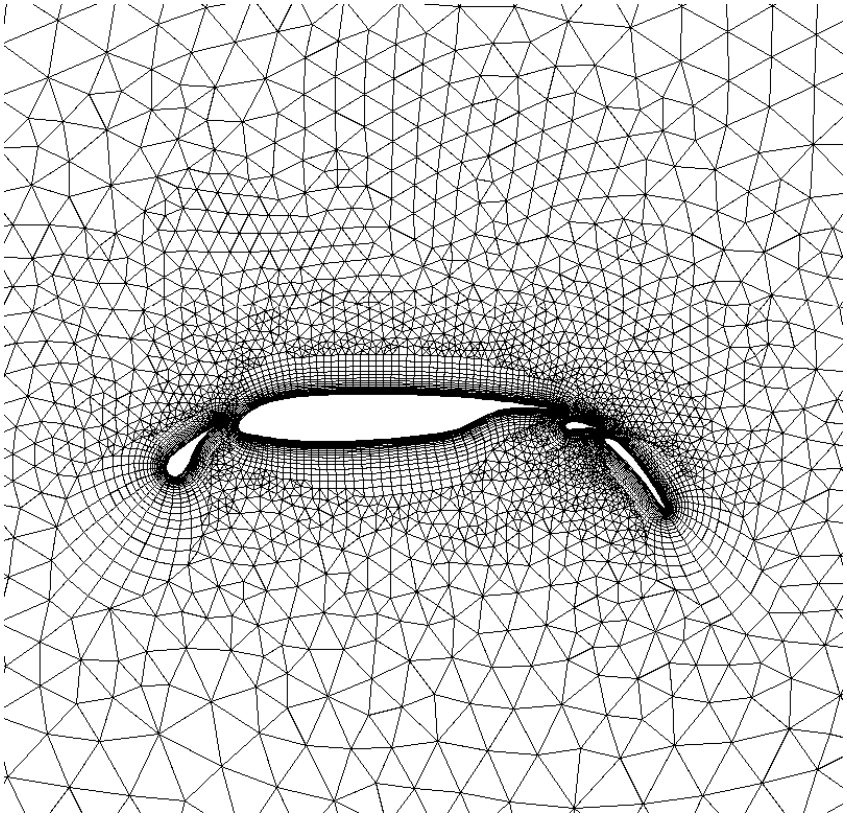


Image courtesy of Roy P. Koomullil, Engineering Research Center, Mississippi State University, http://www.erc.msstate.edu/~roy/

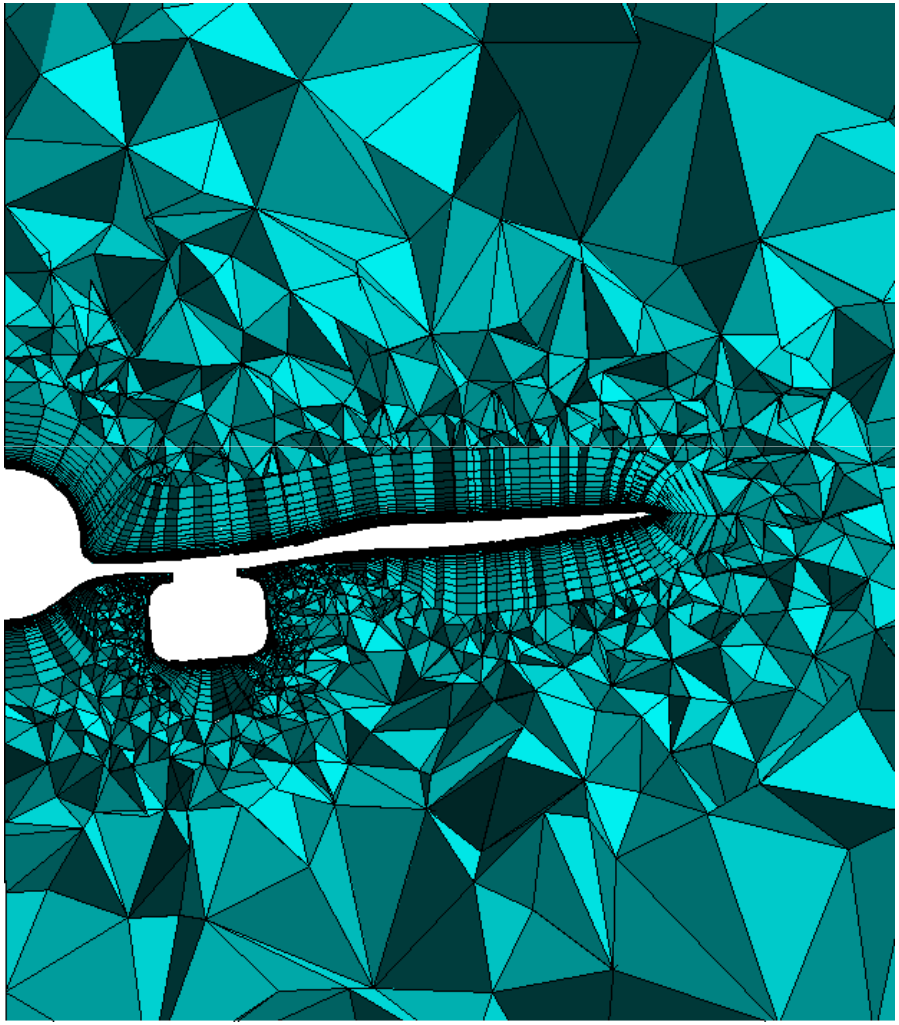
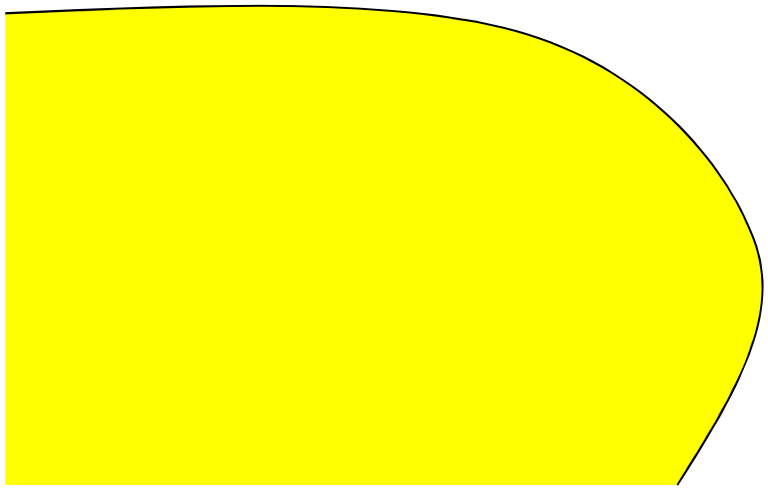


Image courtesy of acelab, University of Texas, Austin, http://acelab.ae.utexas.edu

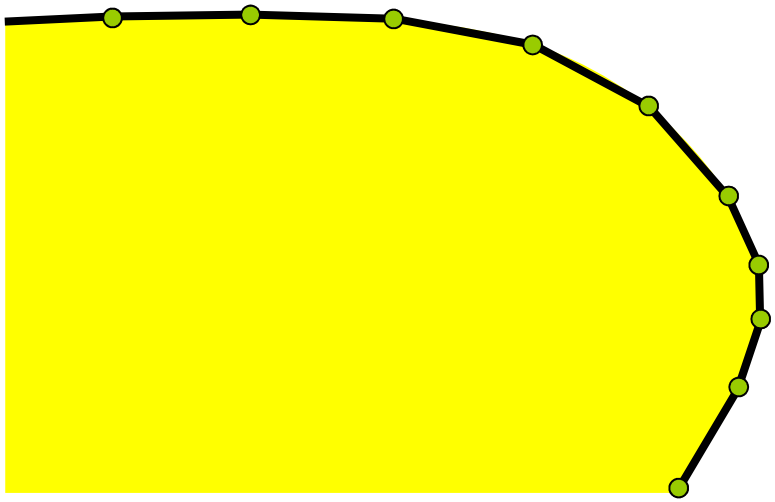
Hybrid Methods

Advancing Layers Method



Hybrid Methods

Advancing Layers Method

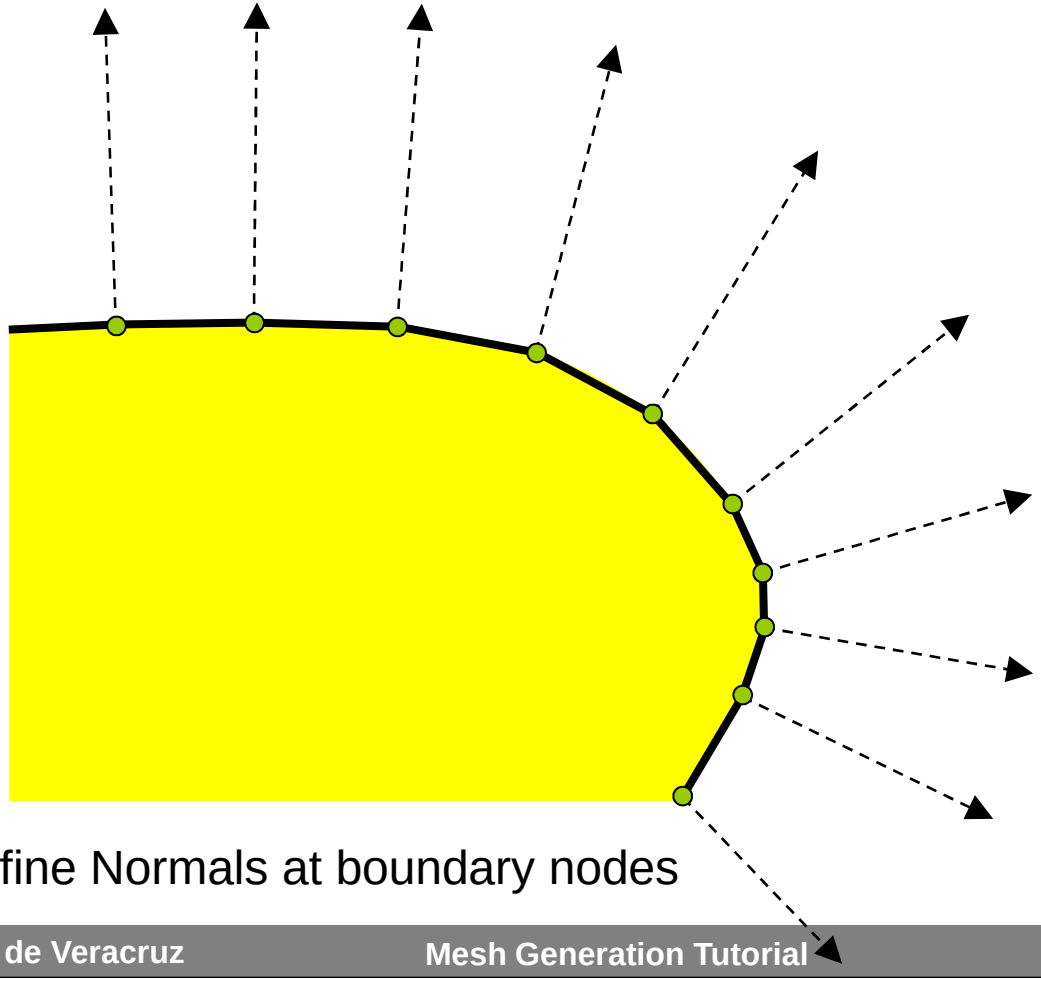


Discretized Boundary

Hybrid Methods

Advancing Layers Method

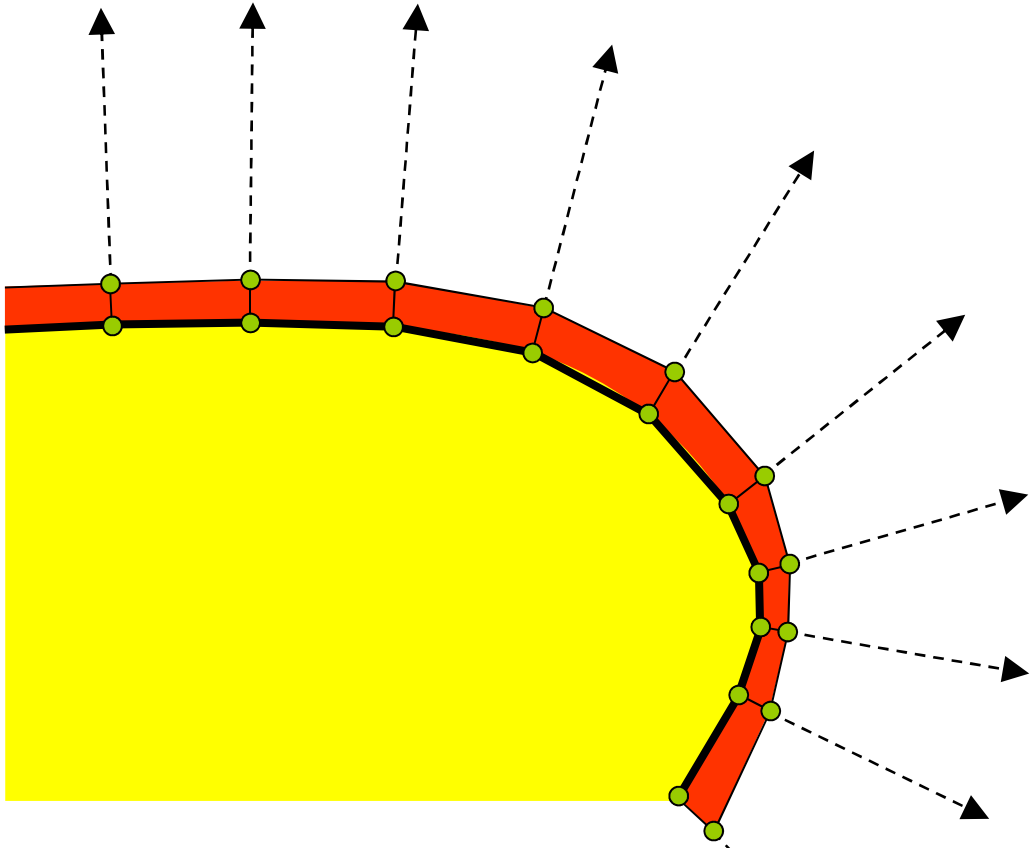
(Pirzadeh, 1994)

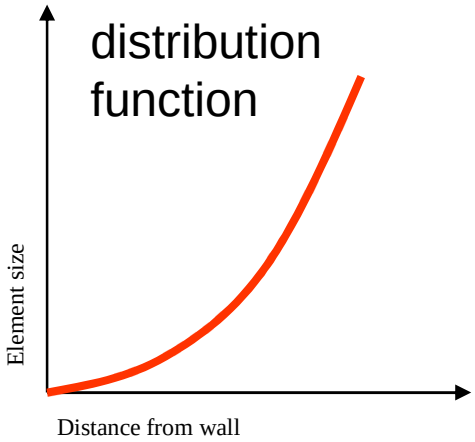


Define Normals at boundary nodes

Hybrid Methods

Advancing Layers Method

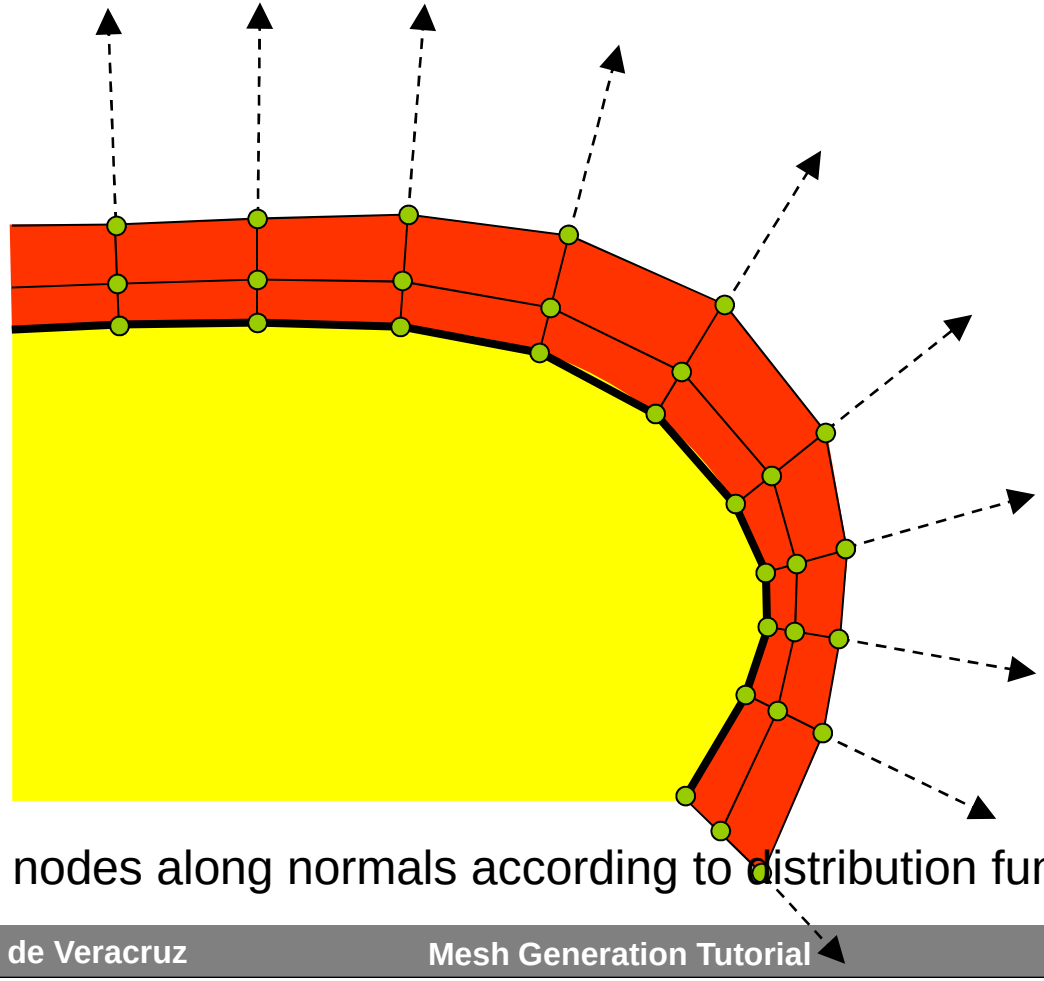


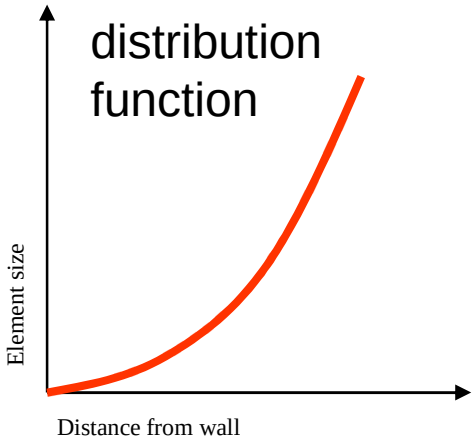


Generate nodes along normals according to distribution function Form layer

Hybrid Methods

Advancing Layers Method

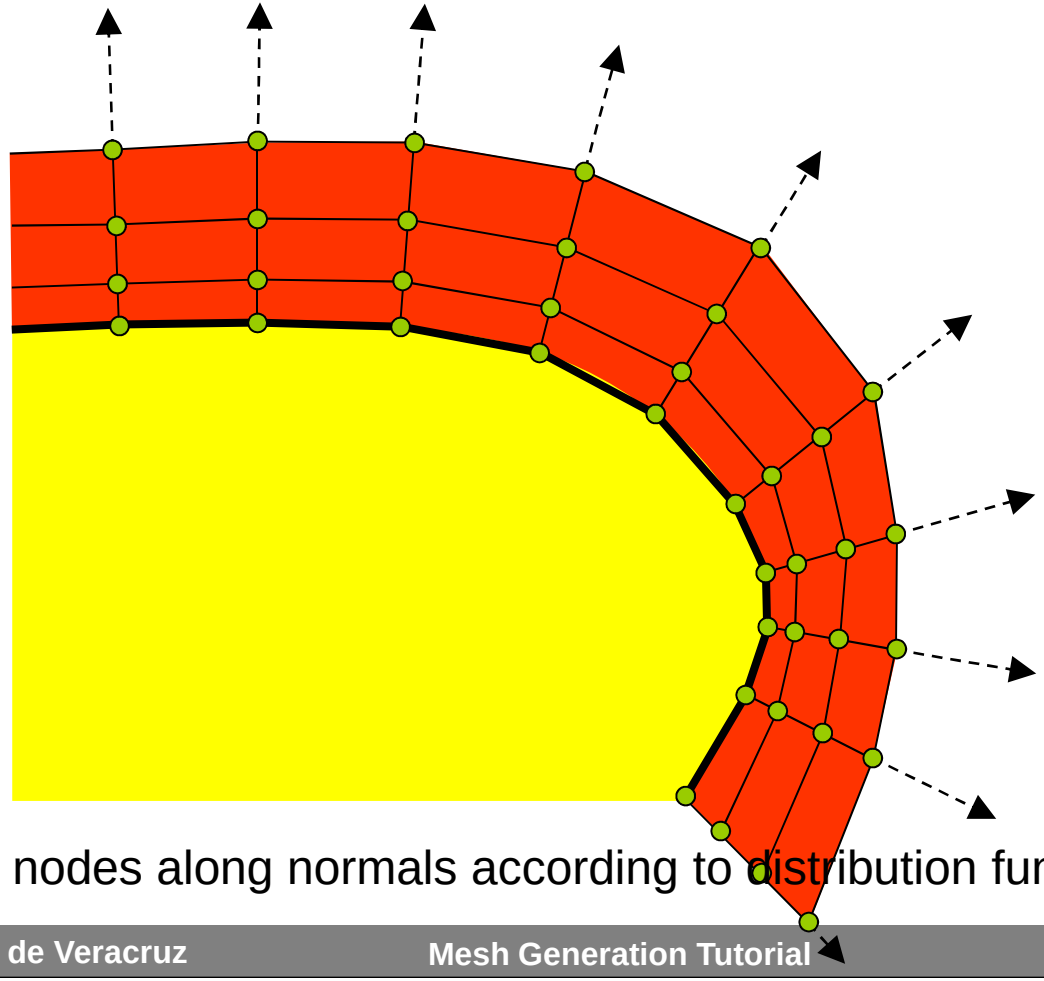


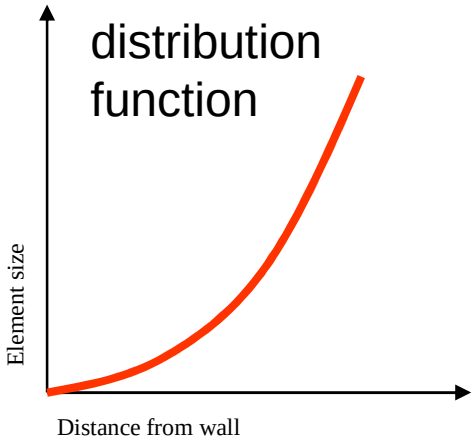


Generate nodes along normals according to distribution function Form layer

Hybrid Methods

Advancing Layers Method

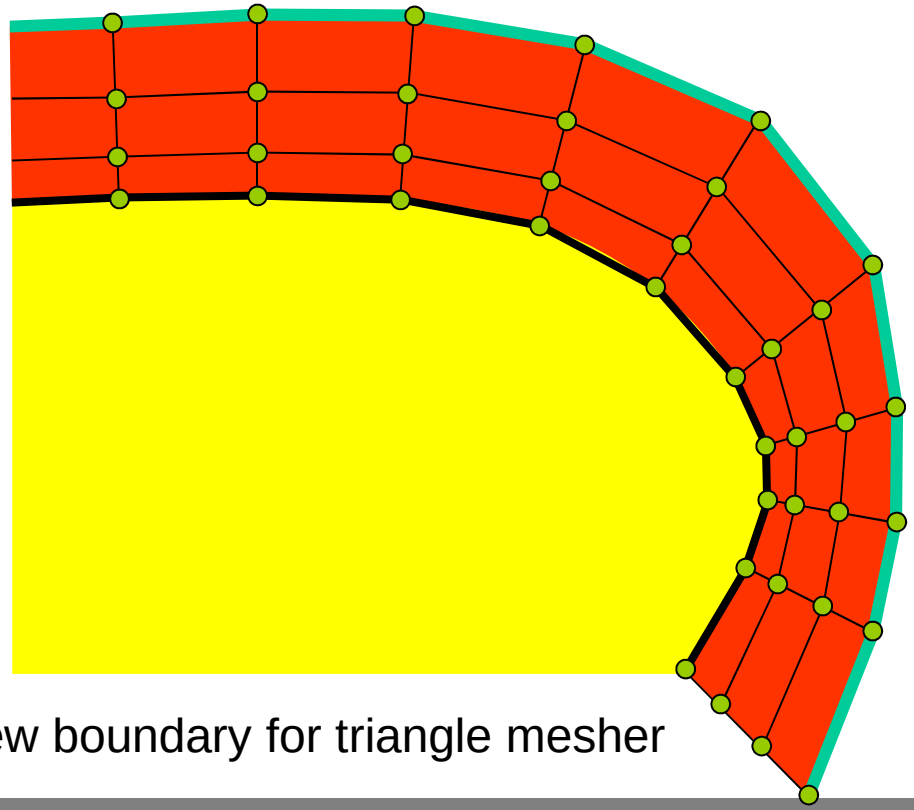




Generate nodes along normals according to distribution function Form layer

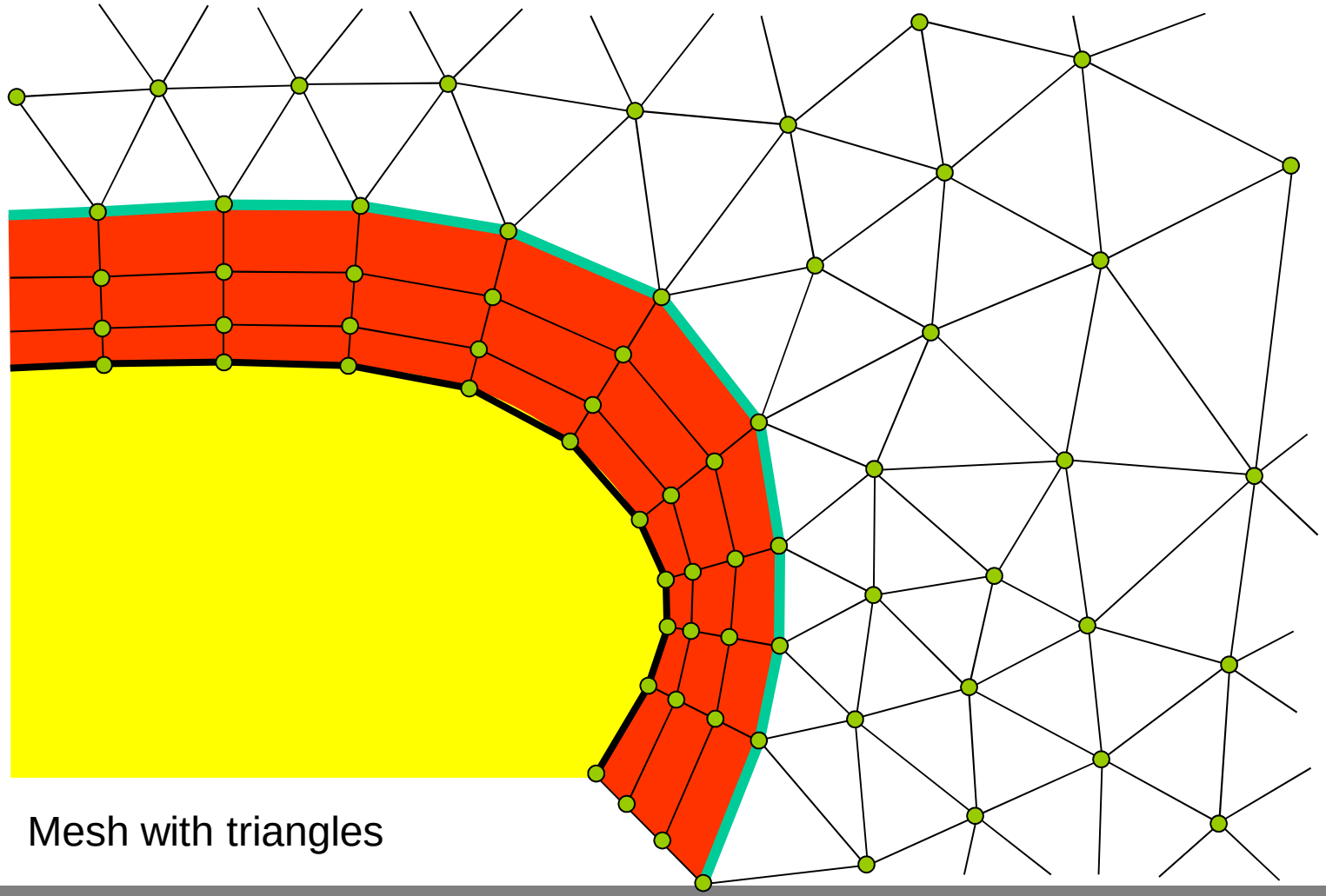
Hybrid Methods

Advancing Layers Method



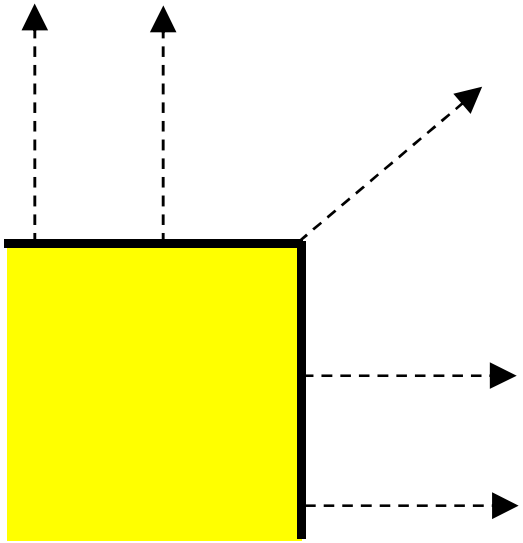
Define new boundary for triangle mesher

Hybrid Methods

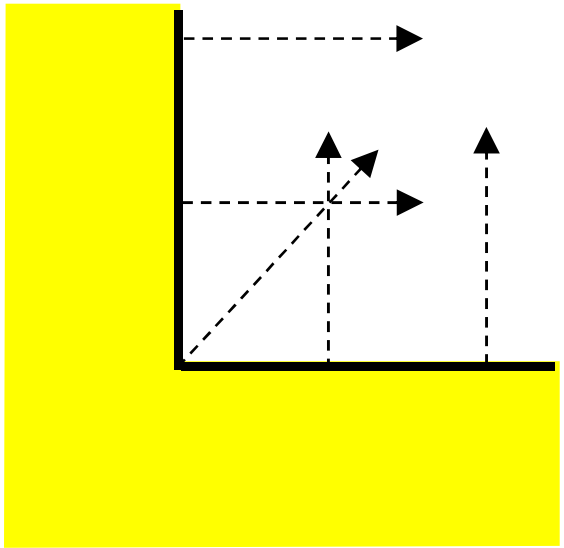


Mesh with triangles

Hybrid Methods

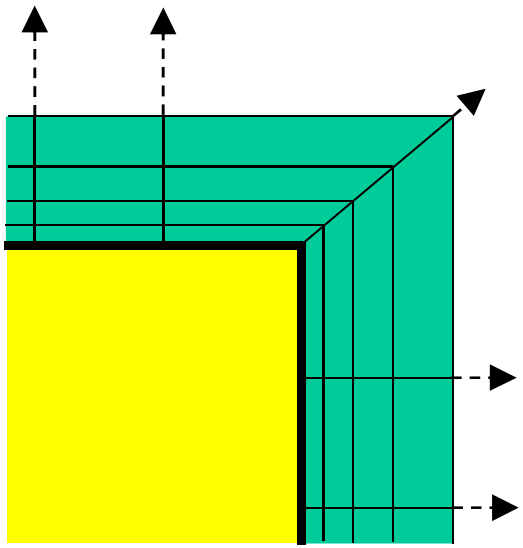


Convex Corner

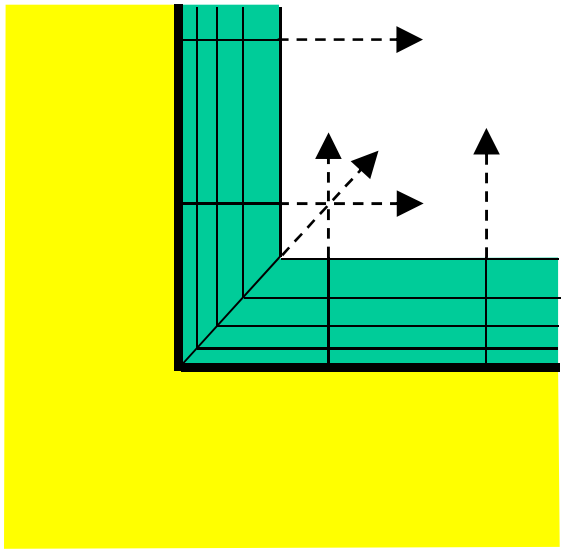


Concave Corner

Hybrid Methods

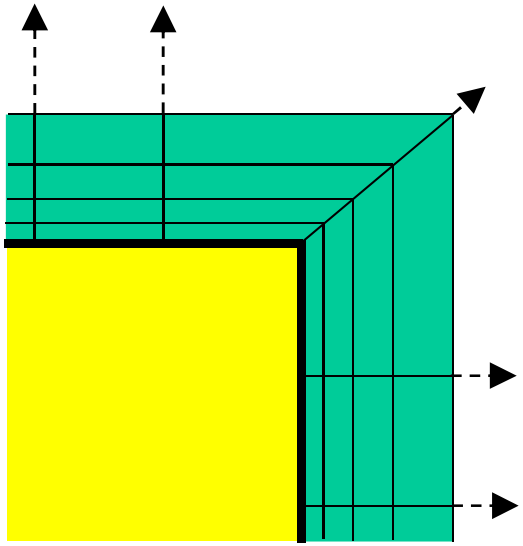


Convex Corner

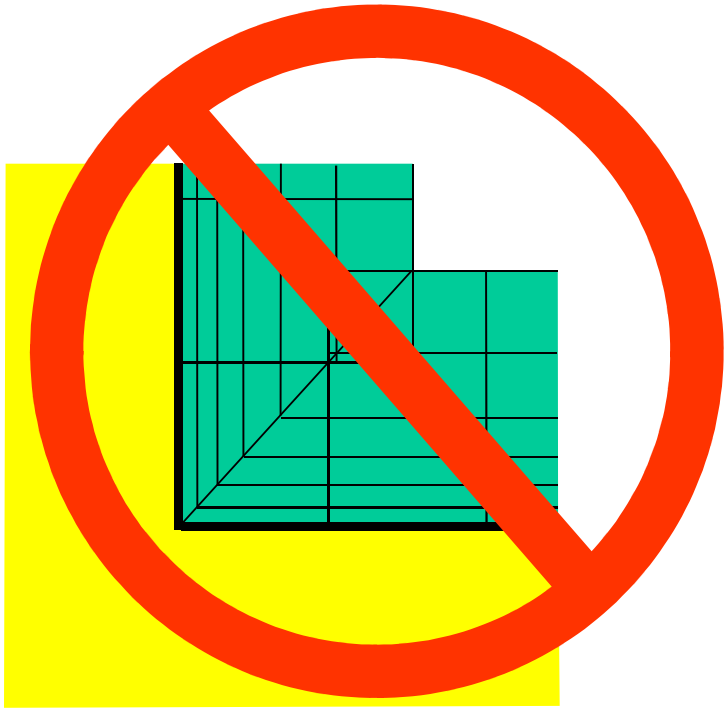


Concave Corner

Hybrid Methods

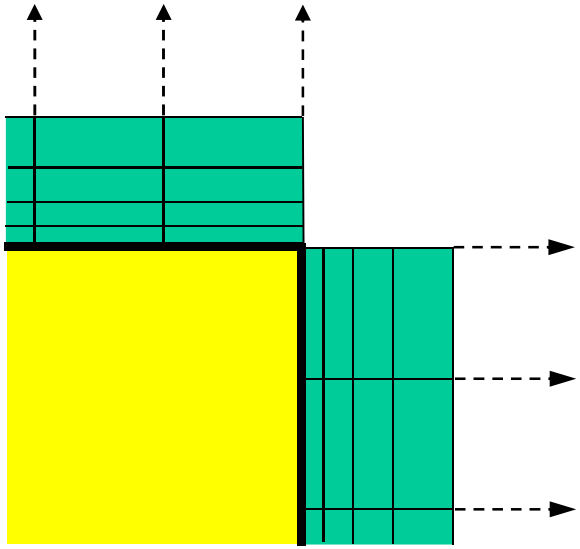


Convex Corner

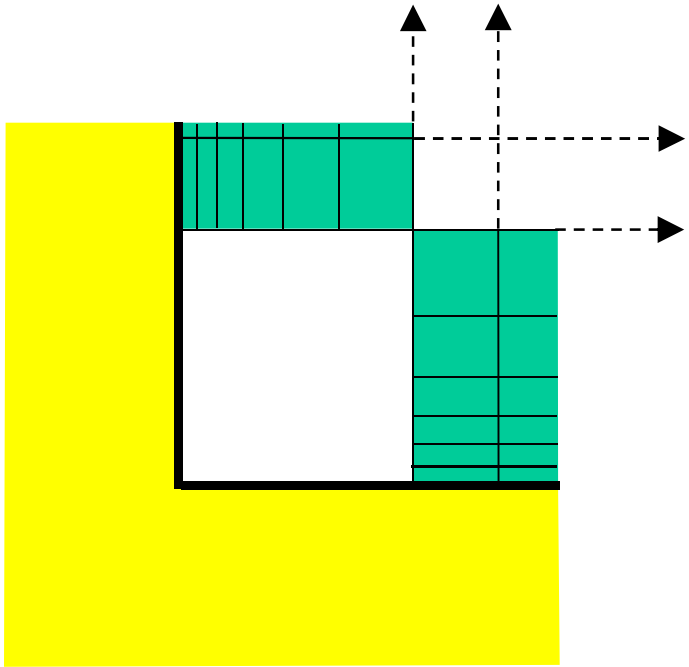


Concave Corner

Hybrid Methods



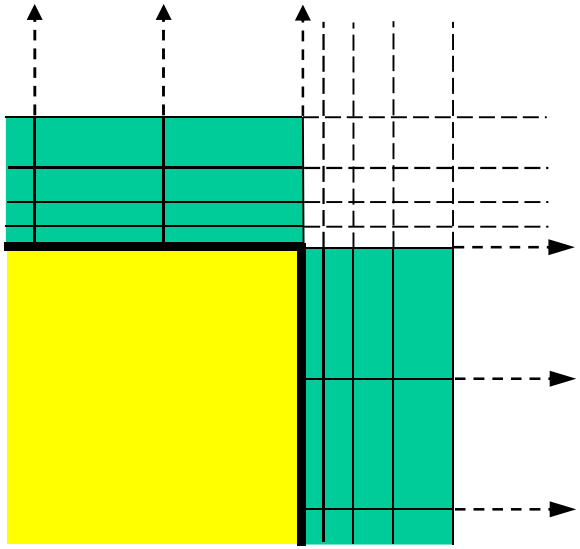
Convex Corner



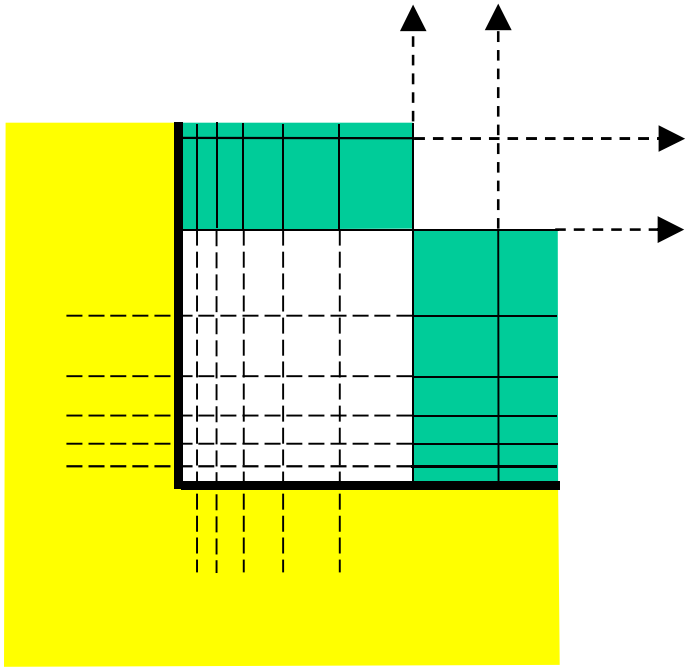
Concave Corner

Blend Regions

Hybrid Methods



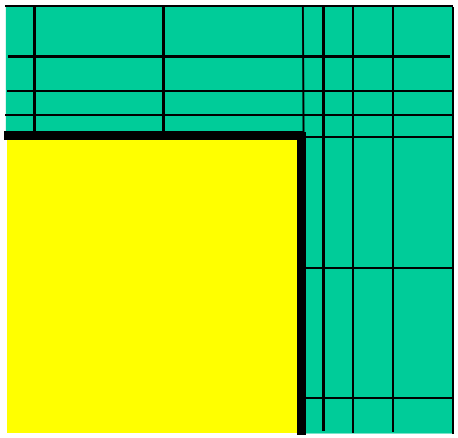
Convex Corner



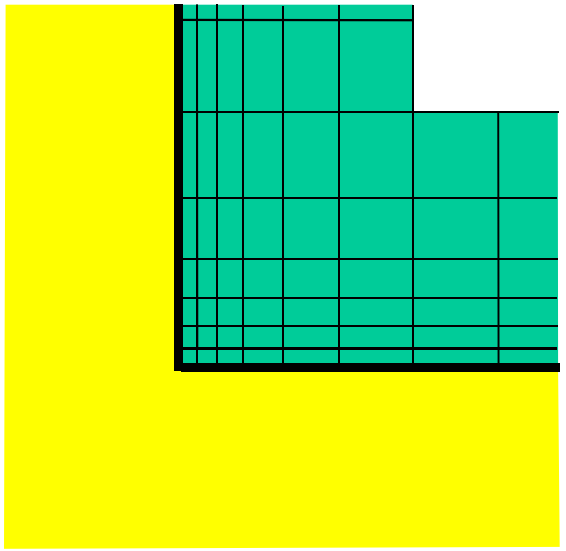
Concave Corner

Blend Regions

Hybrid Methods



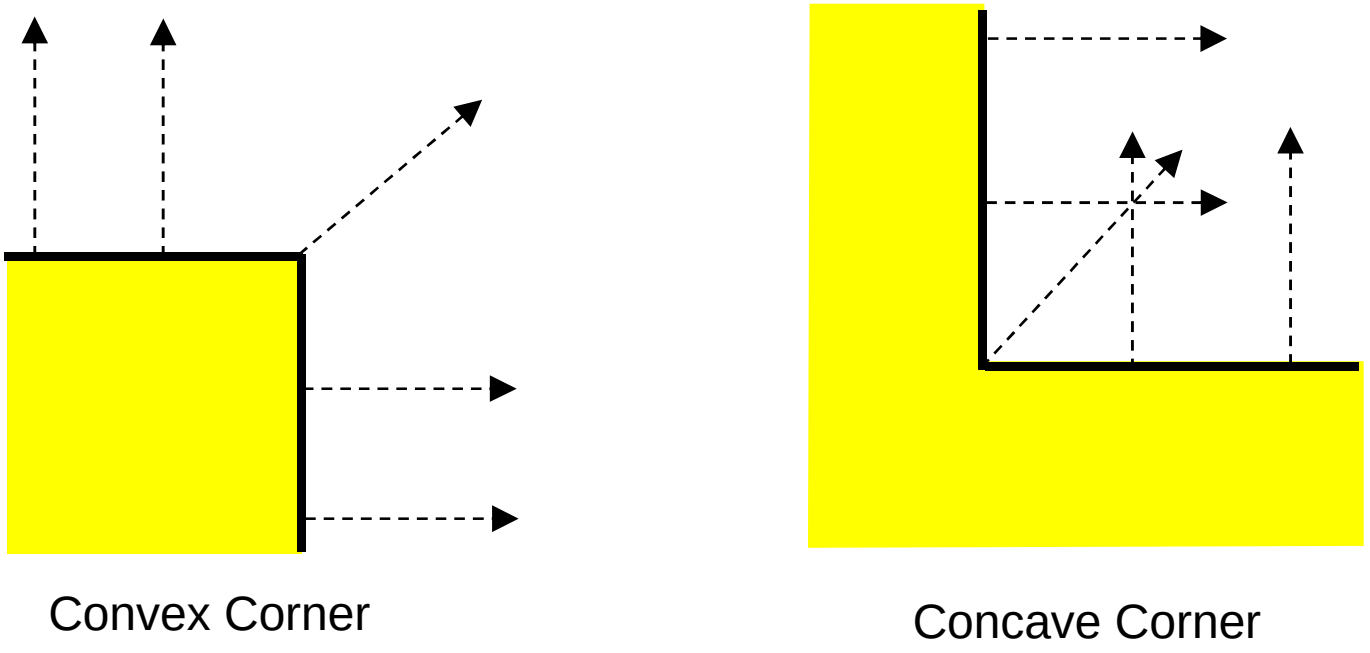
Convex Corner



Concave Corner

Blend Regions

Hybrid Methods

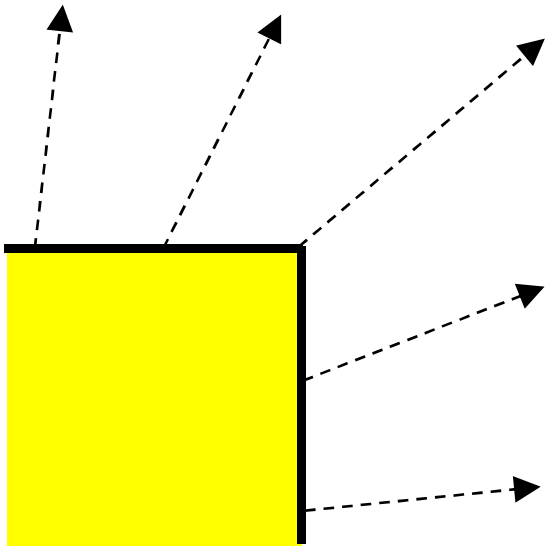


Convex Corner

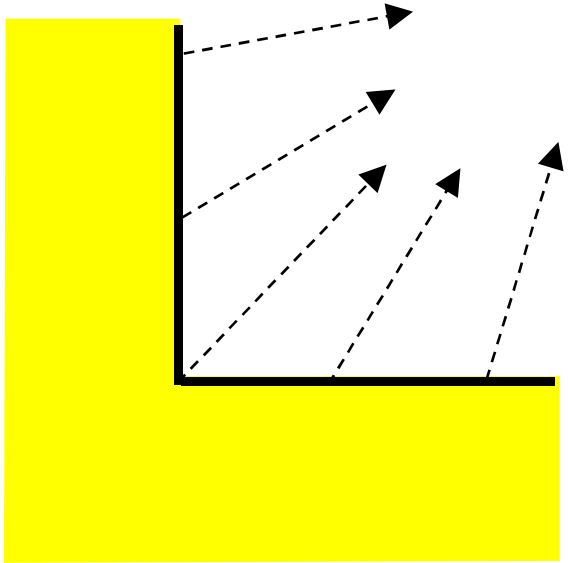
Concave Corner

Smoothed Normals

Hybrid Methods



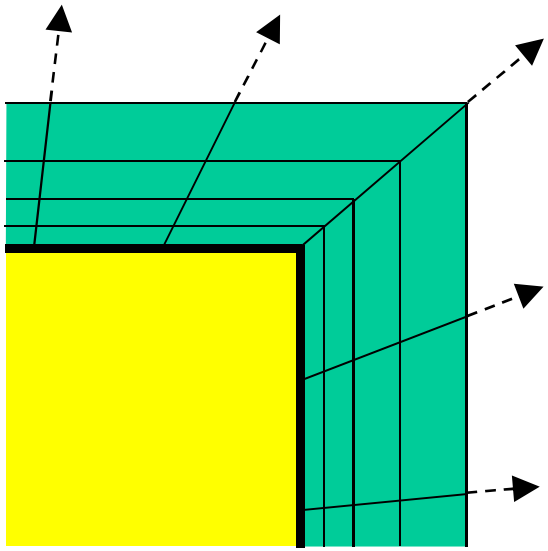
Convex Corner



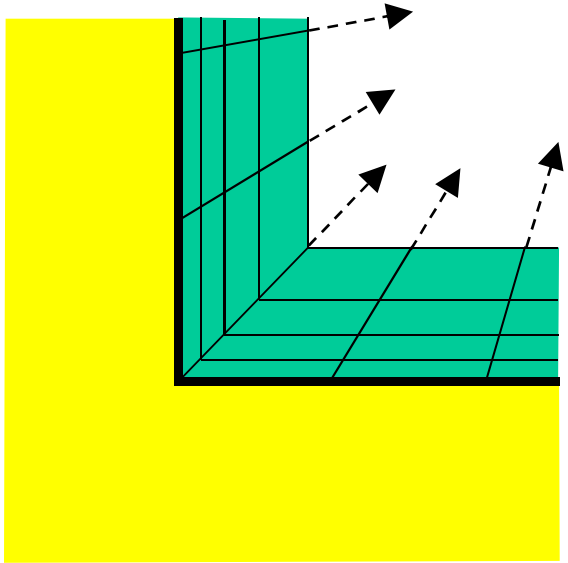
Concave Corner

Smoothed Normals

Hybrid Methods



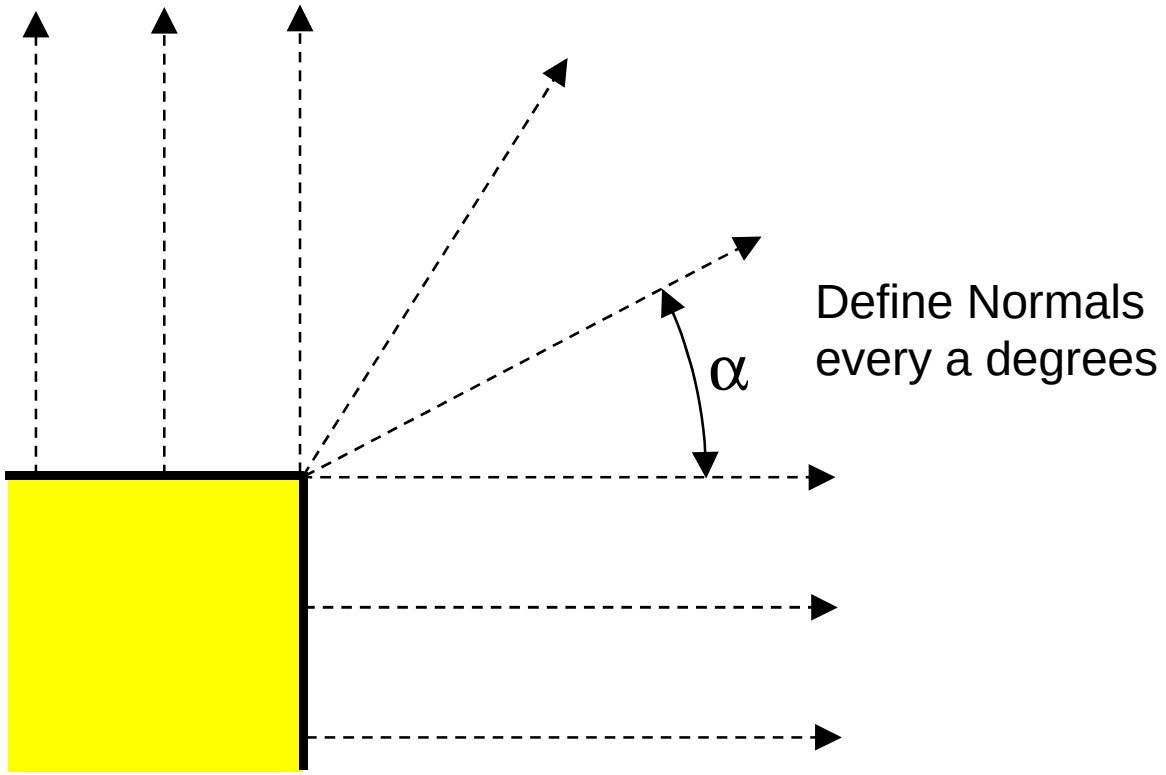
Convex Corner



Concave Corner

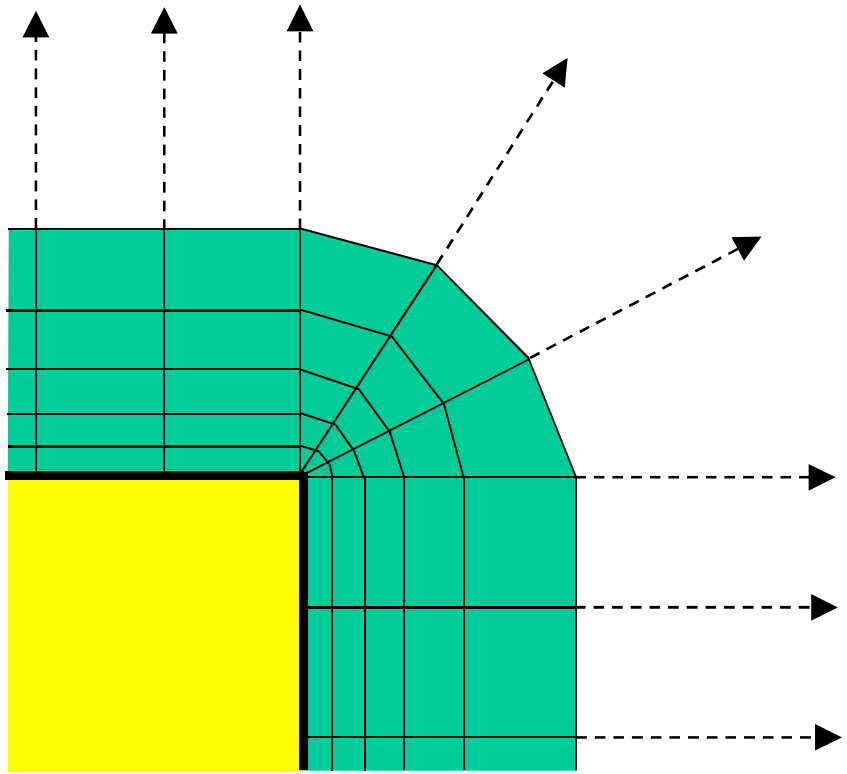
Smoothed Normals

Hybrid Methods



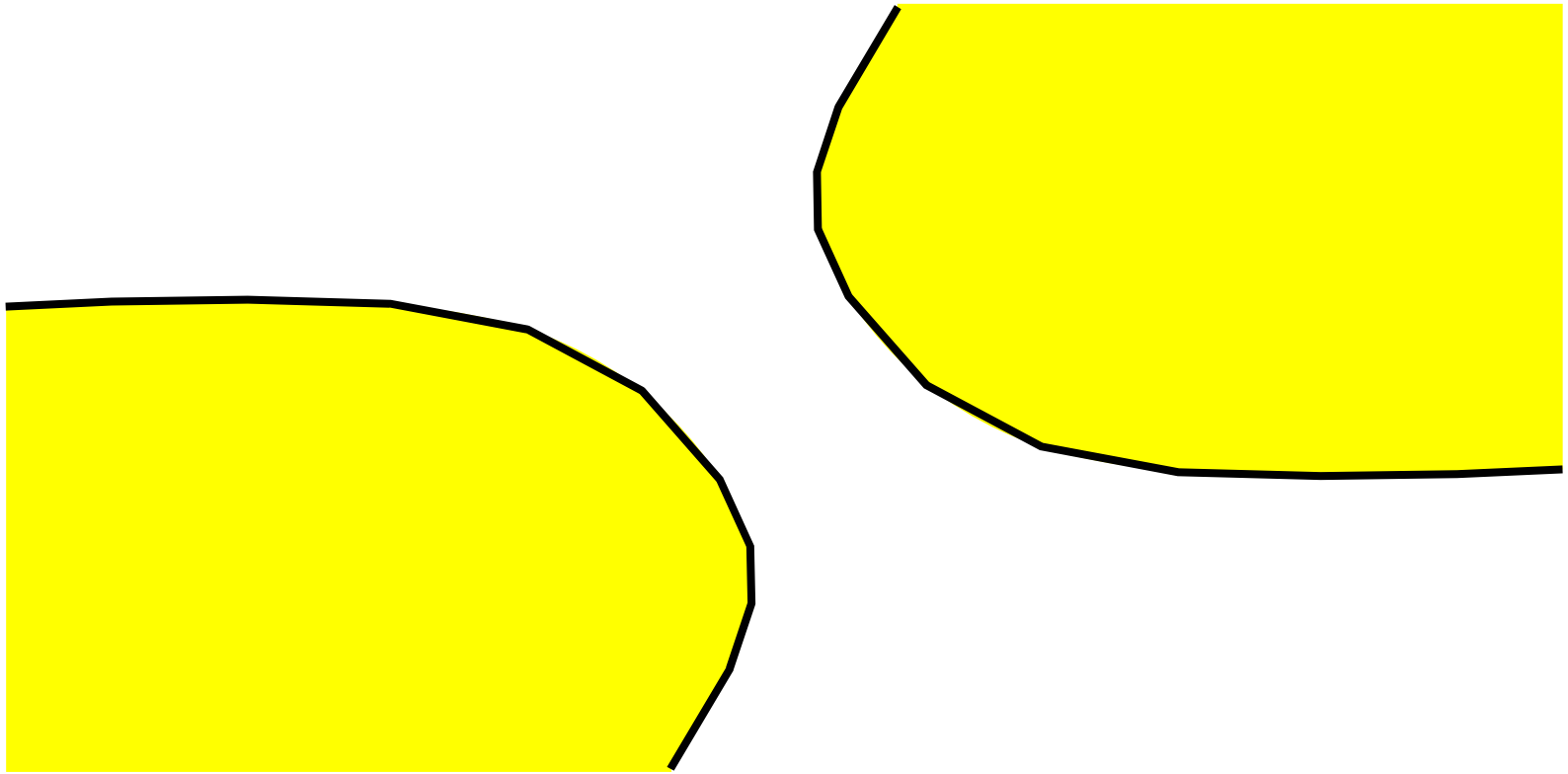
Multiple Normals

Hybrid Methods



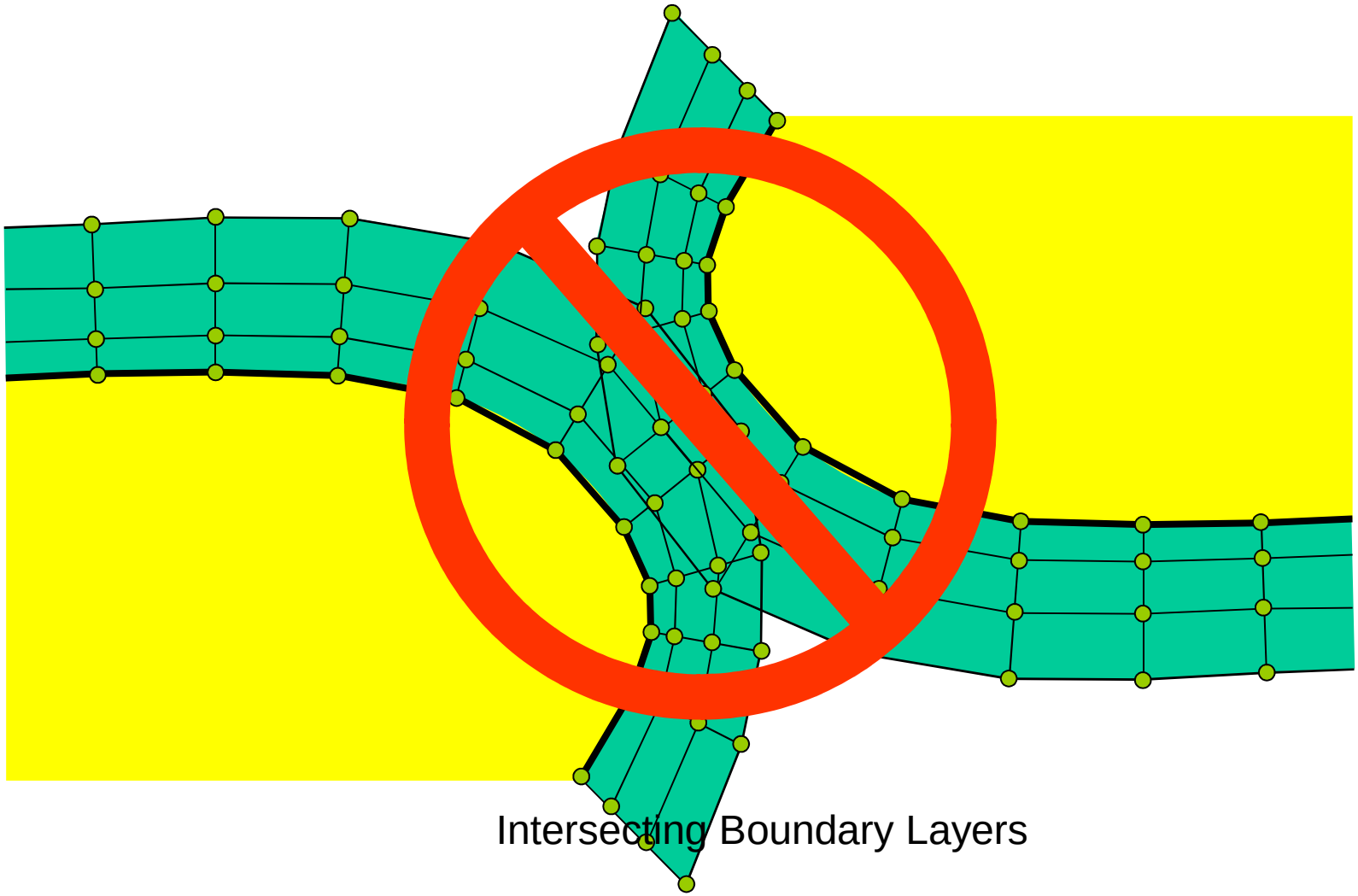
Multiple Normals

Hybrid Methods



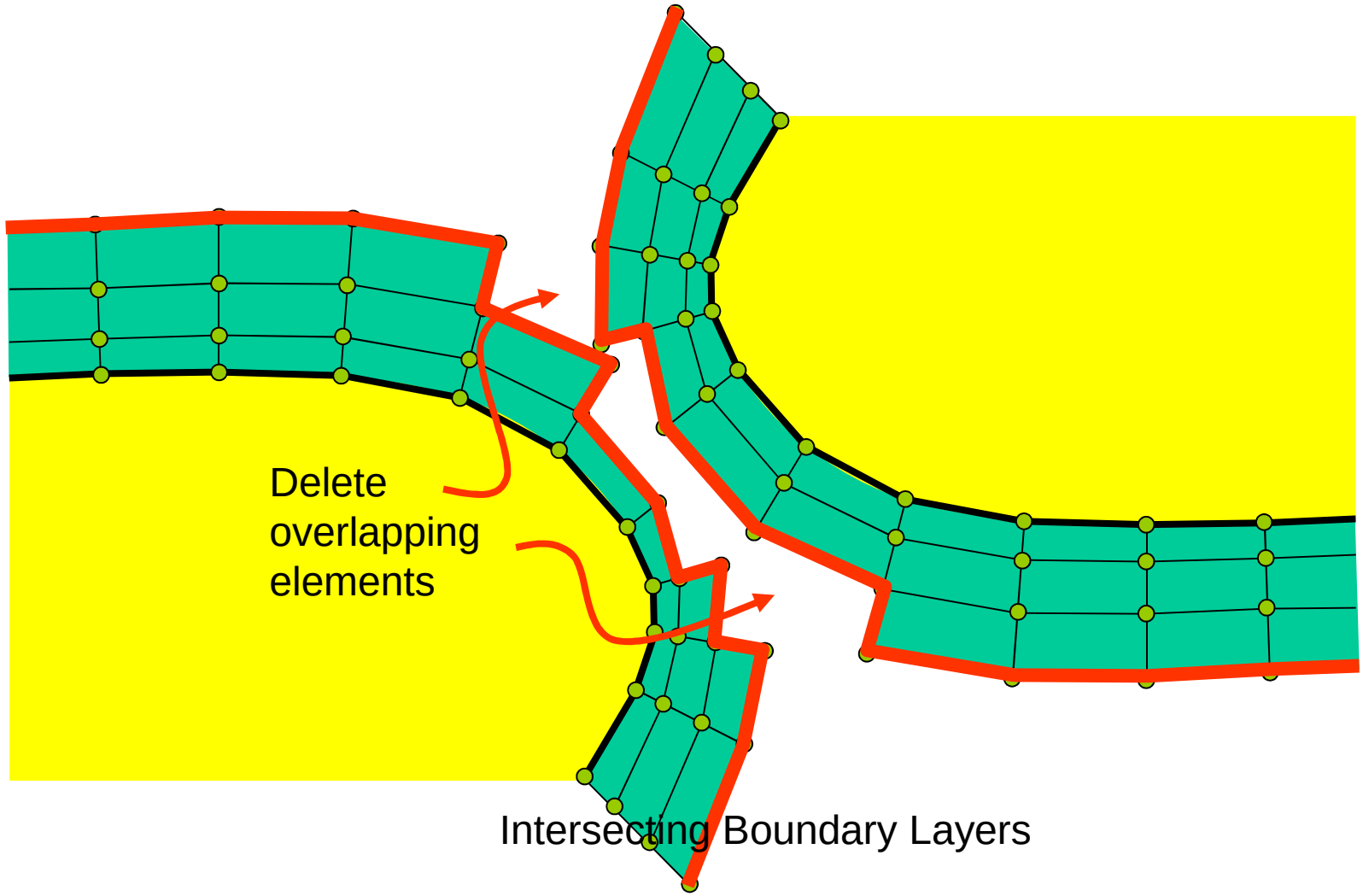
Intersecting Boundary Layers

Hybrid Methods

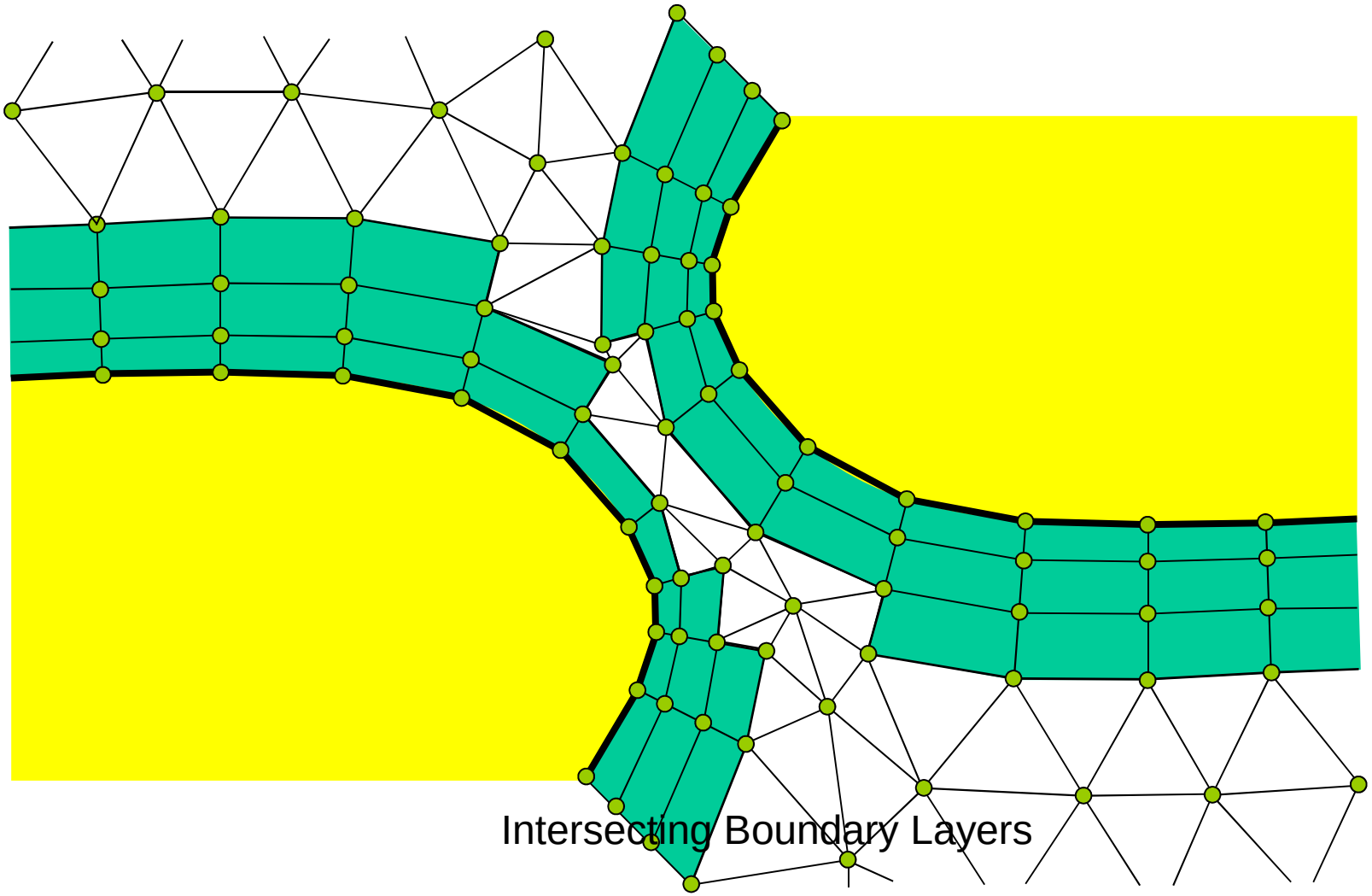


Intersecting Boundary Layers

Hybrid Methods



Hybrid Methods



Intersecting Boundary Layers

Hybrid Methods

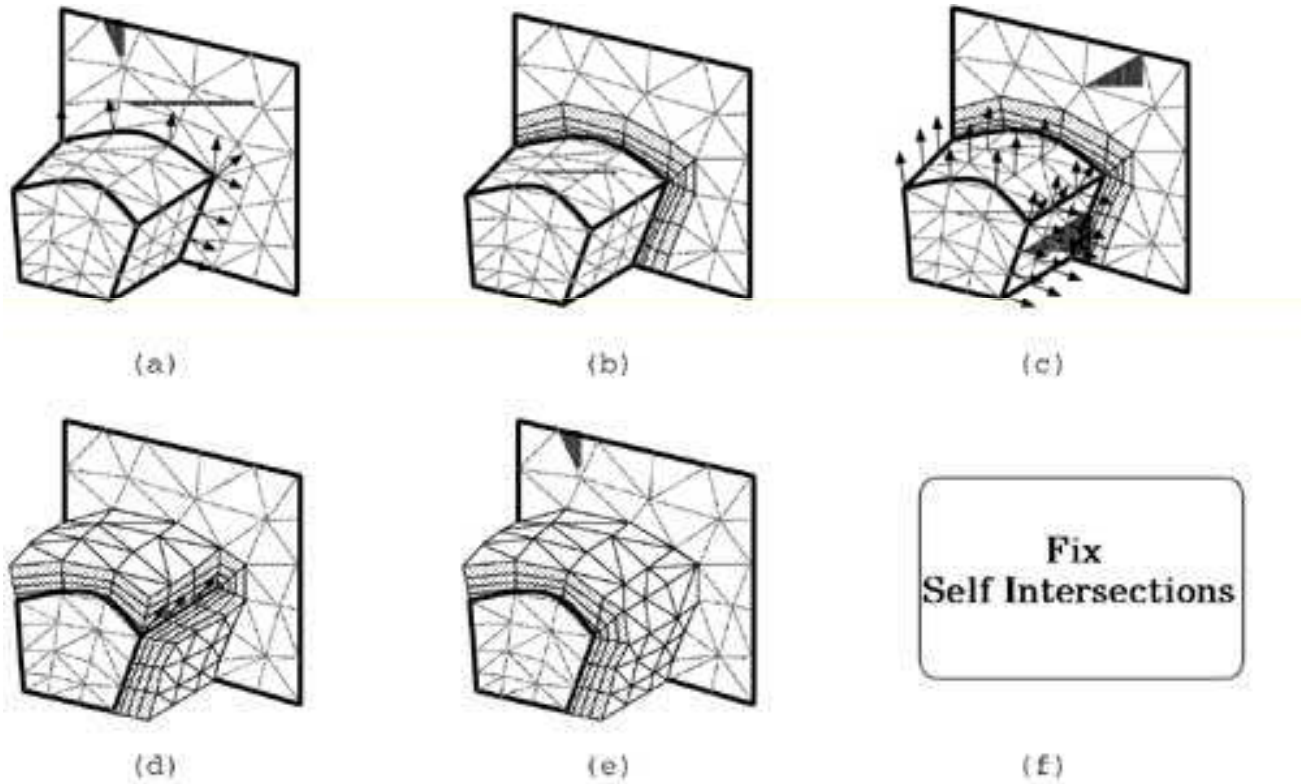
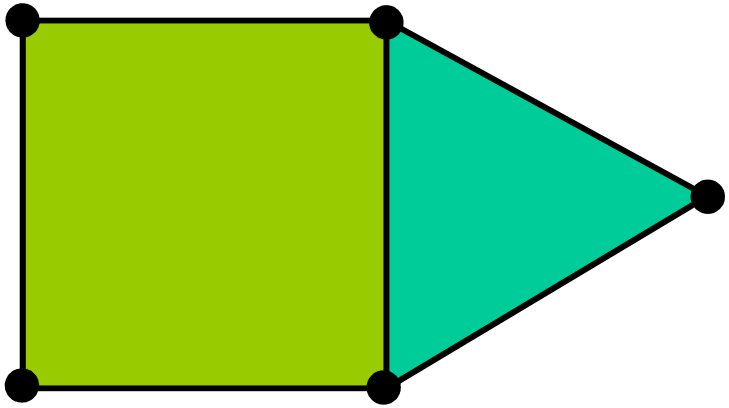


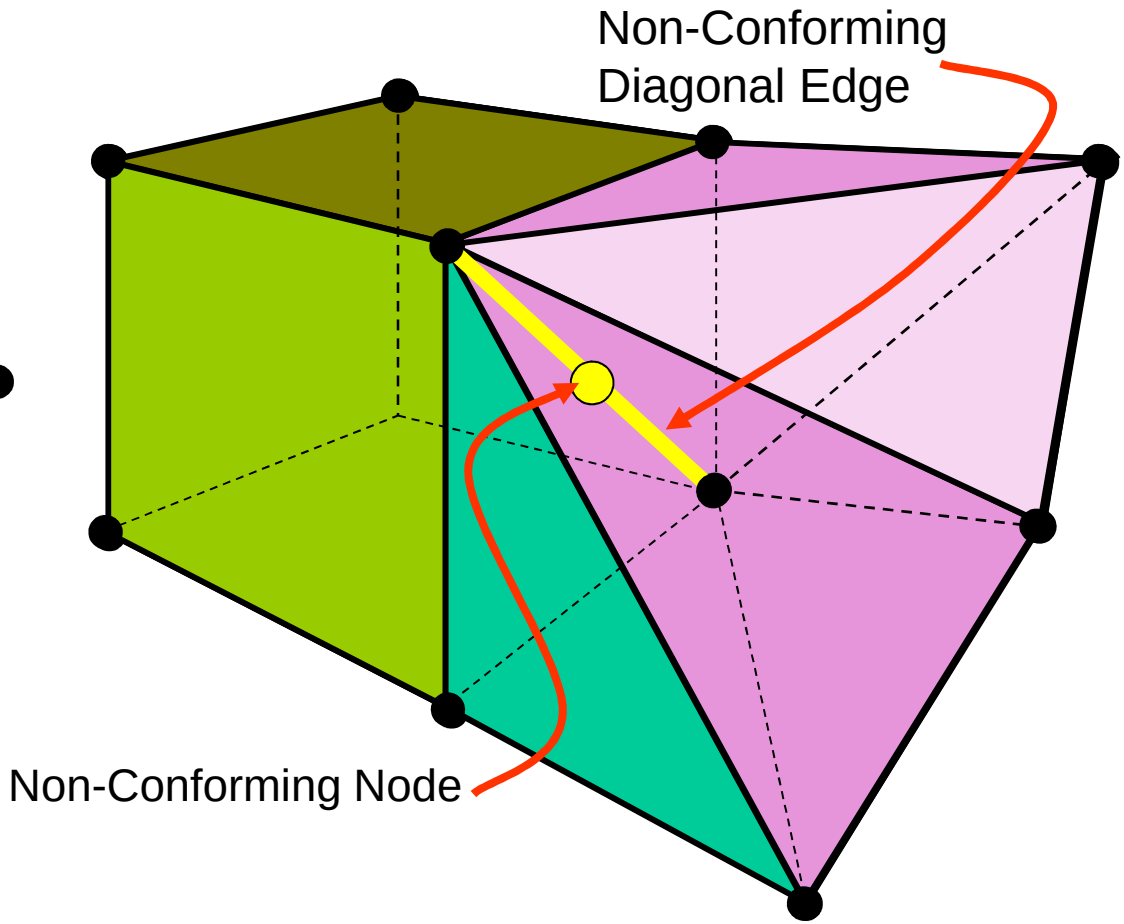
Image courtesy of SCOREC, Rensselaer Polytechnic Institute, http://www.scorec.rpi.edu/

(Garimella, Shephard, 2000)

Hex-Tet Interface

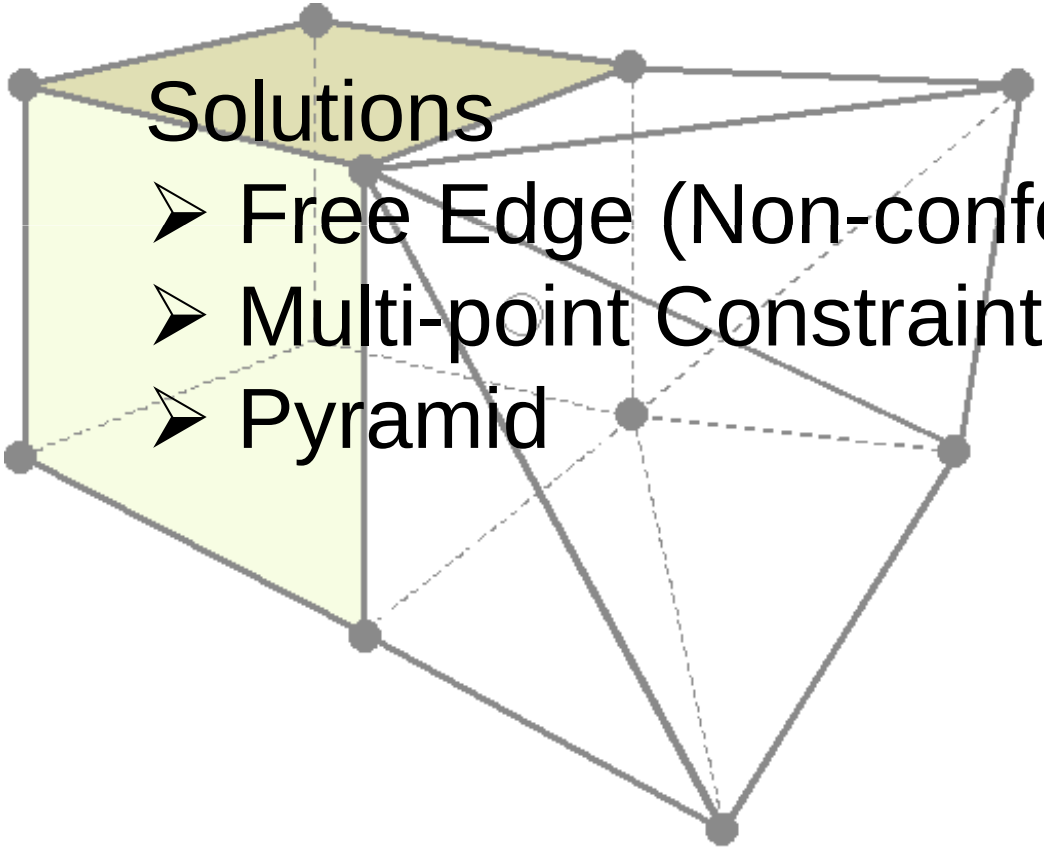


Conforming quad-triangle



Conforming hex-tet?

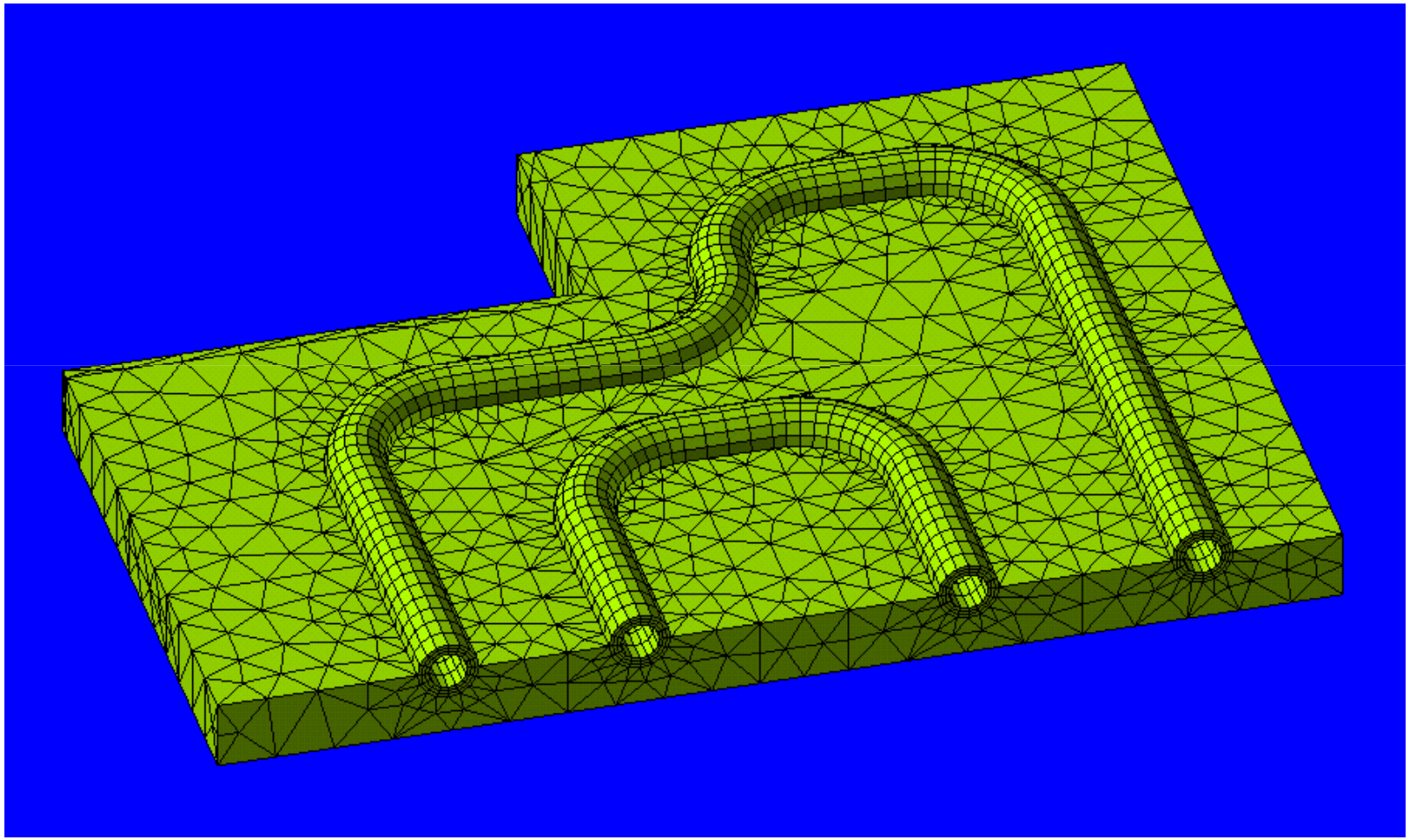
Hex-Tet Interface



Solutions

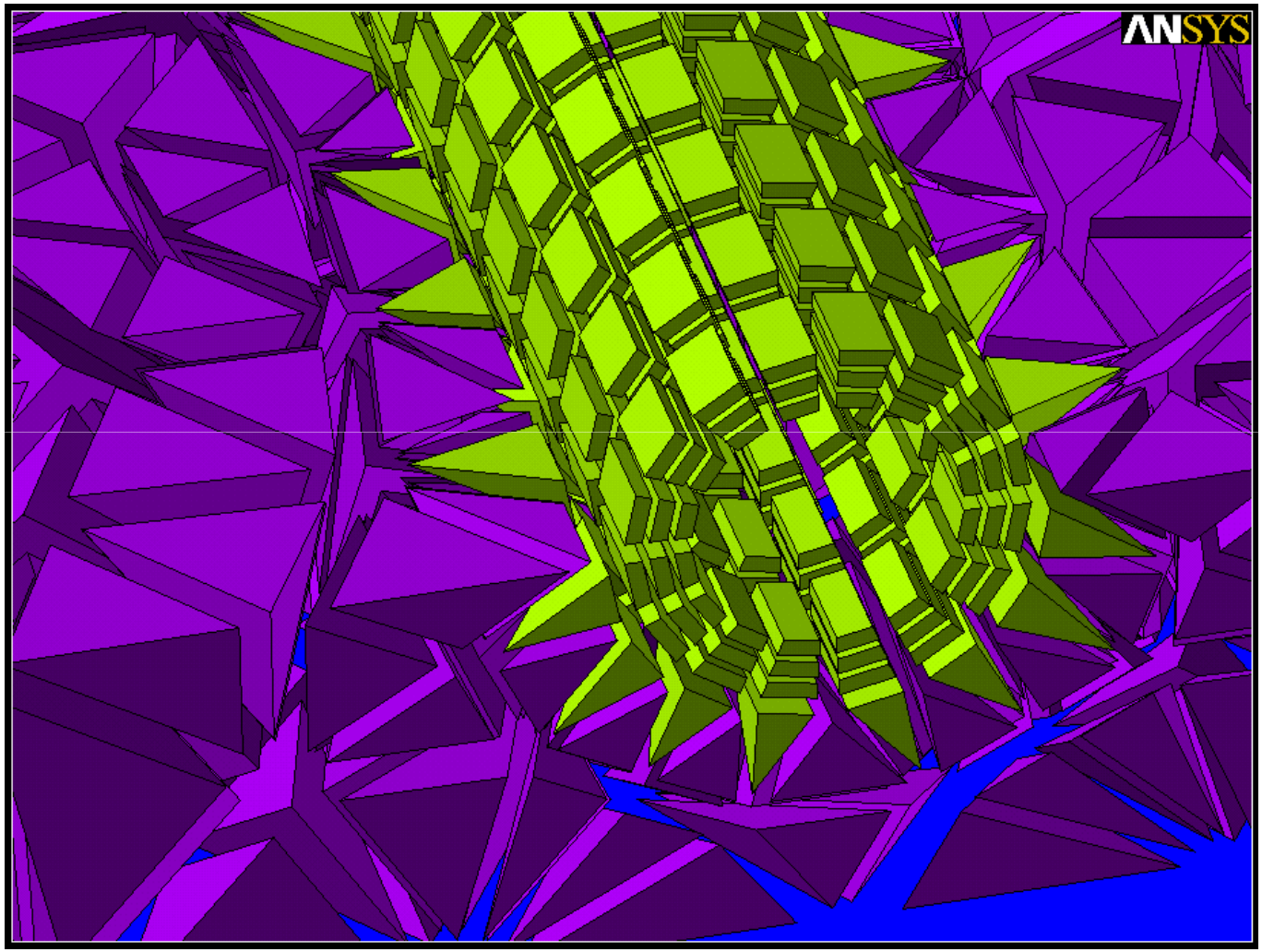
- Free Edge (Non-conforming)
- Multi-point Constraint
- Pyramid

Hex-Tet Interface



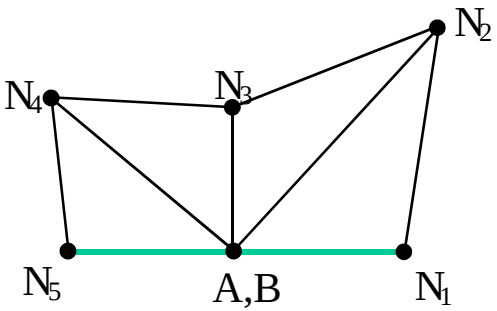
Heat sink meshed with hexes, tets and pyramids

Hex-Tet Interface

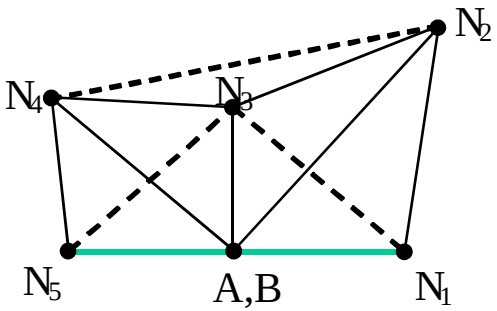


Pyramid Elements for maintaining compatibility between hex and tet elements (Owen,00)

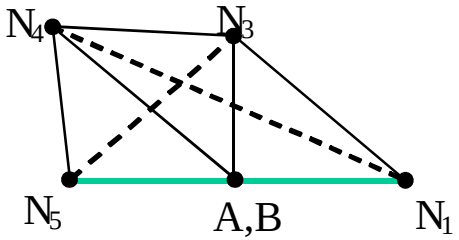
Hex-Tet Interface



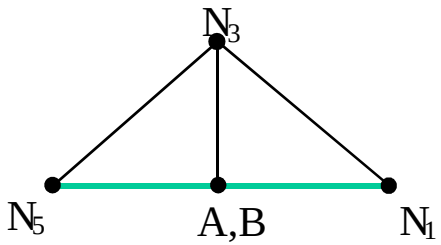
(a)



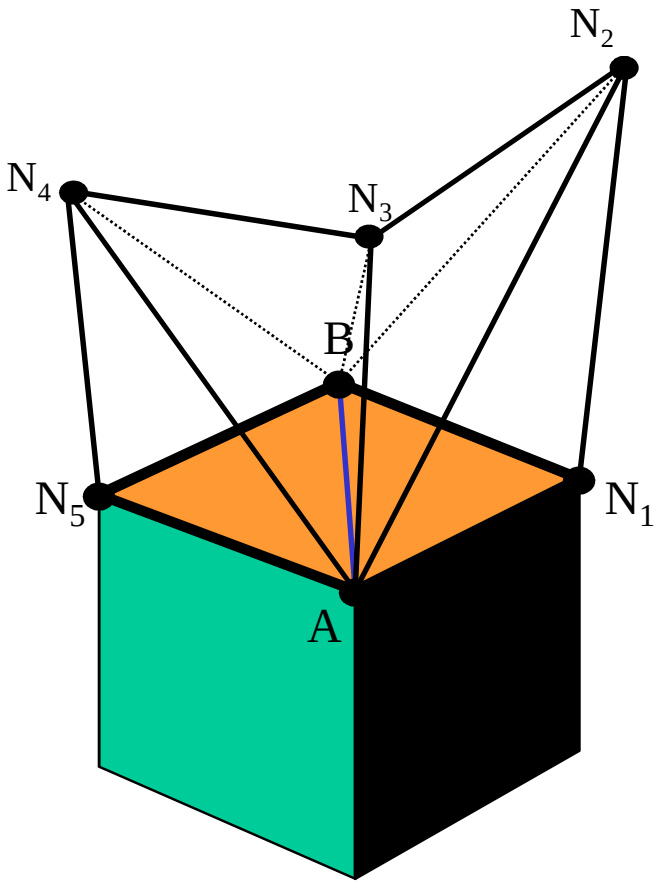
(b)



(c)



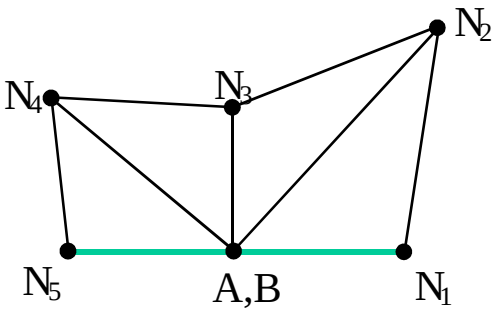
(d)



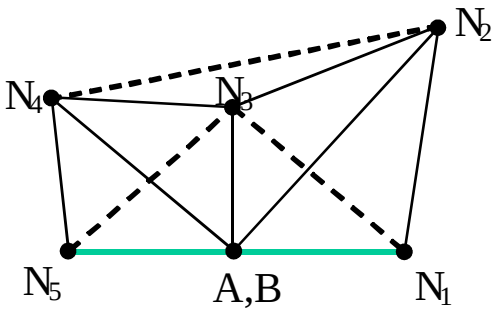
Tetrahedral transformations to form Pyramids

- Use 2-3 swaps to obtain 2 tets at diagonal
- combine 2 tets to form pyramid

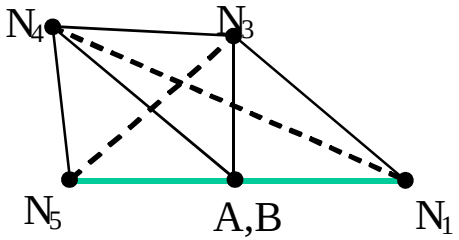
Hex-Tet Interface



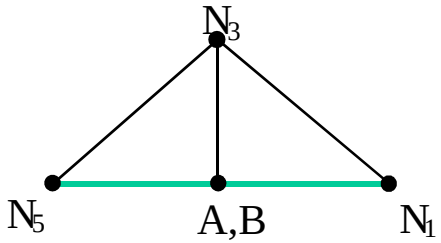
(a)



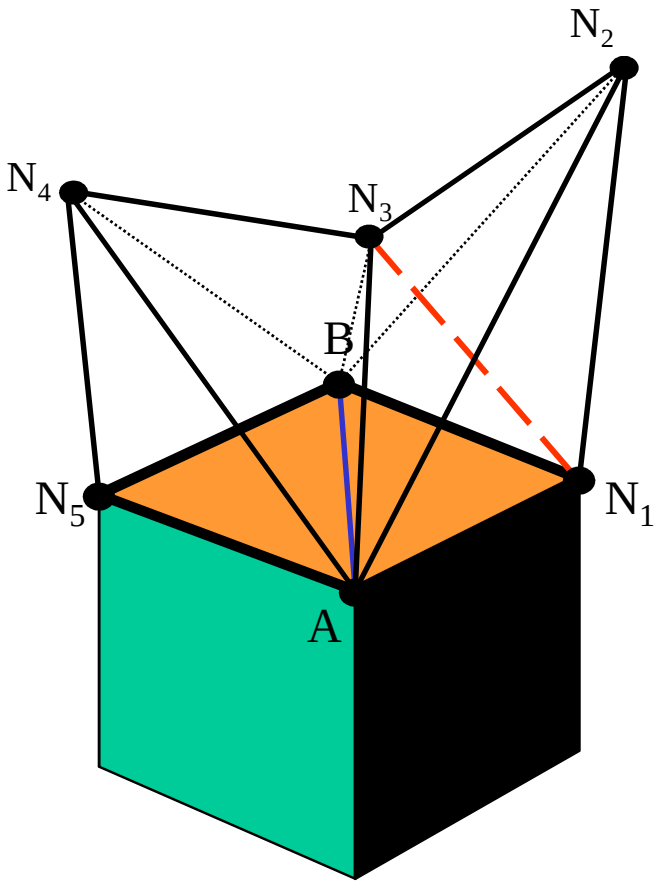
(b)



(c)



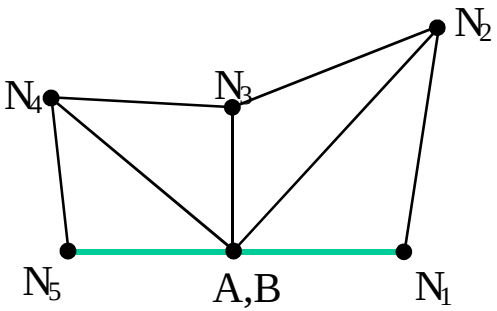
(d)



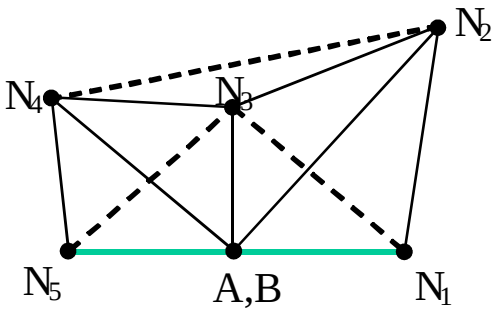
Tetrahedral transformations to form Pyramids

- Use 2-3 swaps to obtain 2 tets at diagonal
- combine 2 tets to form pyramid

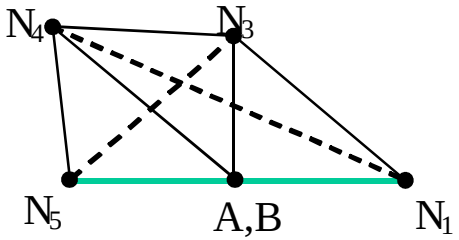
Hex-Tet Interface



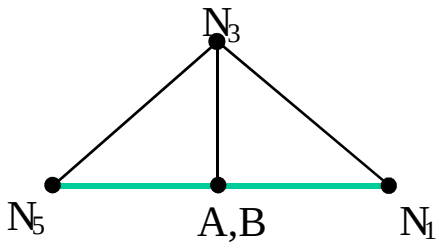
(a)



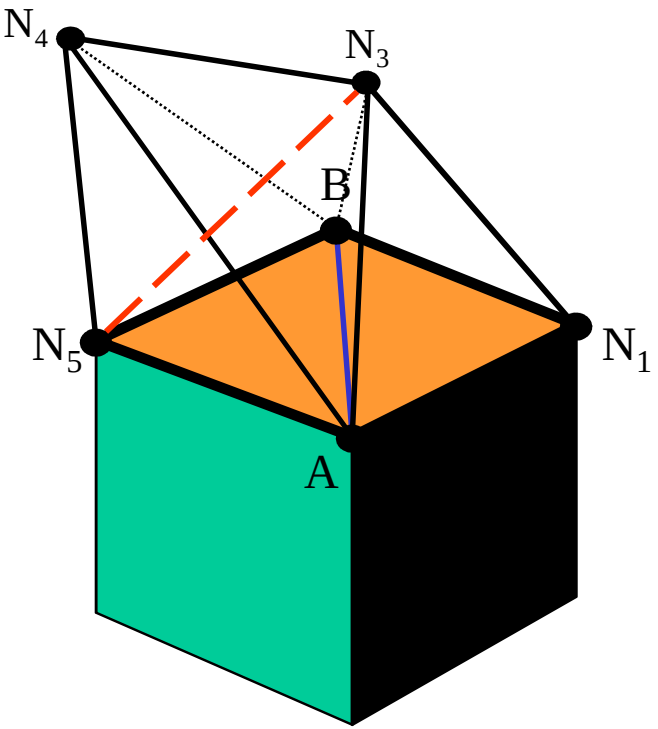
(b)



(c)



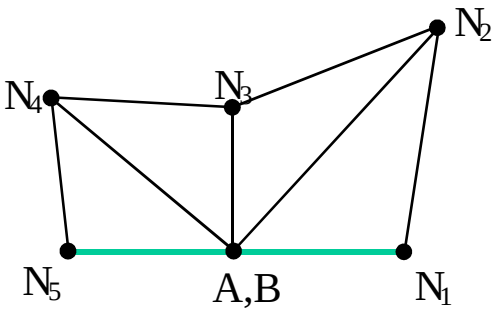
(d)



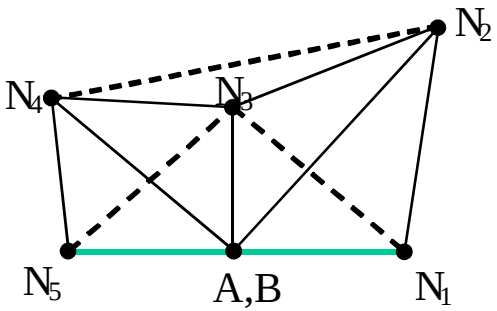
Tetrahedral transformations to form Pyramids

- Use 2-3 swaps to obtain 2 tets at diagonal
- combine 2 tets to form pyramid

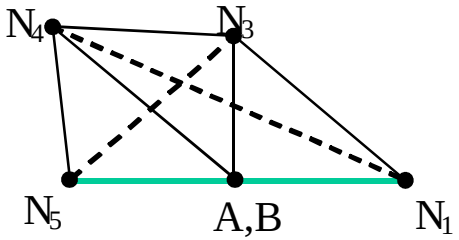
Hex-Tet Interface



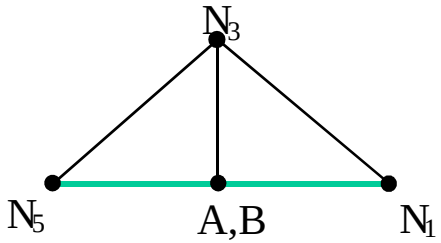
(a)



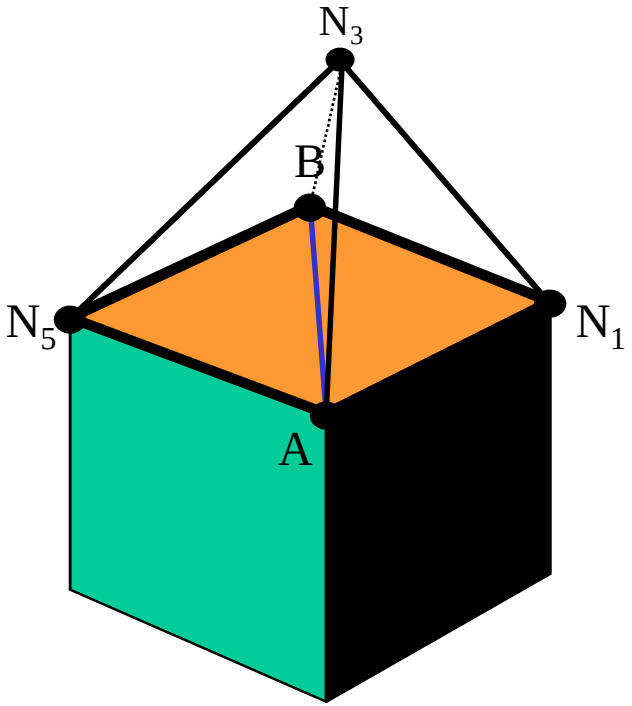
(b)



(c)



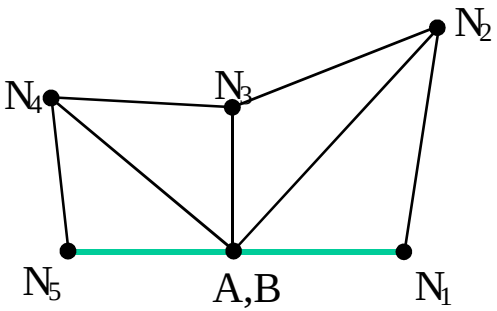
(d)



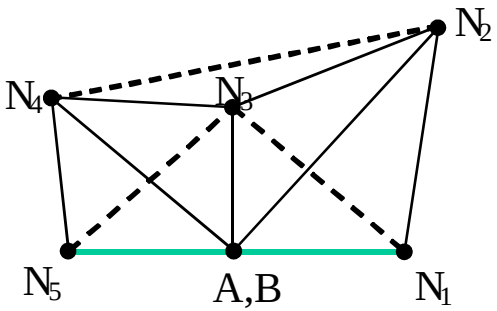
Tetrahedral transformations to form Pyramids

- Use 2-3 swaps to obtain 2 tets at diagonal
- combine 2 tets to form pyramid

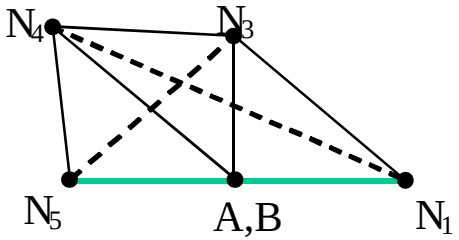
Hex-Tet Interface



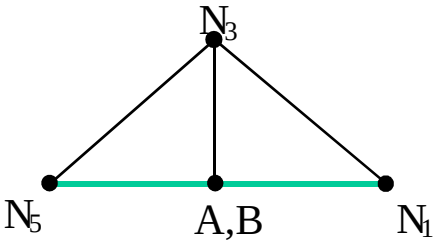
(a)



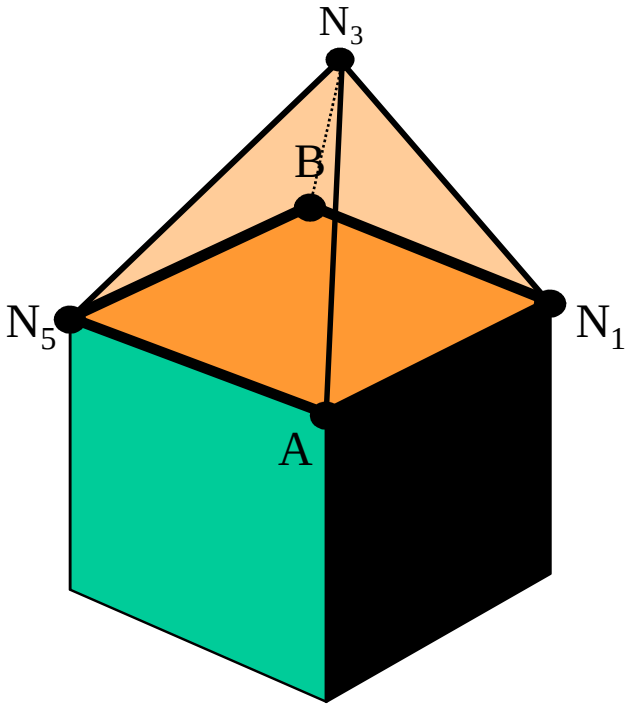
(b)



(c)



(d)

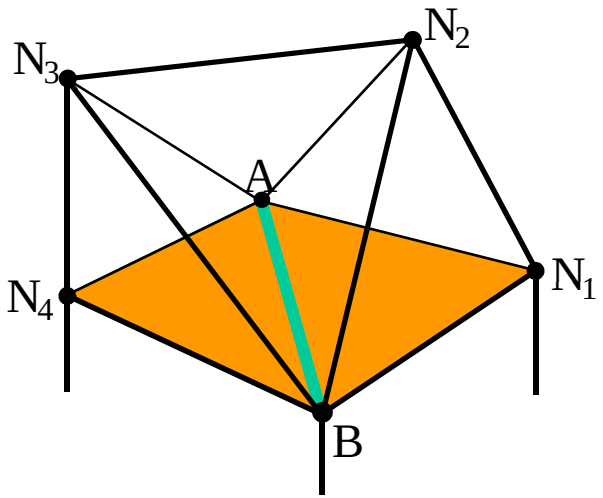


Tetrahedral transformations to form Pyramids

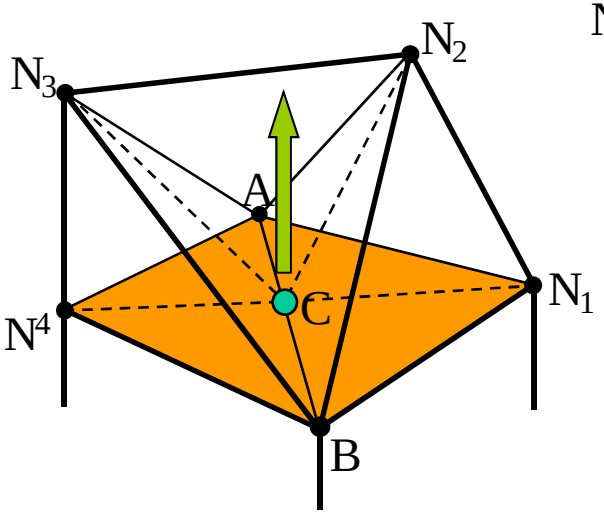
- Use 2-3 swaps to obtain 2 tets at diagonal
- combine 2 tets to form pyramid

Hex-Tet Interface

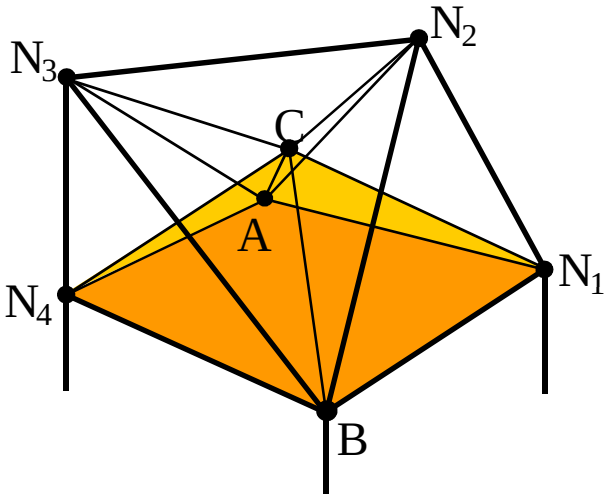
Pyramid Open Method



Non-Conforming Condition:
Tets at quad diagonal A-B



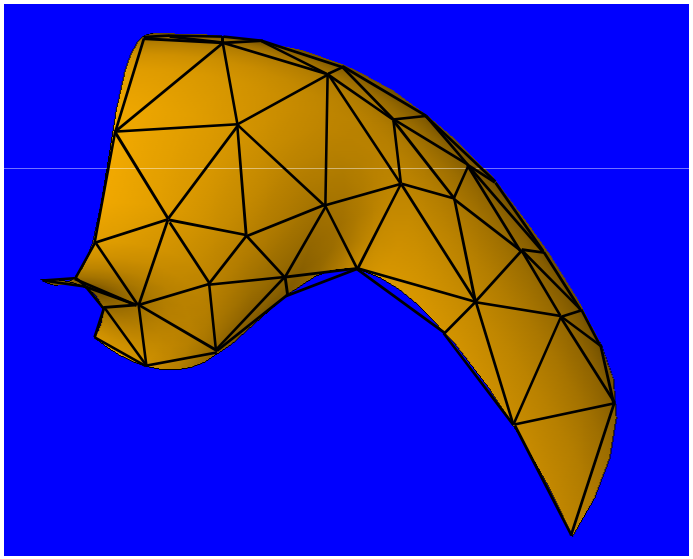
- Insert C at midpoint AB:
- Split all tets at edge AB



- Move C to average $N_1, N_2 \ldots N_n$
- Create New Pyramid A, N_n, B, N_1, C

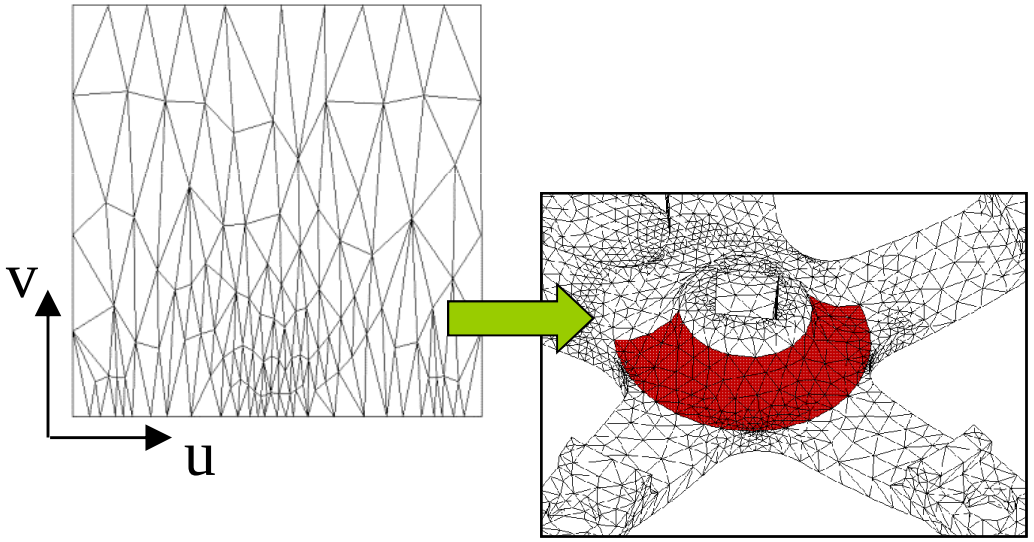
Surface Meshing

Direct 3D Meshing



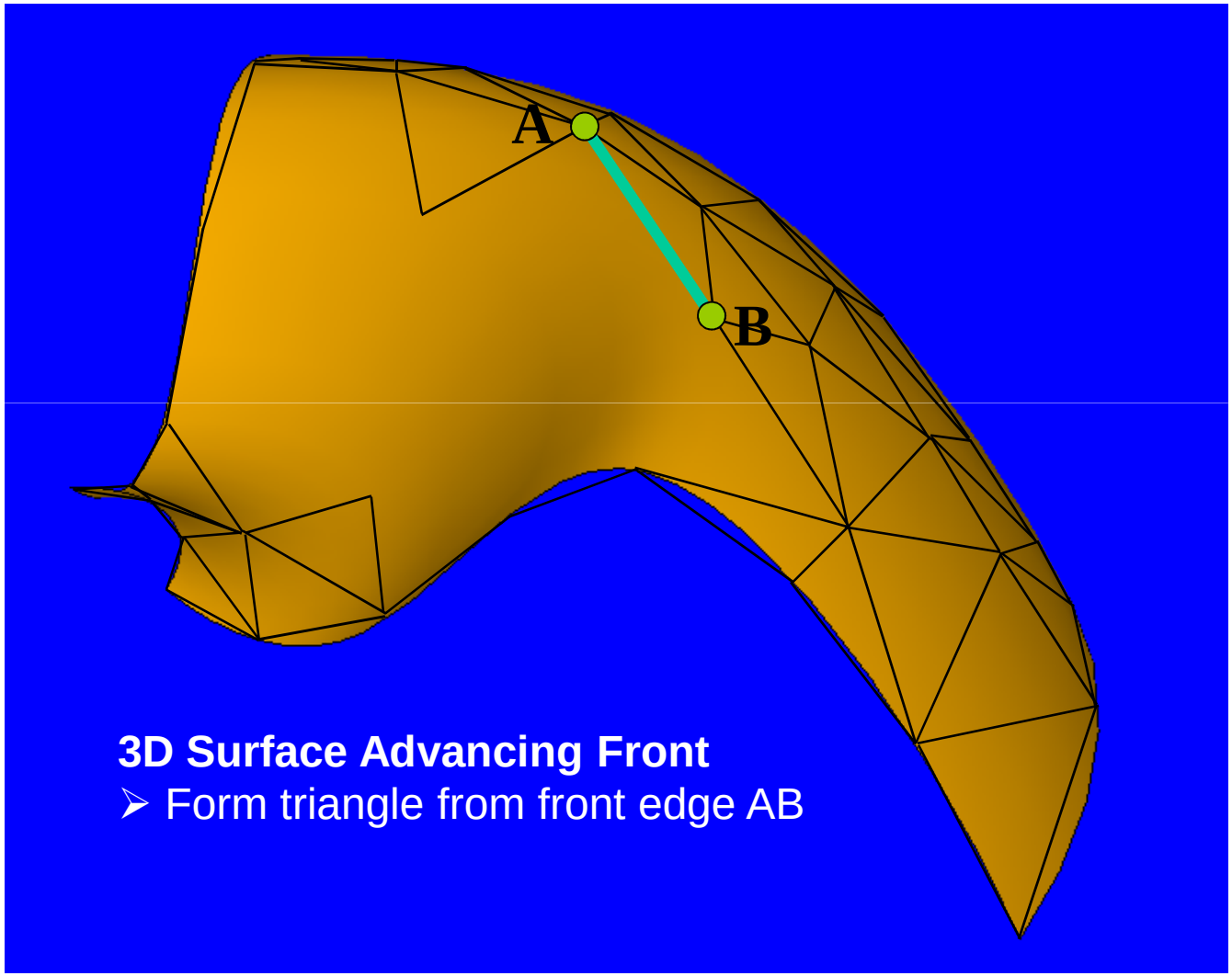
- Elements formed in 3D using actual x-y-z representation of surface

Parametric Space Meshing

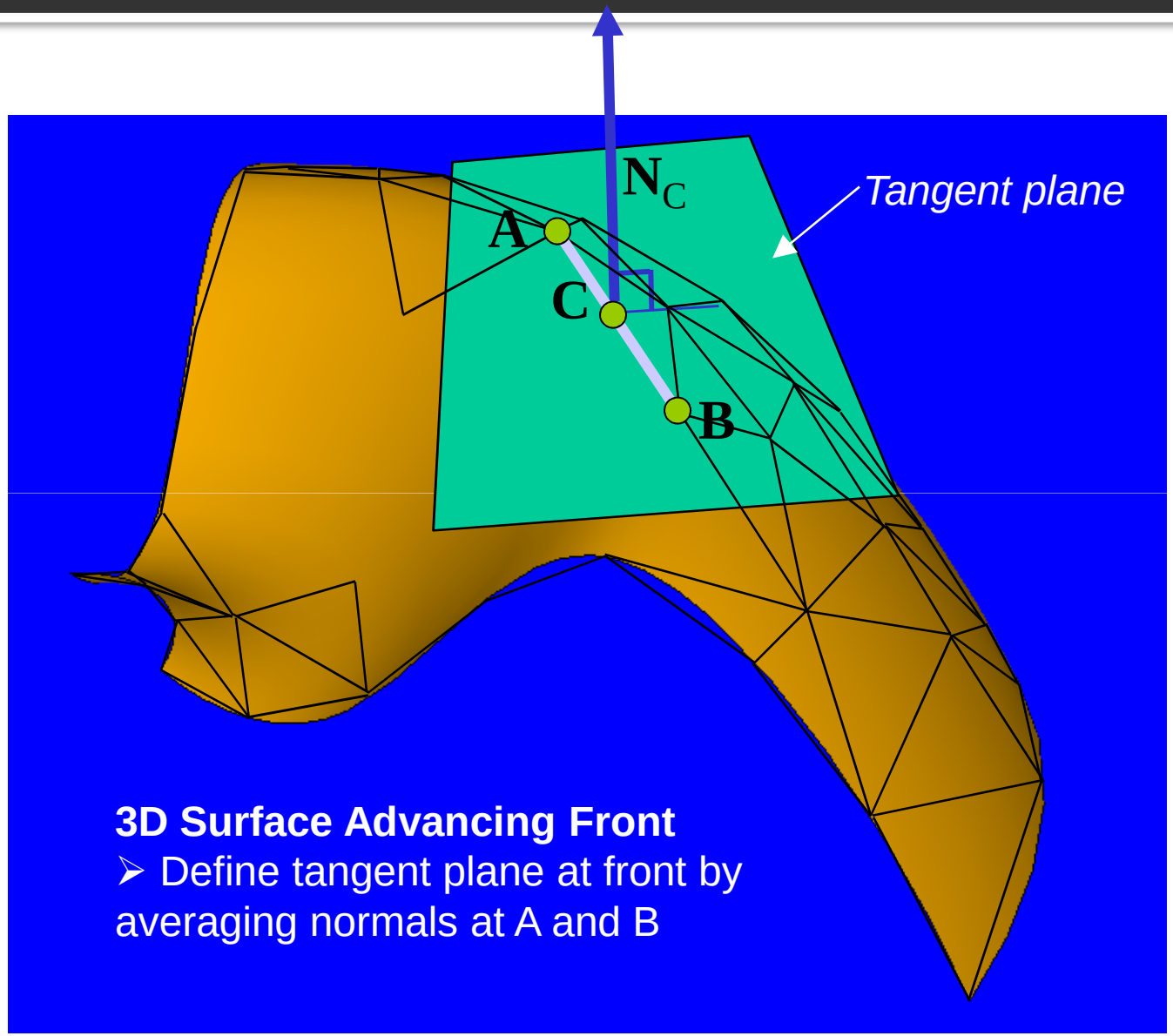


- Elements formed in 2D using parametric representation of surface
- Node locations later mapped to 3D

Surface Meshing



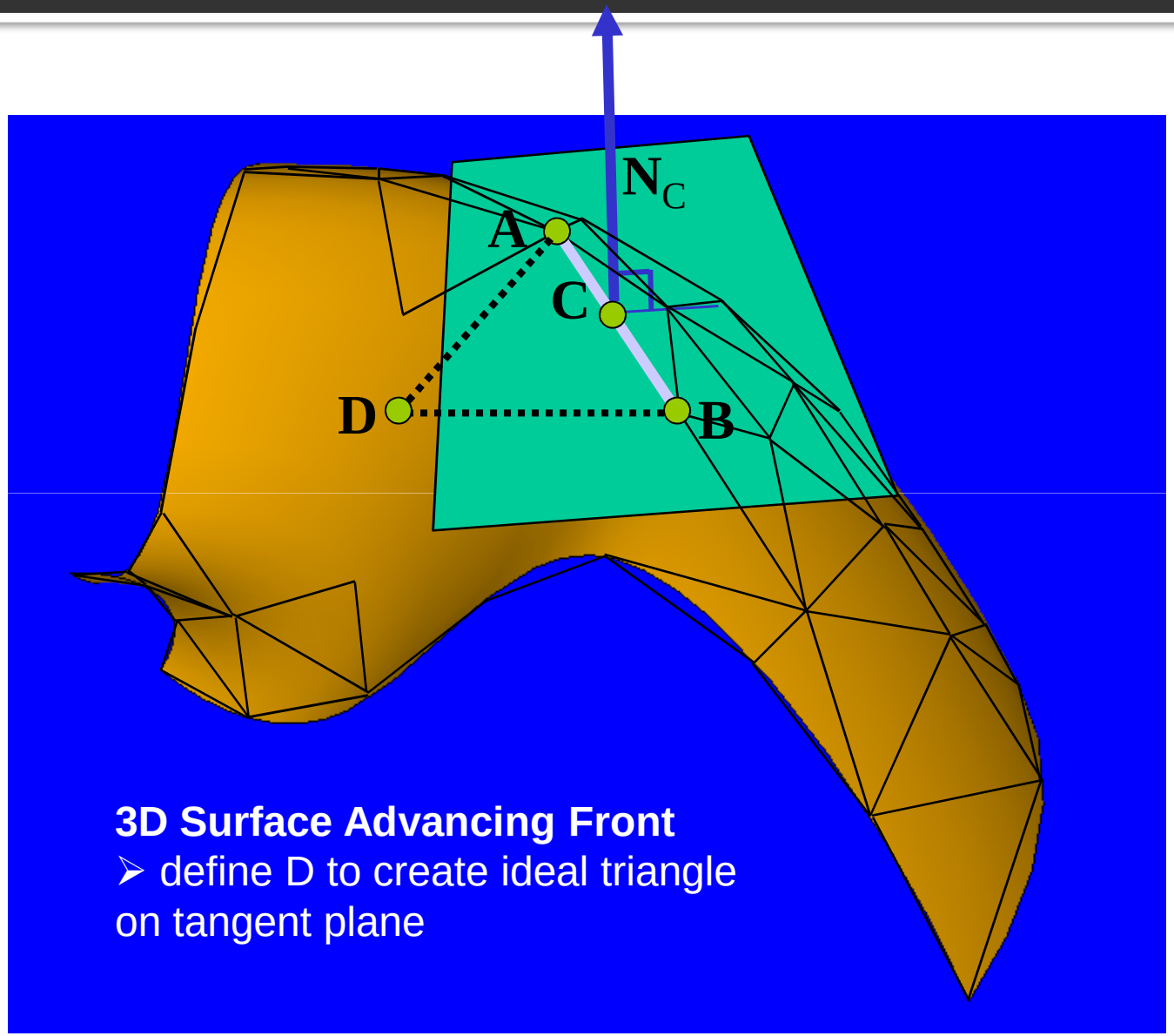
Surface Meshing



3D Surface Advancing Front

➢ Define tangent plane at front by averaging normals at A and B

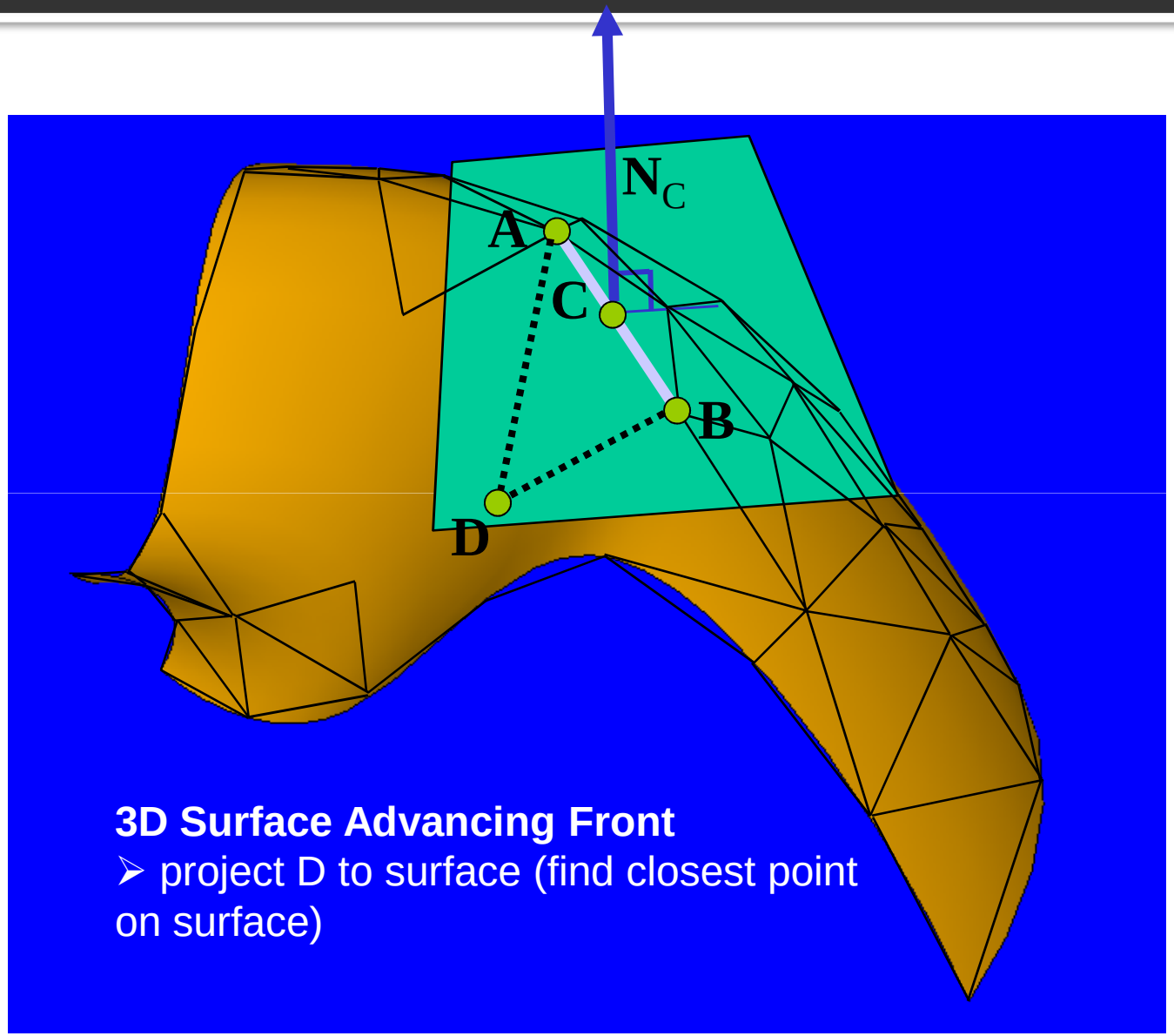
Surface Meshing



3D Surface Advancing Front

➢ define D to create ideal triangle on tangent plane

Surface Meshing



3D Surface Advancing Front

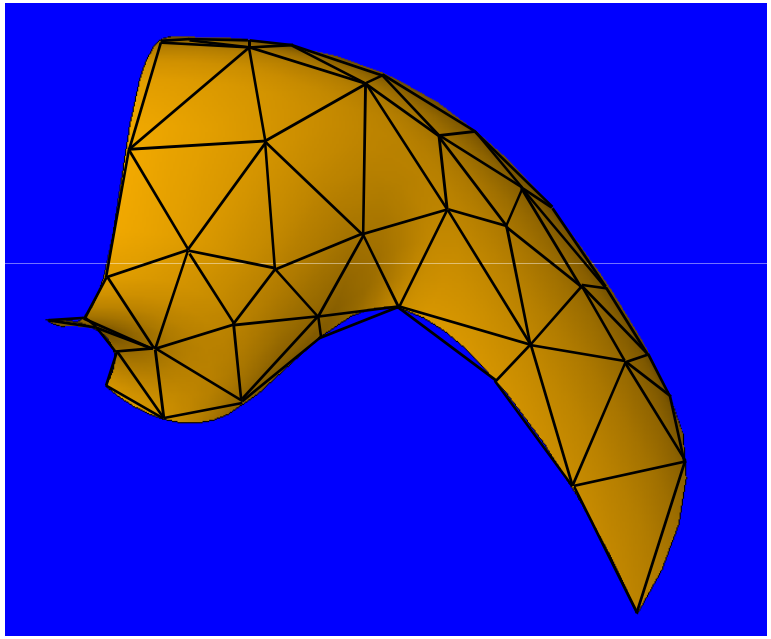
➢ project D to surface (find closest point on surface)

Surface Meshing

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)

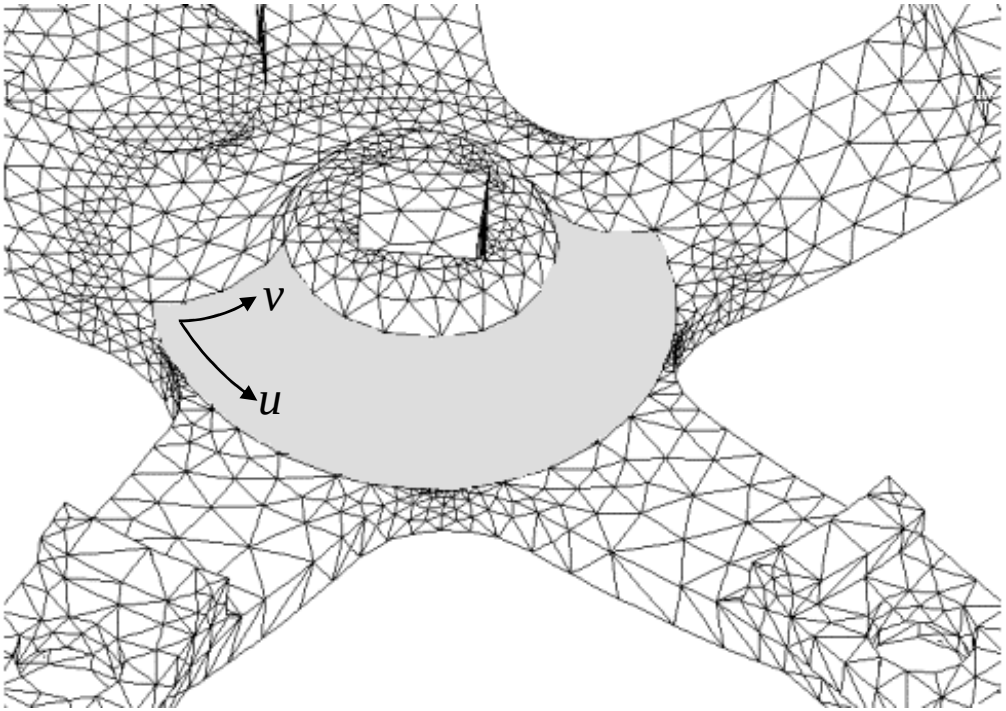
(Lo,96;97); (Cass,96)

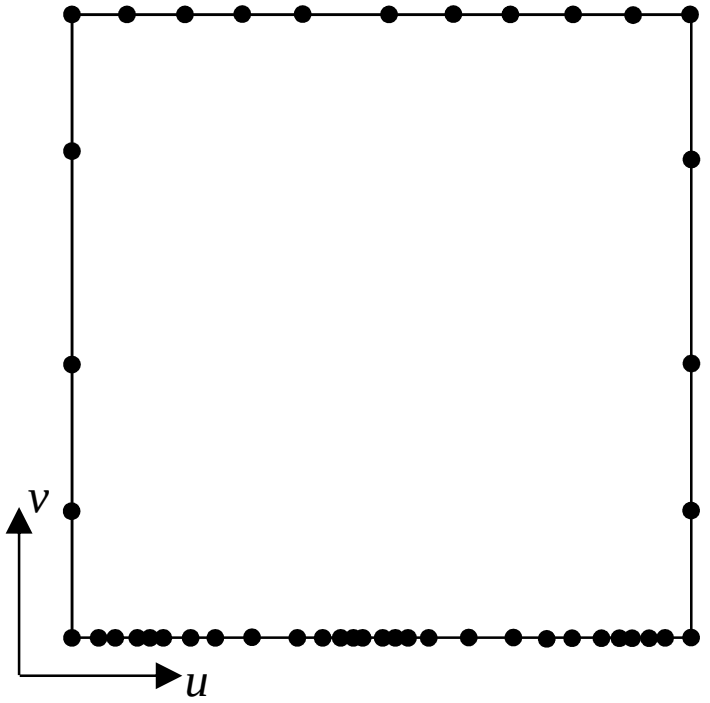


Surface Meshing

Parametric Space Mesh Generation

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

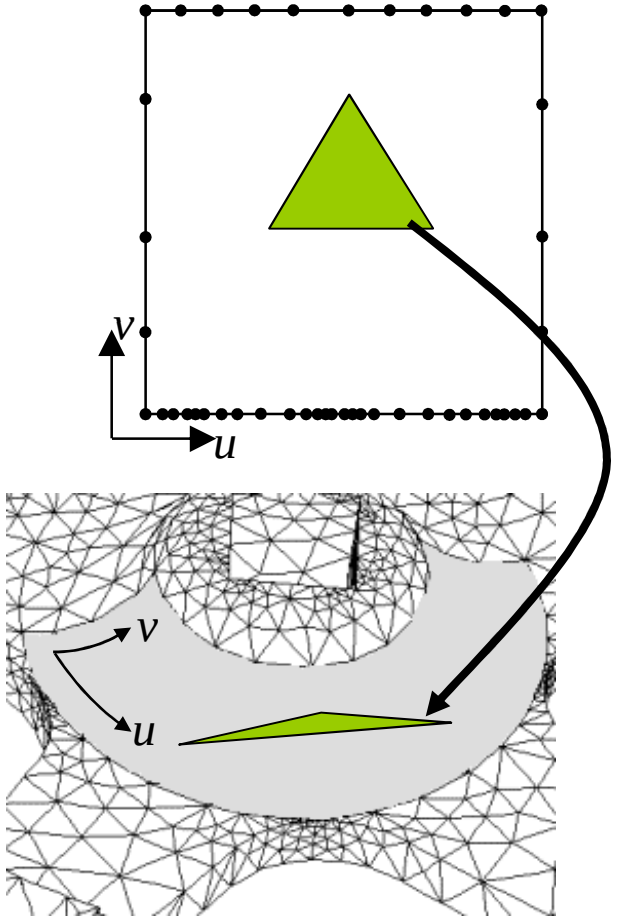




Surface Meshing

Parametric Space Mesh Generation

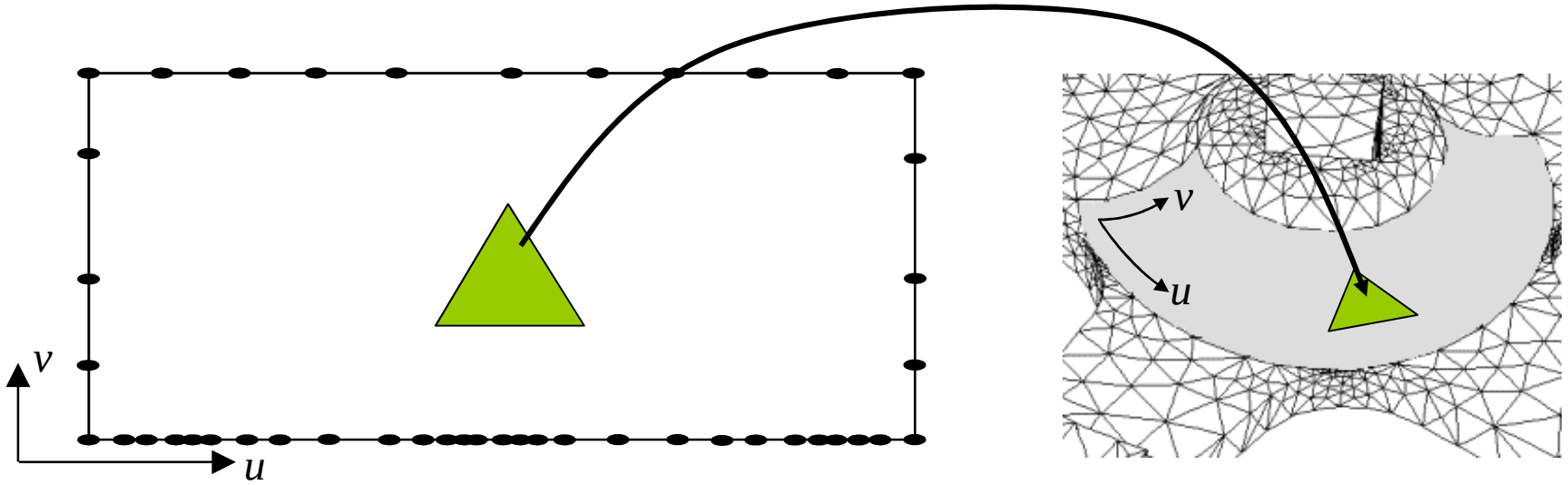
- Isotropic: Target element shapes are equilateral triangles
- Equilateral elements in parametric space may be distorted when mapped to 3D space.
- If parametric space resembles 3D space without too much distortion from *u*-*v* space to *x*-*y*-*z* space, then isotropic methods can be used.



Surface Meshing

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

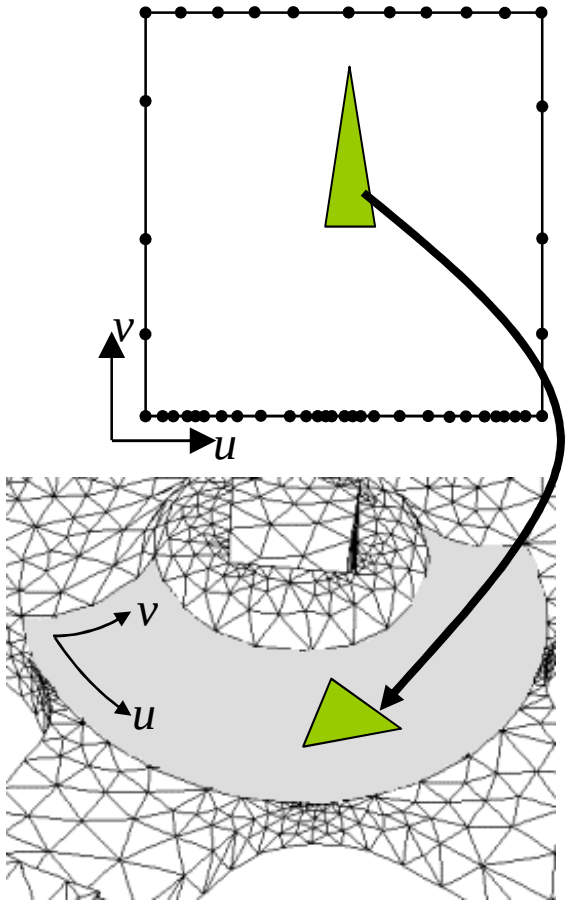


Warped parametric space

Surface Meshing

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



Parametric Space Mesh Generation

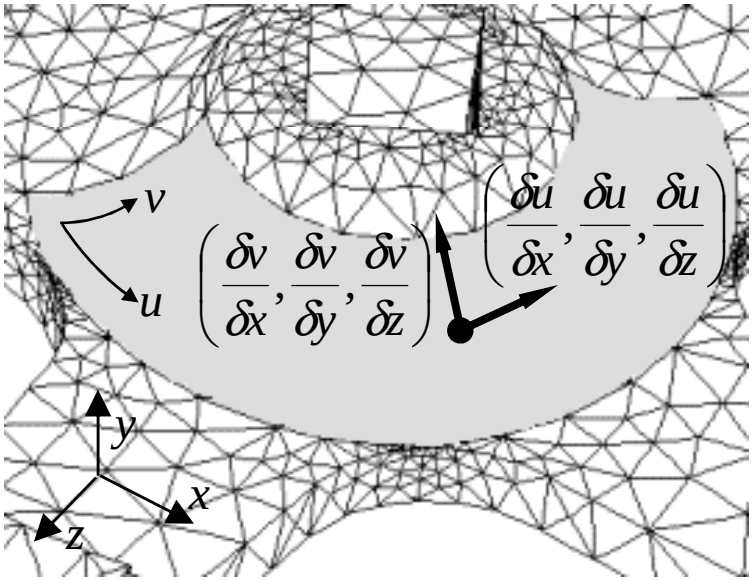
- Stretching is based on field of surface derivatives

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

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

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

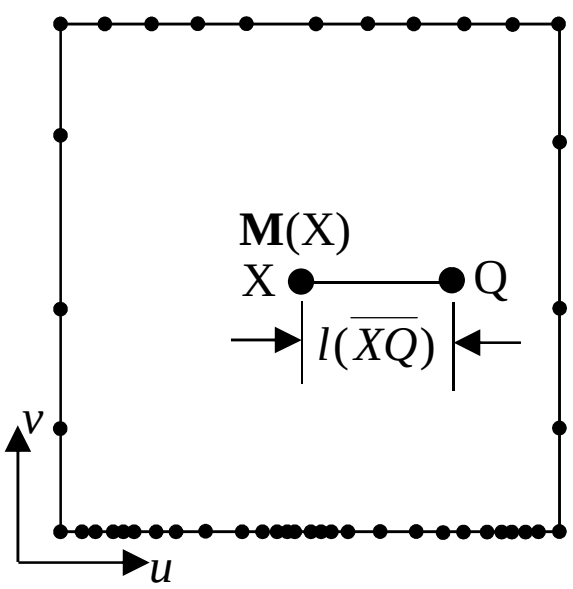
$$E = \Delta\mathbf{u} \cdot \Delta\mathbf{u} \qquad F = \Delta\mathbf{u} \cdot \Delta\mathbf{v} \qquad G = \Delta\mathbf{v} \cdot \Delta\mathbf{v}$$

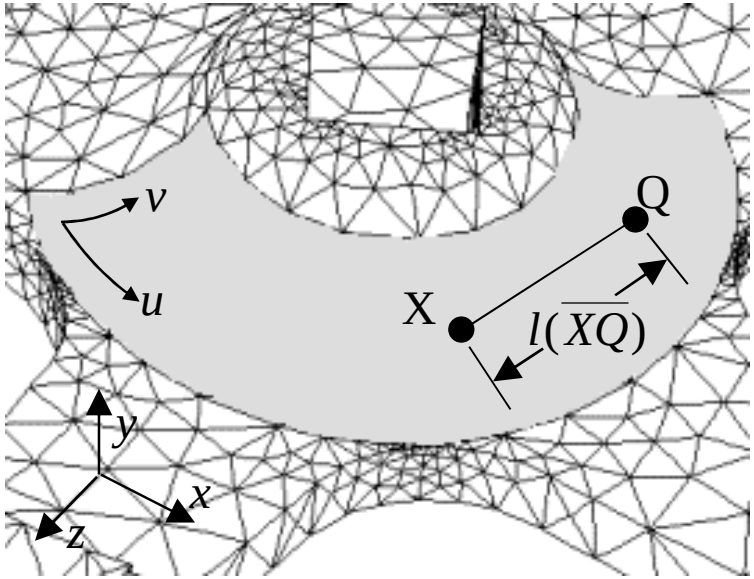


Parametric Space Mesh Generation

- 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:

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

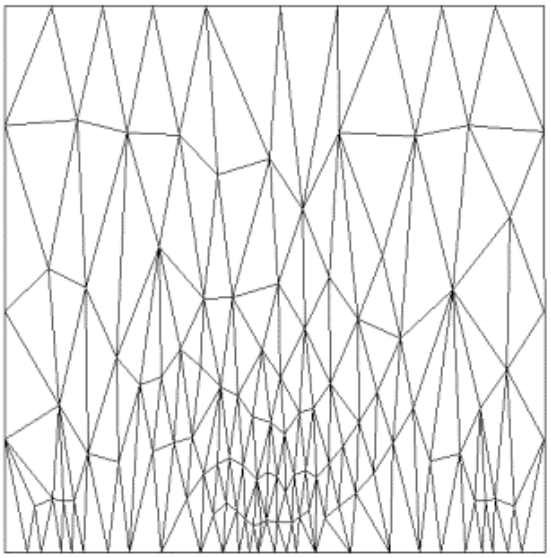


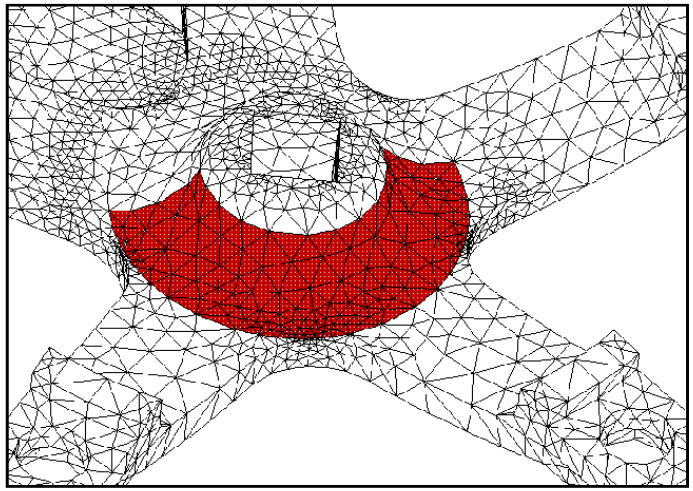


Surface Meshing

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)

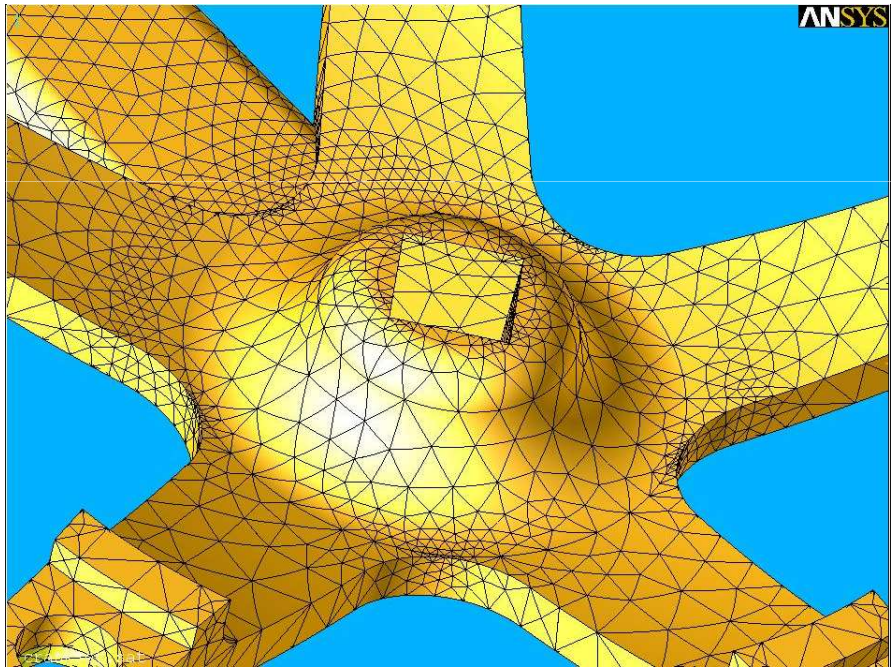




Surface Meshing

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)



Algorithm Characteristics

- 1. Conforming Mesh
  - Elements conform to a prescribed surface mesh
- 2. Boundary Sensitive
  - Rows/layers of elements roughly conform to the contours of the boundary
- 3. Orientation Insensitive
  - Rotating/Scaling geometry will not change the resulting mesh
- 4. Regular Node Valence
  - Inherent in the algorithm is the ability to maintain (nearly) the same number of elements adjacent each node)
- 5. Arbitrary Geometry
  - The algorithm does not rely on a specific class/shape of geometry
- 6. Commercial Viability (Robustness/Speed)
  - The algorithm has been used in a commercial setting

Algorithm Characteristics

		Tris			Tets				Quads								Hexes									
		Quadtree	Delaunay	Adv. Front	Octree	Delaunay	Adv. Front		Mapped	Sub-map	Tri Split	Tri Merge	Q-Morph	Grid-Based	Medial Axis	Paving	Mapped	Sub-map	Sweeping	Tet Split	Tet Merge	H-Morph	Grid-Based	Medial Surf.	Plastering	Whisker W.
1	Conforming Mesh		●	●		●	●		●	●	●	●	●			●	●	●	●			●			●	●
2	Boundary Sensitive			●			●		●	●			●		●	●	●	●	●			●		●	●	●
3	Orientation Insensitive		●	●		●	●		●	●	●	●	●		●	●	●	●	●	●	●	●		●	●	●
4	Regular Node Valence			●			●		●	●			●	●	●	●	●	●	●			●	●	●	●	●
5	Arbitrary Geometry	●	●	●	●	●	●				●	●	●	●	●	●				●	●	●	●	●	●	●
6	Commercially Viability	●	●	●	●	●	●		●	●		●	●	●	●	●	●	●	●		●		●			

References

Blacker, Ted D. and R. J. Myers,. "Seams and Wedges in Plastering: A 3D Hexahedral Mesh Generation Algorithm," *Engineering With Computers*, **2**, 83-93 (1993)

Blacker, Ted D., "The Cooper Tool", *Proceedings, 5th International Meshing Roundtable*, 13-29 (1996)

Blacker, Ted D., and Michael B. Stephenson. "Paving: A New Approach to Automated Quadrilateral Mesh Generation", *International Journal for Numerical Methods in Engineering*, **32**, 811-847 (1991)

Canann, Scott A., Joseph R. Tristano and Matthew L. Staten, "An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral and quad-dominant meshes," *Proc. 7th International Meshing Roundtable*, pp.479-494 (1998)

Canann, S. A., S. N. Muthukrishnan, and R. Phillips, "Topological refinement procedures for triangular finite element meshes," *Engineering with Computers*, **12** (3/4), 243-255 (1996)

Cass, Roger J, Steven E. Benzley, Ray J. Meyers and Ted D. Blacker. "Generalized 3-D Paving: An Automated Quadrilateral Surface Mesh Generation Algorithm", *International Journal for Numerical Methods in Engineering*, **39**, 1475-1489 (1996)

Chew, Paul L., "Guaranteed-Quality Triangular Meshes", *TR 89-983, Department of Computer Science, Cornell University, Ithaca, NY,* (1989)

Cook, W.A., and W.R. Oakes, "Mapping Methods for Generating Three-Dimensional Meshes", *Computers in Mechanical Engineering*, August, 67-72 (1982)

CUBIT Mesh Generation Toolkit, URL:http://endo.sandia.gov/cubit (2001)

Edelsbrunner, H. and N. Shah, "Incremental topological flipping works for regular triangular," *Proc. 8th Symposium on Computational Geometry*, pp. 43-52 (1992)

Field, D. "Laplacian smoothing and Delaunay triangulations," *Communications in Numerical Methods in Engineering*, **4**, 709-712, (1988)

Freitag, Lori A. and Patrick M. Knupp, "Tetrahedral element shape optimization via the Jacobian determinant and the condition number," Proc. 8th International Meshing Roundtable, pp. 247-258 (1999)

Freitag, Lori, "On Combining Laplacian and optimization-based smoothing techniques," *AMD Trends in Unstructured Mesh Genertation*, ASME, **220**, 37-43, July 1997

Garmilla, Rao V. and Mark S. Shephard, "Boundary layer mesh generation for viscous flow simulations," *International Journal for Numerical Methods in Engineering*, 49, 193-218 (2000)

References

George, P. L., F. Hecht and E. Saltel, "Automatic Mesh Generator with Specified Boundary", *Computer Methods in Applied Mechanics and Engineering*, **92**, 269-288 (1991)

George, Paul-Louis and Houman Borouchaki, "Delaunay Triangulation and Meshing", Hermes, Paris, 1998

Joe, B. Geompack90 software *URL*: http://tor-pw1.attcanada.ca/~bjoe/index.htm (1999)

Joe, B., "GEOMPACK - A Software Package for the Generation of Meshes Using Geometric Algorithms", *Advances in Engineering Software*, **56**(13), 325-331 (1991)

Joe, B., "Construction of improved quality triangulations using local transformations," SIAM Journal on Scientific Computing, 16, 1292-1307 (1995)

Lau, T.S. and S.H. Lo, "Finite Element Mesh Generation Over Analytical Surfaces", *Computers and Structures*, **59**(2), 301-309 (1996)

Lau, T.S., S. H. Lo and C. K. Lee, "Generation of Quadrilateral Mesh over Analytical Curved Surfaces," *Finite Elements in Analysis and Design*, **27**, 251-272 (1997)

Khawaja, Aly and Yannis Kallinderis, "Hybrid grid generation for turbomachinery and aerospace applications," *International Journal for Numerical Methods in Engineering*, **49**, 145-166 (2000)

Knupp, P. "Algebraic Mesh Quality Metrics for Unstructured Initial Meshes," SAND 2001-1448J, Sandia National Laboratories, submitted 2001

Lawson, C. L., "Software for C1 Surface Interpolation", *Mathematical Software III*, 161-194 (1977)

Lee, C.K, and S.H. Lo, "A New Scheme for the Generation of a Graded Quadrilateral Mesh," *Computers and Structures*, **52**, 847-857 (1994)

Liu, Anwei, and Barry Joe, "Relationship between tetrahedron shape measures," *BIT*, **34**, 268-287 (1994)

Liu, Shang-Sheng and Rajit Gadh, "Basic LOgical Bulk Shapes (BLOBS) for Finite Element Hexahedral Mesh Generation", *5th International Meshing Roundtable*, 291-306 (1996)

Lo, S. H., "Volume Discretization into Tetrahedra - II. 3D Triangulation by Advancing Front Approach", *Computers and Structures*, **39**(5), 501-511 (1991)

Lo, S. H., "Volume Discretization into Tetrahedra-I. Verification and Orientation of Boundary Surfaces", *Computers and Structures*, **39**(5), 493-500 (1991)

Lohner, R. K. Morgan and O.C. Zienkiewicz, "Adaptive grid refinement for thr compressible Euler equations, "in I. Babuska, O. C. Zienkiewicz, J. Gago, and E.R. Oliviera (Eds.) Accuracy estimates and adaptive Refinements in Finite Element Computations, Wiley, p. 281-297 (1986)

References

Lohner, R., “Progress in Grid Generation via the Advancing Front Technique”, *Engineering with Computers*, **12**, 186-210 (1996)

Lohner, Rainald, Paresh Parikh and Clyde Gumbert, "Interactive Generation of Unstructured Grid for Three Dimensional Problems", *Numerical Grid Generation in Computational Fluid Mechanics `88*, Pineridge Press, 687-697 (1988)

Lohner, Rainald and Juan Cebral, “Generation of non-isotropic unstructured grids via directional enrichment,” *International Journal for Numerical Methods in Engineering*, **49**, 219-232 (2000)

Marcum, David L. and Nigel P. Weatherill, “Unstructured Grid Generation Using Iterative Point Insertion and Local Reconnection”, *AIAA Journal*, **33**(9),1619-1625 (1995)

Marcum, David L. and J. Adam Gaither “Unstructured Surface Grid Generation using Global Mapping and Physical Space Approximation”, *Proceedings 8th International Meshing Roundtable*, 397-406 (1999)

Mitchell, Scott A., "High Fidelity Interval Assignment", *Proceedings, 6th International Meshing Roundtable*, 33-44 (1997)

Murdoch, Peter, and Steven E. Benzley, “The Spatial Twist Continuum”, *Proceedings, 4th International Meshing Roundtable*, 243-251 (1995)

Owen, Steve J. and Sunil Saigal, "H-Morph: An Indirect Approach to Advancing Front Hex Meshing". *International Journal for Numerical Methods in Engineering*, Vol 49, No 1-2, (2000) pp. 289-312

Owen, Steven J. "Constrained Triangulation: Application to Hex-Dominant Mesh Generation", *Proceedings, 8th International Meshing Roundtable*, 31-41 (1999)

Owen, Steven J. and Sunil Saigal, "Formation of Pyramid Elements for Hex to Tet Transitions", *Computer Methods in Applied Mechanics in Engineering*, Accepted for publication (approx. November 2000)

Owen, Steven J. and Sunil Saigal, "Surface Mesh Sizing Control", *International Journal for Numerical Methods in Engineering*, Vol. 47, No. 1-3, (2000) pp.497-511.

Owen, Steven J., M. L. Staten, S. A. Canann, and S. Saigal "Q-Morph: An Indirect Approach to Advancing Front Quad Meshing", *International Journal for Numerical Methods in Engineering*, Vol 44, No. 9, (1999) pp. 1317-1340

Pirzadeh S. “Viscous unstructured grids by the advancing-layers method”, In proceedings 32nd Aerospace Sciences Meeting and Exhibit, AIAA-94-0417, Reno, NV 1994.

Price, M.A. and C.G. Armstrong, “Hexahedral Mesh Generation by Medial Surface Subdivision: Part I”, *International Journal for Numerical Methods in Engineering*. **38**(19), 3335-3359 (1995)

References

Price, M.A. and C.G. Armstrong, "Hexahedral Mesh Generation by Medial Surface Subdivision: Part II," *International Journal for Numerical Methods in Engineering*, **40**, 111-136 (1997)

Rebay, S., "Efficient Unstructured Mesh Generation by Means of Delaunay Triangulation and Bowyer-Watson Algorithm", *Journal Of Computational Physics*, **106**,125-138 (1993)

Ruppert, Jim, "A New and Simple Algorithm for Quality 2-Dimensional Mesh Generation", *Technical Report UCB/CSD 92/694, University of California at Berkely, Berkely California* (1992)

Salem, Ahmed Z. I., Scott A. Canann and Sunil Saigal, "Mid-node admissible spaces for quadratic triangular arbitrarily curved 2D finite elements," International Journal for Numerical Methods in Engineering, **50**, 253-272 (2001)

Schneiders, Robert, "A Grid-Based Algorithm for the Generation of Hexahedral Element Meshes", *Engineering With Computers*, **12**, 168-177 (1996)

Shephard, Mark S. and Marcel K. Georges, "Three-Dimensional Mesh Generation by Finite Octree Technique", *International Journal for Numerical Methods in Engineering*, **32**, 709-749 (1991)

Sheffer, Alla and E. de Sturler, "Surface Parameterization for Meshing by Triangulation Flattening", *Proceedings, 9th International Meshing Roundtable* (2000)

Sheffer, Alla, Michal Etzion, Ari Rappoport, Michal Bercovier, "Proceedings 7th International Meshing Roundtable," pp.347-364 (1998)

Shewchuk, Jonathan Richard, "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator", *URL:* http://www.cs.cmu.edu/~quake/triangle.html (1996)

Shimada K., "Physically-based Mesh Generation: Automated triangulation of surfaces and volumes via bubble packing," Ph.D. Thesis, MEDept., MIT, Cambridge , MA (1993)

Shimada, K., "Anisotropic triangular meshing of parametric surfaces via close packing ofellipsoidal bubbles," Proc. 6th International Meshing Roundtable, pp. 375-390(1997)

Staten, Matthew L. and Scott A. Canann, "Post Refinement Element Shape Improvement for Quadrilateral Meshes," **AMD-220**, *Trends in Unstructured Mesh Generation*, ASME 1997 pp.9-16

Staten, M. L., S. A. Canann and S. J. Owen, "BMSweep: Locating interior nodes during sweeping", Engineering With Computers, vol 5, No. 3. (1999) pp.212-218

Staten, Mathew. L., Steven J. Owen, Ted D. Blacker, "Unconstrained Plastering", Proceedings 14th International meshing Roundtable (2005)

Talbert, J.A., and A.R. Parkinson, "Development of an Automatic, Two Dimensional Finite Element Mesh Generator using Quadrilateral Elements and Bezier Curve Boundary Definitions", *International Journal for Numerical Methods in Engineering*, **29**, 1551-1567 (1991)

References

Tam, T. K. H. and C. G. Armstrong. “2D Finite Element Mesh Generation by Medial Axis Subdivision”, *Advances in Engineering Software*, **13**, 313-324 (1991)

Taniguchi, Takeo, Tomoaki Goda, Harald Kasper and Werner Zielke, "Hexahedral Mesh Generation of Complex Composite Domain", *5th International Conference on Grid Generation in Computational Field Simmulations, Mississippi State University.* 699-707 (1996)

Tautges, Timothy J., Shang-sheng Liu, Yong Lu, Jason Kraftcheck and Rajit Gadh, "Feature Recognition Applications in Mesh Generation", *Trends in Unstructured Mesh Generation*, ASME, AMD-**220**, 117-121 (1997)

Tautges, Timothy J., Ted Blacker and Scott Mitchell, “The Whisker-Weaving Algorithm: A Connectivity Based Method for Constructing All-Hexahedral Finite Element Meshes,” *International Journal for Numerical Methods in Engineering*, **39**, 3327-3349 (1996)

Tautges, Timothy J., “The Common Geometry Module (CGM): A Generic Extensible Geometry Interface,” *Proceedings, 9th International Meshing Roundtable*, pp. 337-347 (2000)

Thompson, Joe F, Bharat K. Soni, Nigel P. Weatherill Eds. *Handbook of Grid Generation*, CRC Press, 1999

Watson, David F., “Computing the Delaunay Tesselation with Application to Voronoi Polytopes”, *The Computer Journal*, **24**(2) 167-172 (1981)

Weatherill, N. P. and O. Hassan, “Efficient Three-dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints”, *International Journal for Numerical Methods in Engineering*, **37**, 2005-2039 (1994)

White, David R. and Paul Kinney, “Redesign of the Paving Algorithm: Robustness Enhancements through Element by Element Meshing,” *Proceedings, 6th International Meshing Roundtable*, 323-335 (1997)

White, David R.,. “Automated Hexahedral Mesh Generation by Virtual Decomposition”, *Proceedings, 4th International Meshing Roundtable*, pp.165-176 (1995)

Yerry, Mark A. and Mark S, Shephard, “Three-Dimensional Mesh Generation by Modified Octree Technique”, *International Journal for Numerical Methods in Engineering*, **20**, 1965-1990 (1984)