Intro to Mesh Generation

Michael M. Wolf
April 20, 2005
Overview

• Introduction to Mesh Generation
• Mesh Quality
• Serial Meshing Methods
  - Quadtree/Octree
  - Advancing Front
  - Delaunay
• Parallel Mesh Generation
  - Why Parallel?
  - Categorization Parallel methods
  - Subdomains, interfaces, separators
• CSAR Mesh Repair in Rocket Simulation
Introduction to Meshing

- CAD (Continuous Model)
- Mesh (Discrete Model)
  - Domain on which to compute
Simulation Process

CAD Model → Mesh → Problem Initialization

Computation → Visualization
Adaptive Simulation Process

- CAD Model
- Mesh
- Problem Initialization
- Computation
- Error Estimation
- Remesh/Refine/Repair
- Visualization

$\varepsilon > \varepsilon$ and $\varepsilon < \varepsilon$
Types of Meshes: Typical Element Types

- **2D**
  - Triangles, Quadrilaterals

- **3D**
  - Tetrahedra, Hexahedra, Prisms, Pyramids
Types of Meshes: Regular vs Irregular

- Regular (Structured)
  - Interior nodes attached to same number of elements
- Irregular (Unstructured)
  - Interior nodes attached to variable number of elements
Mesh Quality

- Poor quality elements often yield poor solutions
- Usually regular tetrahedron (4 equilateral faces) is prototypic good element
- How to quantify “Good” element
  - Dihedral angles
  - Volume
  - Skew
  - Algebraic means
  - Etc.
Overview

• Introduction to Mesh Generation
• Mesh Quality
• Serial Meshing Methods
  - Quadtree/Octree
  - Advancing Front
  - Delaunay
• Parallel Mesh Generation
  - Why Parallel?
  - Categorization Parallel methods
  - Subdomains, interfaces, separators
• CSAR Mesh Repair in Rocket Simulation
Serial Meshing Methods

• Going to present 2D versions of methods but 3D equivalents are similar
• Focus on Triangle methods but there are numerous interesting Quad methods
Quadtree/Octree

- Setup Bounding Box
Quadtrees/Octrees

- Recursively build Quadtrees to resolve geometry
• Recursively Build Quadtree to resolve geometry
- Recursively Build Quadtree to resolve geometry
Quadtree/Octree

- Add nodes to:
  - Intersection of 2 quadtree lines
  - Intersection of boundary and quadtree line
- Remove nodes not outside boundary
• Mesh Structure using nodes with triangles
Quadtree/Octree

- Final Mesh
Advancing Front
• Place nodes around boundary.
Advancing Front

- Front initially set to be boundary.
Advancing Front

- Loop through all edges on front.
  - Find vertex which is optimal for each edge
Advancing Front

- Create triangle
- Remove edge from front
- Add new edges to front
Advancing Front

- Check radius around optimal node for nodes currently on front
Advancing Front

- If frontal node is found in radius, use instead
Advancing Front

- If choice between multiple nodes, chose best quality element
- Continue until finished
Delaunay
Empty Circle (Sphere) Property:
No other vertex is contained within the circumcircle of any triangle
Empty Circle Property:
No other vertex is contained within the circumcircle of any triangle
Delaunay

Empty Circle Property: No other vertex is contained within the circumcircle of any triangle
Delaunay
Empty Circle Property:
No other vertex is contained within the circumcircle of any triangle
Empty Circle Property:
No other vertex is contained within the circumcircle of any triangle
Delaunay
Valid Delaunay Triangulation
Valid Delaunay Triangulation
Non-Delaunay Triangulation
Non-Delaunay Triangulation
Delaunay - Node Insertion

Want to insert one node
Delaunay - Node Insertion (Lawson)

Lawson Algorithm
1. Subdivide triangle that contains new point
Delaunay - Node Insertion (Lawson)

Lawson Algorithm
2. Empty circle check for new and surrounding triangles
Delaunay - Node Insertion (Lawson)

**Lawson Algorithm**
3. Move diagonal if necessary and recheck
Delaunay - Node Insertion

Want to insert one node
Delaunay – Node Insertion (Bowyer-Watson)

**Bowyer-Watson Algorithm**
1. Find all triangles whose circumcircle contains the new node.
Bowyer-Watson Algorithm
1. Find all triangles whose circumcircle contains the new node.
Delaunay – Node Insertion (Bowyer-Watson)

**Bowyer-Watson Algorithm**

2. Remove edges interior to these triangles
Delaunay – Node Insertion (Bowyer-Watson)

**Bowyer-Watson Algorithm**
3. Connect nodes of this empty space to new node.
Delaunay

• Begin with Bounding Triangles

* From S. Owen
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)
• Insert boundary nodes using Delaunay method (Lawson or Bowyer-Watson)
Delaunay

• Delete outside triangles
Delaunay - Interior Nodes

Grid Based
• Nodes introduced based on a regular lattice
Delaunay – Interior Nodes

Grid Based
• Nodes introduced based on a regular lattice
Delaunay – Interior Nodes

**Centroid**
- Nodes introduced at triangle centroids
- Continues until edge length, $l \approx h$
Delaunay – Interior Nodes

**Centroid**
- Nodes introduced at triangle centroids
- Continues until edge length, $l \approx h$
Delaunay – Interior Nodes

Circumcenter

- Nodes introduced at triangle circumcenters
- Order of insertion based on minimum angle of any triangle
- Continues until minimum angle > predefined minimum (α ≈ 30°)

(Chew, Ruppert, Shewchuk)
Delaunay – Interior Nodes

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 – Interior Nodes

**Voronoi-Segment**

- Nodes introduced at midpoint of segment connecting the circumcircle centers of two adjacent triangles
Delaunay – Interior Nodes

**Voronoi-Segment**

- Nodes introduced at midpoint of segment connecting the circumcircle centers of two adjacent triangles
Delaunay – Interior Nodes

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

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

Boundary Intersection
- Nodes and edges introduced where Delaunay edges intersect boundary

* From S. Owen
**Boundary Intersection**

- Nodes and edges introduced where Delaunay edges intersect boundary
Delaunay – Constrained Boundary

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
Delaunay – Constrained Boundary

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

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

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

• Introduction to Mesh Generation
• Mesh Quality
• Serial Meshing Methods
  - Quadtree/Octree
  - Advancing Front
  - Delaunay
• Parallel Mesh Generation
  - Why Parallel?
  - Categorization Parallel methods
  - Subdomains, interfaces, separators
• CSAR Mesh Repair in Rocket Simulation
Parallel Mesh Generation

• Why Parallel?
  - Meshes require too much memory to generate serially
  - Mesh generation becomes computational bottleneck in simulation
  - Already have parallel simulation and need to remesh/repair/refine
Categorization of Parallel Mesh Generation

• Nikos Chrisochoides in [1] advocated the use of “off-the-shelf” serial mesh generators to develop parallel mesh generator.

• Using this idea parallel mesh generators can be categorized by:
  – Underlying sequential mesh generation algorithm
  – Parallel Coupling
Categorization of Parallel Mesh Generation

• Underlying sequential mesh generation algorithm
  – Octree
  – Delaunay
  – Etc.

• Parallel Coupling
  – Process interface meshed before subproblems meshed
  – Subproblems meshed and then process interface meshed
  – Process interface and subproblems simultaneously meshed
• Process Boundaries must be well chosen
  - Load must be balanced
  - Process boundaries should be well spaced
  - Process boundaries should not form small angle with other process boundaries or physical boundaries

• Usually not a problem if mesh partitioner is reasonable

• Constrained optimization

• Changing domains can pose a problem
Overview

• Introduction to Mesh Generation
• Mesh Quality
• Serial Meshing Methods
  - Quadtree/Octree
  - Advancing Front
  - Delaunay
• Parallel Mesh Generation
  - Why Parallel?
  - Categorization Parallel methods
  - Subdomains, interfaces, separators
• CSAR Mesh Repair in Rocket Simulation
Mesh Repair in Rocket Simulation

- Independent Study with Professor Heath and Damrong Guoy
- Want to improve mesh quality of adaptively refined mesh in rocket simulation
- Center for the Simulation of Advanced Rockets (CSAR)
- Terry Wilmarth and Phil Alexander also working on aspects of this project
Evolving Geometry of Rocket

- Shrinking solid propellant
- Expanding gas flow
- Deforming due to high pressure
- Crack propagation

Courtesy of Damrong Guoy, CSAR
Evolving Geometry

- http://www.cse.uiuc.edu/~jiao/Rocprop/movies/starslice_entropy.mpg
- http://www.cse.uiuc.edu/~jiao/Rocprop/results.html

Courtesy of Jim Jiao (via Damrong Guoy), CSAR
Poor Quality Elements

• Elements are distorted as a result of the changing geometry
• Elements in expanding region are stretched
• Elements in compressed region are flattened
Solving Mesh Distortion problem

• Mesh Smoothing
  - Moderate change in geometry
• Local mesh repair
  - Significant distortion in local region
• Global remeshing
  - Severe deformity beyond repair

Courtesy of Damrong Guoy, CSAR
Local Mesh Repair

- Repair local distortion
- Preserve large part of the mesh
- Locally refine and coarsen the mesh
- Many basic operations

Courtesy of Damrong Guoy, CSAR
Local Mesh Repair

• Basic operations
  - Vertex relocation
  - Vertex insertion
  - Edge contraction
  - Connectivity flip

Courtesy of Damrong Guoy, CSAR
Local Mesh Repair

- Basic operations
  - Vertex relocation
  - Vertex insertion
  - Edge contraction
  - Connectivity flip

Courtesy of Damrong Guoy, CSAR
Local Mesh Repair

- Basic operations
  - Vertex relocation
  - Vertex insertion
  - Edge contraction
  - Connectivity flip

Courtesy of Damrong Guoy, CSAR
Local Mesh Repair

• Basic operations
  - Vertex relocation
  - Vertex insertion
  - Edge contraction
  - Connectivity flip

![Diagram of tetrahedra and connectivity operations](image)

Courtesy of Damrong Guoy, CSAR
Simmetrix

- Using Simmetrix software (M. Shephard) for mesh repair
  - Linux, MacOS X, Windows
  - Serial and parallel (?)
  - Geometric and discrete model support
Damrong’s Global Remeshing Result

reportNumMeshEntity() num entity in mesh:-
  13979 vertices
  85533 edges
  139095 faces
  67540 regions

reportMeshQualityStatistics() supported metrics:-
1. aspect ratio = longest edge by shortest altitude
2. smallest dihedral angle
3. largest dihedral angles
4. volume skewness = ((optimal size) - (size)) / (optimal size)
5. rbyR = unitized ratio of inradius to circumradius
6. volume

<table>
<thead>
<tr>
<th></th>
<th>minimum</th>
<th>average</th>
<th>maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>aspect ratio</td>
<td>1.32</td>
<td>2.93</td>
<td>22.09</td>
</tr>
<tr>
<td>small dih angle</td>
<td>2.97</td>
<td>41.57</td>
<td>68.88</td>
</tr>
<tr>
<td>large dih angle</td>
<td>72.59</td>
<td>105.15</td>
<td>155.65</td>
</tr>
<tr>
<td>volume skewness</td>
<td>0.006635360</td>
<td>0.520993266</td>
<td>0.992803313</td>
</tr>
<tr>
<td>rbyR</td>
<td>0.103372444</td>
<td>0.670502505</td>
<td>0.996991125</td>
</tr>
<tr>
<td>volume</td>
<td>0.000000625</td>
<td>0.000034702</td>
<td>0.000242023</td>
</tr>
</tbody>
</table>

Courtesy of Damrong Guoy, CSAR
Before Mesh Repair

reportNumMeshEntity() num entity in mesh:-
143389 vertices
935693 edges
1560104 faces
767799 regions

reportMeshQualityStatistics() supported metrics:-
1. aspect ratio = longest edge by shortest altitude
2. smallest dihedral angle (degree)
3. largest dihedral angles (degree)
4. volume skewness = ((optimal size) - (size)) / (optimal size)
5. rbyR = unitized ratio of inradius to circumradius
6. volume

<table>
<thead>
<tr>
<th></th>
<th>minimum</th>
<th>average</th>
<th>maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>aspect ratio</td>
<td>1.24</td>
<td>4.56</td>
<td>169.83</td>
</tr>
<tr>
<td>small dih angle</td>
<td>0.71</td>
<td>34.87</td>
<td>69.77</td>
</tr>
<tr>
<td>large dih angle</td>
<td>71.08</td>
<td>116.54</td>
<td>178.36</td>
</tr>
<tr>
<td>volume skewness</td>
<td>0.000208097</td>
<td>0.707317863</td>
<td>0.999999960</td>
</tr>
<tr>
<td>rbyR</td>
<td>0.000517771</td>
<td>0.490753236</td>
<td>0.999882321</td>
</tr>
<tr>
<td>volume</td>
<td>0.000000044</td>
<td>0.000003054</td>
<td>0.000028641</td>
</tr>
</tbody>
</table>
After Mesh Repair

reportNumMeshEntity() num entity in mesh:-
39211 vertices
219771 edges
336631 faces
156070 regions

reportMeshQualityStatistics() supported metrics:-
1. aspect ratio = longest edge by shortest altitude
2. smallest dihedral angle (degree)
3. largest dihedral angles (degree)
4. volume skewness = ((optimal size) - (size)) / (optimal size)
5. rbyR = unitized ratio of inradius to circumradius
6. volume

<table>
<thead>
<tr>
<th></th>
<th>minimum</th>
<th>average</th>
<th>maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>aspect ratio</td>
<td>1.29</td>
<td>3.28</td>
<td>29.84</td>
</tr>
<tr>
<td>small dih angle</td>
<td>2.50</td>
<td>38.90</td>
<td>69.07</td>
</tr>
<tr>
<td>large dih angle</td>
<td>72.34</td>
<td>108.67</td>
<td>173.16</td>
</tr>
<tr>
<td>volume skewness</td>
<td>0.004655401</td>
<td>0.550685298</td>
<td>0.999879335</td>
</tr>
<tr>
<td>rbyR</td>
<td>0.014748121</td>
<td>0.631093733</td>
<td>0.997732522</td>
</tr>
<tr>
<td>volume</td>
<td>0.0000000214</td>
<td>0.000015025</td>
<td>0.000096563</td>
</tr>
</tbody>
</table>
Future Work (near future)

• Better improvement of mesh quality
  - Learn how to use Symmetrix better
  - More iterative mesh-repairing strategy
• Parallel mesh-repair
Acknowledgements


