An Introduction to Mesh Generation Algorithms

Steve Owen
Overview

• The Simulation Process
• Geometry Basics
• The Mesh Generation Process
• Meshing Algorithms
  – Tri/Tet Methods
  – Quad/Hex Methods
  – Hybrid Methods
  – Surface Meshing
• Algorithm Characteristics
Simulation Process

1. Build CAD Model
2. Mesh
3. Apply Loads and Boundary Conditions
4. Computational Analysis
5. Visualization

2 kN
Adaptive Simulation Process

1. Build CAD Model

2. Mesh

3. Apply Loads and Boundary Conditions

Adaptivity Loop

4. Computational Analysis

5. Error Estimation

Error > ε

Analysis Code supplies meshing parameters

Error < ε

6. Remesh/Refine/Improve

7. Visualization
Geometry

Mesh Generation

Geometry Engine
Geometry

vertices: x, y, z

location
Geometry

vertices: $x,y,z$
location

curves: bounded by two vertices
Geometry

vertices: $x, y, z$
location

curves: bounded by two vertices

surfaces: closed set of curves
Geometry

vertices: $x, y, z$
location

curves: bounded by two vertices

volumes: closed set of surfaces

surfaces: closed set of curves
Geometry

**body**: collection of volumes

**surfaces**: closed set of curves

**curves**: bounded by two vertices

**volumes**: closed set of surfaces

**vertices**: $x,y,z$ location
Geometry

**Vertices**: \((x, y, z)\) location

**Curves**: bounded by two vertices

**Surfaces**: closed set of curves

**Volumes**: closed set of surfaces

**Loops**: ordered set of curves on surface

**Body**: collection of volumes
Geometry

vertices: \(x, y, z\) location

curves: bounded by two vertices

coedges: orientation of curve w.r.t. loop

volumes: closed set of surfaces

surfaces: closed set of curves (loops)

loops: ordered set of curves on surface

body: collection of volumes
**Geometry**

- **vertices**: $x,y,z$ location
- **curves**: bounded by two vertices
- **coedges**: orientation of curve w.r.t. loop
- **surfaces**: closed set of curves (loops)
- **loops**: ordered set of curves on surface
- **shell**: oriented set of surfaces comprising a volume
- **volumes**: closed set of surfaces (shells)
- **body**: collection of volumes

---

*Note: The diagram illustrates the geometric relationships and definitions provided in the text.*
Geometry

- **vertices**: x, y, z location
- **curves**: bounded by two vertices
- **coedges**: orientation of curve w.r.t. loop
- **surfaces**: closed set of curves (loops)
- **loops**: ordered set of curves on surface
- **shell**: oriented set of surfaces comprising a volume
- **volumes**: closed set of surfaces (shells)
- **body**: collection of volumes
Geometry

Computational Modeling Sciences Department

**vertices**: $x,y,z$ location

**curves**: bounded by two vertices

**coedges**: orientation of curve w.r.t. loop

**surfaces**: closed set of curves (loops)

**cofaces**: oriented surface w.r.t. shell

**loops**: ordered set of curves on surface

**shell**: oriented set of surfaces comprising a volume

**volumes**: closed set of surfaces (shells)

**body**: collection of volumes
Manifold Geometry:
Each volume maintains its own set of unique surfaces
Non-Manifold Geometry: Volumes share matching surfaces
Mesh Generation Process

**Mesh Vertices** → **Apply Manual Sizing, Match Intervals** → **Mesh Curves** → **Verify/correct for sizing criteria on curves**

For each surface:
- **Set up sizing function for surface** → **Mesh surface** → **Smooth/Cleanup surface mesh** → **Verify Quality**

For each volume:
- **Set up sizing function for volume** → **Mesh volume** → **Smooth/Cleanup volume mesh** → **Verify Quality**
Meshing Algorithms

Mesh/Grid Generation Algorithms

Structured
- Quad/Hex
- Tri/Tet
  - Mapped
  - Geometry Decomposition
    - TFI
    - Elliptic
  - Hyperbolic
  - Sweeping
  - Multi-block
  - Sub-mapping
  - Medial Object

Unstructured
- Quad/Hex
- Tri/Tet
  - Indirect
  - Direct
  - Octree
  - Delaunay
  - Bubble

Hex-Tet

Dual Methods
- Whisker-Weaving
- Sheet manipulation
- Dicing
- Advancing Layers (Hybrid)
Tri/Tet Methods

Octree
Advancing Front
Delaunay

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

http://www.analytical-solutions.com
• 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

Computational Modeling Sciences Department

QMG, Cornell University
Octree/Quadtree

QMG,
Cornell University
• Begin with boundary mesh - define as initial \textit{front}
• For each edge (face) on front, locate ideal node C based on front AB
• Determine if any other nodes on current front are within search radius $r$ of ideal location C (Choose D instead of C)
• Book-Keeping: New *front edges* added and deleted from *front* as triangles are formed
• Continue until no *front edges* remain on *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*
• Where multiple choices are available, use best quality (closest shape to equilateral)
• Reject any that would intersect existing front
• Reject any inverted triangles ($|AB \times AC| > 0$)
• (Lohner,88;96)(Lo,91)
Advancing Front

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/
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
Computational Modeling Sciences Department

Delaunay

Non-Delaunay Triangulation
Lawson Algorithm

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

(Lawson, 77)
Delaunay

Lawson Algorithm
• Locate triangle containing X
• Subdivide triangle
• Recursively check adjoining triangles to ensure empty-circle property. Swap diagonal if needed
• (Lawson,77)
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)
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 \(n+1\)?
• Begin with Bounding Triangles (or Tetrahedra)
- Insert boundary nodes using Delaunay method
  (Lawson or Bowyer-Watson)
• Insert boundary nodes using Delaunay method (Lawson or Bowyer-Watson)
• Insert boundary nodes using Delaunay method
(Lawson or Bowyer-Watson)
• Insert boundary nodes using Delaunay method
  (Lawson or Bowyer-Watson)
• Insert boundary nodes using Delaunay method
  (Lawson or Bowyer-Watson)
Delaunay

- Recover boundary
- Delete outside triangles
- Insert internal nodes
Grid Based
• Nodes introduced based on a regular lattice
• Lattice could be rectangular, triangular, quadtree, etc…
• Outside nodes ignored

Node Insertion
Grid Based
• Nodes introduced based on a regular lattice
• Lattice could be rectangular, triangular, quadtree, etc…
• Outside nodes ignored

Node Insertion
Centroid
• Nodes introduced at triangle centroids
• Continues until edge length, $l \approx h$

Node Insertion
Centroid
• Nodes introduced at triangle centroids
• Continues until edge length, \( l \approx h \)

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 ($\alpha \approx 30^\circ$)

Node Insertion (Chew, Ruppert, Shewchuk)
Delaunay

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

Node Insertion (Chew, Ruppert, Shewchuk)
Advancing Front
• “Front” structure maintained throughout
• Nodes introduced at ideal location from current front edge

(Node Insertion)

(Marcum, 95)
Advancing Front
• “Front” structure maintained throughout
• Nodes introduced at ideal location from current front edge

(Node Insertion)

(Marcum, 95)
Delaunay

Node Insertion

Voronoi-Segment
• Nodes introduced at midpoint of segment connecting the circumcircle centers of two adjacent triangles

(Rebay, 93)
Delaunay

Voronoi-Segment
• Nodes introduced at midpoint of segment connecting the circumcircle centers of two adjacent triangles

(Rebay, 93)

Node Insertion
Delaunay

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

(Node Insertion) (George, 91)
Delaunay

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

Node Insertion

(George, 91)
Boundary Intersection

- Nodes and edges introduced where Delaunay edges intersect boundary
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

Boundary Constrained
Local Swapping
- Edges swapped between adjacent pairs of triangles until boundary is maintained

Boundary Constrained
Delaunay

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

Boundary Constrained
Local Swapping
• Edges swapped between adjacent pairs of triangles until boundary is maintained
Local Swapping
• Edges swapped between adjacent pairs of triangles until boundary is maintained

(George,91)(Owen,99)
Local Swapping Example
• Recover edge CD at vector $V_s$

Boundary Constrained
Local Swapping Example

- Make a list (queue) of all edges $E_i$, that intersect $V_s$

Boundary Constrained
Local Swapping Example

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

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.
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
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

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)

Boundary Constrained
Delaunay

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

DE = edge to be recovered

2-3 Swap

ABCE
ACBD

ABC = non-conforming face

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

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

Boundary Constrained
Delaunay

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

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

2-3 Swap

DE = edge recovered

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

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

Boundary Constrained
**Delaunay**

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

**2-3 Swap**

\[ ABCE \rightarrow BAED \]
\[ ACBD \rightarrow CBED \]
\[ DE = edge \text{ recovered} \]

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

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

**Boundary Constrained**
Delaunay

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
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

Algorithm

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

Geometry Requirements

- 4 topological sides
- Opposite sides must have similar intervals

(Mapped Meshing)

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

Geometry Requirements
• 6 topological surfaces
• opposite surfaces must have similar mapped meshes

3D Mapped Meshing
Structured

Block-Structured

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

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

Mapped Meshing
Structured

Geometry Requirements
- Blocky-type surfaces (principally 90 degree angles)

Sub-Mapping

(White, 95)
Structured

Automatically decomposes surface into mappable regions based on assigned intervals

Sub-Mapping

(White, 95)
Structured

Geometry Requirements
- source and target surfaces topologically similar
- linking surfaces mapable or submapable

Sweeping
Structured

Geometry Requirements
• source and target surfaces topologically similar
• linking surfaces mapable or submapable

Sweeping

linking surfaces

source

target
Structured

Geometry Requirements
• source and target surfaces topologically similar
• linking surfaces mapable or submapable

Sweeping
Structured

Geometry Requirements
• source and target surfaces topologically similar
• linking surfaces mapable or submapable

Sweeping
Structured

Geometry Requirements
• source and target surfaces topologically similar
• linking surfaces mapable or submapable

Sweeping
Structured

Geometry Requirements
• source and target surfaces topologically similar
• linking surfaces mapable or submapable

Sweeping
Structured

Geometry Requirements
• source and target surfaces topologically similar
• linking surfaces *mapable* or *submapable*

Sweeping
Structured

Geometry Requirements
• source and target surfaces topologically similar
• linking surfaces mapable or submapable

Sweeping
Structured

Gambit, Fluent Inc.

Sweeping

Cubit, Sandia National Labs
Sweeping

1-to-1 sweepable

Sweep Direction
Sweeping

Sweep Direction

n-to-1 sweepable
Sweeping

Sweep Direction

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

Sweep Direction
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.
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 Voronoi Graph
(Sheffer, 98)
Indirect Quad

Triangle splitting
• Each triangle split into 3 quads
• Typically results in poor angles
Indirect Hex

Tetrahedra splitting
• Each tetrahedra split into 4 hexahedra
• Typically results in poor angles
Indirect Hex

Tetrahedra splitting
• Each tetrahedra split into 4 hexahedra
• Typically results in poor angles
Indirect Hex

Tetrahedra splitting
• Each tetrahedra split into 4 hexahedra
• Typically results in poor angles

(Taniguchi, 96)
Indirect Hex

Example of geometry meshed by tetrahedra splitting
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
Indirect

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
Indirect

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
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)
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-Quad

Computational Modeling Sciences Department
Unstructured-Quad
Unstructured-Quad
Unstructured-Quad
Unstructured-Quad

Q-Morph
Unstructured-Quad

Q-Morph
Indirect

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
Unstructured-Hex

• 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

Grid-Based
Unstructured-Hex

• 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
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

Computational Modeling Sciences Department

Grid-Based

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

Form new row and check for overlap

(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-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

Computational Modeling Sciences Department

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
Whisker Weaving

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

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

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

Unconstrained layer of hexahedra (DOF = 2)
Unconstrained Plastering

Unconstrained column of hexahedra (DOF = 1)
Unconstrained Plastering

A single hexahedra is defined \((\text{DOF} = 0)\)
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 acelab, University of Texas, Austin, http://acelab.ae.utexas.edu/

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

Advancing Layers Method
Hybrid Methods

Advancing Layers Method

Discretize 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

Distance from wall

Element size

distribution function
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 Meshes

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

Define Normals every $\alpha$ degrees

Multiple Normals
Hybrid Methods

Multiple Normals
Hybrid Methods

Intersecting Boundary Layers
Hybrid Methods

Intersecting Boundary Layers
Hybrid Methods

Delete overlapping elements

Intersecting Boundary Layers
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

Non-Conforming Node

Non-Conforming Diagonal Edge

Conforming hex-tet?
Solutions

• Free Edge (Non-conforming)
• Multi-point Constraint
• Pyramid
Hex-Tet Interface

Heat sink meshed with hexes, tets and pyramids
Pyramid Elements for maintaining compatibility between hex and tet elements (Owen, 00)
Tetrahedral transformations to form Pyramids

- Use 2-3 swaps to obtain 2 tets at diagonal
- Combine 2 tets to form pyramid
Hex-Tet Interface

Tetrahedral transformations to form Pyramids
• Use 2-3 swaps to obtain 2 tets at diagonal
• combine 2 tets to form pyramid
Hex-Tet Interface

Tetrahedral transformations to form Pyramids

• Use 2-3 swaps to obtain 2 tets at diagonal
• combine 2 tets to form pyramid
Hex-Tet Interface

Tetrahedral transformations to form Pyramids
• Use 2-3 swaps to obtain 2 tets at diagonal
• combine 2 tets to form pyramid
Hex-Tet Interface

Tetrahedral transformations to form Pyramids
• Use 2-3 swaps to obtain 2 tets at diagonal
• Combine 2 tets to form pyramid
Non-Conforming Condition: Tets at quad diagonal A-B

Pyramid Open Method

• 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

3D Surface Advancing Front
• form triangle from front edge AB
Surface Meshing

3D Surface Advancing Front
• Define tangent plane at front by averaging normals at A and B
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)
Parametric Space Mesh Generation

- Parameterization of the NURBS provided by the CAD model can be used to reduce the mesh generation to 2D
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.
Parametric Space Mesh Generation

- Parametric space can be “customized” or \textit{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
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

\[ \Delta \mathbf{u} = \begin{pmatrix} \frac{\delta u}{\delta x'} & \frac{\delta u}{\delta y'} & \frac{\delta u}{\delta z} \\ \frac{\delta v}{\delta x'} & \frac{\delta v}{\delta y'} & \frac{\delta v}{\delta z} \end{pmatrix}, \quad \Delta \mathbf{v} = \begin{pmatrix} \frac{\delta v}{\delta x} & \frac{\delta v}{\delta y} & \frac{\delta v}{\delta z} \\ \delta x & \delta y & \delta z \end{pmatrix} \]

- Metric, \( \mathbf{M} \) can be defined at every location on surface. Metric at location \( \mathbf{X} \) is:

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

\[ E = \Delta \mathbf{u} \cdot \Delta \mathbf{u} \quad F = \Delta \mathbf{u} \cdot \Delta \mathbf{v} \quad G = \Delta \mathbf{v} \cdot \Delta \mathbf{v} \]
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:

\[ l(\overline{XQ}) \approx \sqrt{\text{M}(\text{X})^T \overline{XQ}} \]
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 $\textbf{M}(X)$.
• Can use Delaunay (George, 99) or Advancing Front Methods (Tristano, 98)
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

<table>
<thead>
<tr>
<th>Algorithm Characteristics</th>
<th>Tris</th>
<th>Tets</th>
<th>Quads</th>
<th>Hexes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Quadtree</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Delaunay</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Octree</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Adv. Front</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Adv. Delaunay</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mapped</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sub-map</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tri Split</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tri Merge</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tri Morph</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Grid-Based</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Medial Axis</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Paving</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mapped</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sub-map</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sweeping</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tet Split</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tet Merge</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tet Morph</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Grid-Based</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Medial Surf.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Plastering</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Whisker W.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Conforming Mesh
2. Boundary Sensitive
3. OrientationInsensitive
4. Regular Node Valence
5. Arbitrary Geometry
6. Commercially Viability
More Info

Meshing Research Corner

http://www.andrew.cmu.edu/~sowen/mesh.html
References


References

Computational Modeling Sciences Department


References

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


References


