CS 164 & CS 266: Computational Geometry

Week 10

Lecture 10b: Mesh generation

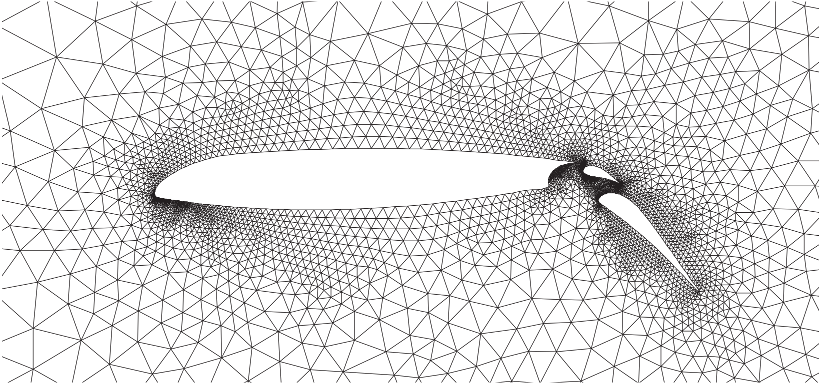
David Eppstein
University of California, Irvine

Spring Quarter, 2022

Main idea

What is a mesh?

Input: a 2d or 3d region in which we want to simulate airflow, heat, strain, or other physical properties



Mesh: subdivision into simple shapes such as triangles: "elements"

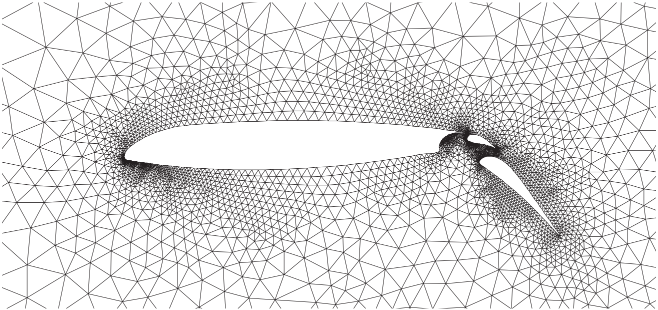
Finite element analysis

Once we have a mesh, solve a big system of linear equations:

Variables: air velocity and density within each triangle

Boundary conditions: constant velocity and density far from wing

Equations: Relate flows and densities between neighboring triangles (e.g. total air in must equal total air out)



Solution: Steady state flow

How does the mesh affect this analysis?

Number of triangles $\Rightarrow$ size of system of equations

Fewer triangles: faster

Size of elements compared to size of features of input shape and solution flow $\Rightarrow$ accuracy of simulation

Smaller triangles: more accurate

Shape of elements $\Rightarrow$ “stiffness” of system of equations $\Rightarrow$ speed and accuracy of iterative numerical methods

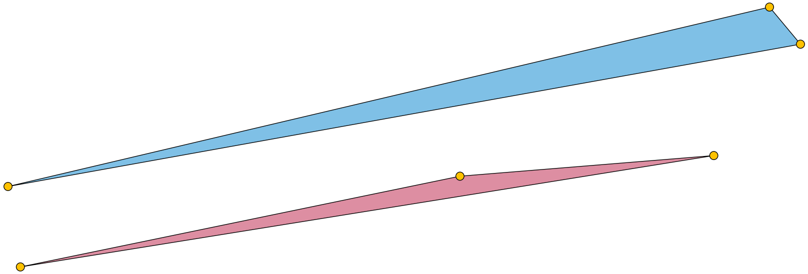
Less-sharp triangles: easier to solve

Precise relation of shape to stiffness not well understood

What shapes should we aim for?

Possibility 1: Avoid sharp (near-zero) angles

Possibility 2: Sharp ok, but avoid wide (near-π) angles



Possibility 3: Whether a shape is good or bad depends on the solution values near it, and not on the shape itself

Lower bounds on numbers of triangles

Simple shapes may require many triangles

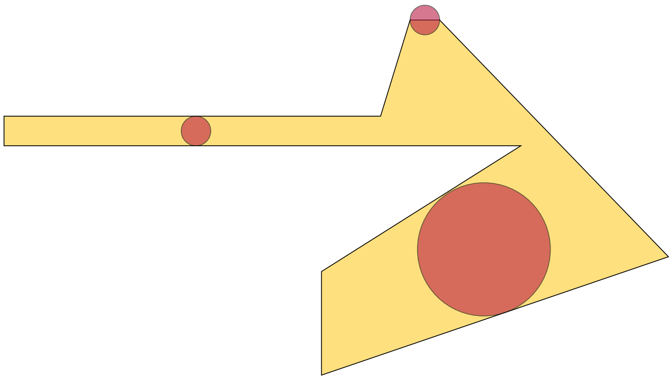
If we forbid sharp angles ($< \varepsilon$ for some $\varepsilon > 0$)
then $1 \times x$ rectangle requires $\Omega(x)$ triangles
even though $n = 4 = O(1)$

Corollary: Number of triangles cannot be a function only of n
and we cannot get a time bound depending only on n

[Bern et al. 1990]

Local feature size

Radius of smallest circle that intersects two non-touching polygon edges

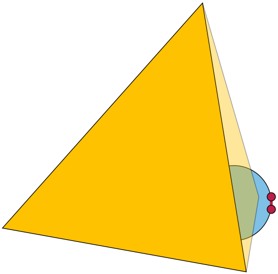


Area vs local feature size

Claim: In a triangle mesh with no sharp angles (all angles $> \varepsilon$), each point in a triangle of area A has local feature size $\Omega(\sqrt{A})$

Ideas of proof:

- No sharp angles $\Rightarrow$ all sides of triangle have length $\Omega(\sqrt{A})$
- If any point of the triangle is near two disjoint features $\Rightarrow$ a point on a side triangle side is near the same two features
- For the adjacent triangle on that side to avoid extending past those features, it would need a sharp angle



Lower bound on number of triangles

For a domain (polygon) D, In a triangle mesh of D with no sharp angles, let N be the number of triangles in the mesh, and let $a(p)$ be the area of the triangle containing any point p.

Then the integral of the constant function $1/\text{area}$ over a single triangle is one, and combining with the inequality of area versus local feature size gives:

$$N = \int_D \frac{1}{a(p)} dx\, dy = \Omega\left(\int_D \frac{1}{\operatorname{lfs}(p)^2} dx\, dy\right).$$

Corollary: If we can find a triangulation where every triangle has diameter at least proportional to the local feature size, it will automatically use an optimal number $O(N)$ of triangles

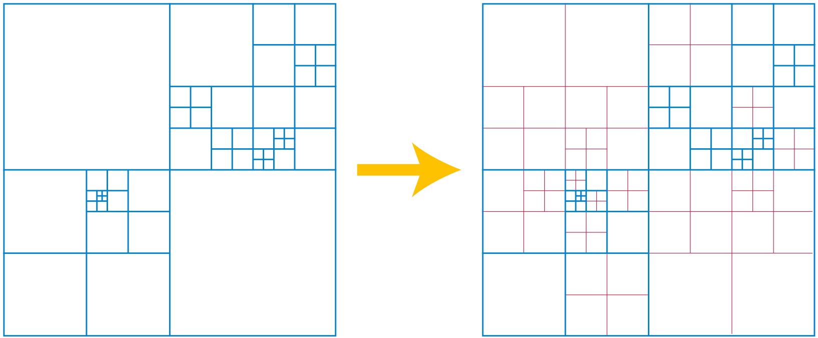
[Ruppert 1993]

Quadtree-based meshing

Balanced quadtree

Construct a quadtree normally (recursively split overfull squares)

Then, while any square has neighboring squares $< \frac{1}{2}$ its size, split it

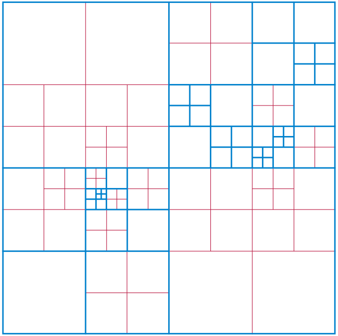


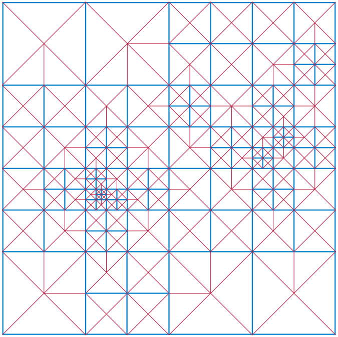
If a square of side length s is split, it must be within distance $2s$ of a smaller square of the original quadtree

Triangulating a balanced quadtree

Add a vertex at the center of each square

Connect it to the vertices on the boundary of the square



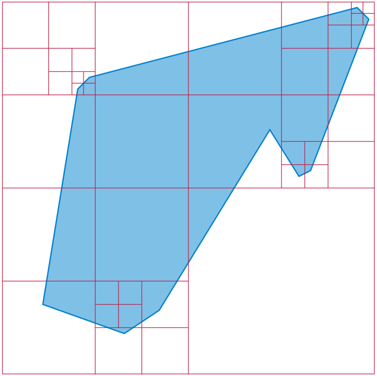


All triangles are isosceles right triangles, angles 45° and 90°

Quadtree-based meshing

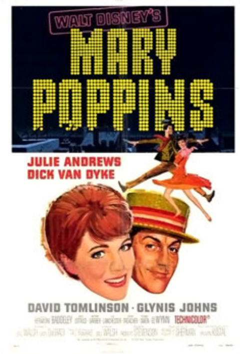
- Surround whole polygon in a bounding square
- Recursively subdivide squares crossed by non-touching edges

- More subdivision + messy case analysis for squares crossed by boundary
- Balance
- Triangulate empty squares



Quadtree mesh analysis

Mary Poppins:
"practically perfect in every way"



All angles bounded away from 0 and 180°

(except for sharp angles of input polygon)

All triangles have diameter proportional to local feature size $\Rightarrow O(N)$ triangles, within a constant factor of optimal

Construction takes time linear in mesh size

[Bern et al. 1990]

Theoretically perfect, but practically?

Constant factor in $O(N)$ bound on number of triangles is large

Using a quadtree causes many triangle edges to be aligned with coordinate axes or diagonal, alignment may lead to unwanted bias in simulations performed with these meshes

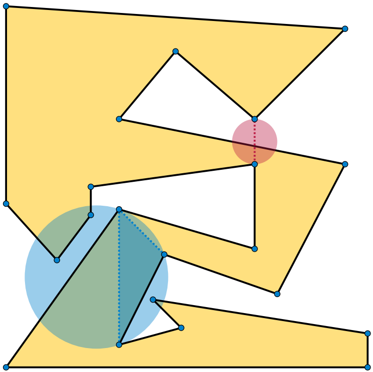
Incremental Delaunay meshing

Constrained Delaunay triangulation

Like Delaunay triangulation, but forced to include all edges of input polygon

Dual of Voronoi diagam for distance along curves inside polygon

Empty circle property of triangles: stuff separated from the triangle by a polygon edge doesn't count

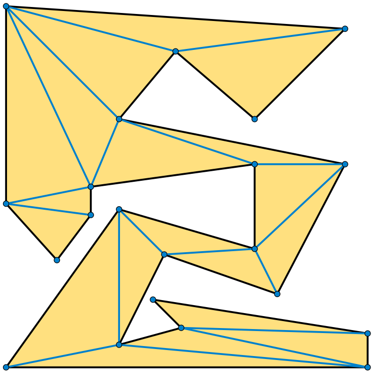


Constrained Delaunay triangulation

Like Delaunay triangulation, but forced to include all edges of input polygon

Dual of Voronoi diagam for distance along curves inside polygon

Empty circle property of triangles: stuff separated from the triangle by a polygon edge doesn't count

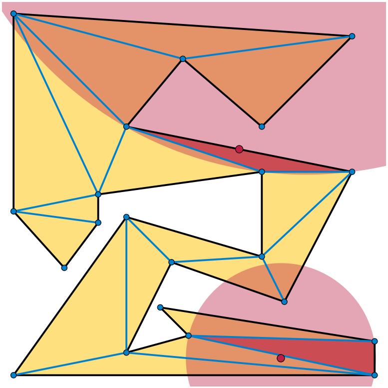


Incremental Delaunay meshing

Start with constrained Delaunay

While some triangle Δ has a too-sharp angle:

- Find center c of circle through vertices of Δ
- If c is visible to Δ, add c to input
- Otherwise, add the midpoint of the boundary edge that blocks visibility



Incremental Delaunay analysis

Same theoretical guarantees as quadtrees on avoiding sharp angles and using optimal number of triangles

Variations of this method have been proven to generate meshes with minimum angle $\geq 26.5^\circ$

In practice, angles greater than 30° are often possible

Avoids the practical issues of quadtree meshing

[Chew 1993; Ruppert 1993; Shewchuk 2014]

Mesh smoothing

The main idea

Even for meshing algorithms that guarantee shape and number of triangles is good, we can often do better

Move interior vertices of the mesh one at a time to better positions

No quality guarantees but if we start with a good mesh and are careful to only make improvements, it will stay good

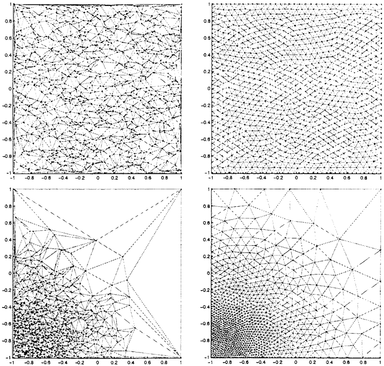
Lloyd's algorithm

Repeatedly replace each point with the centroid of its Voronoi cell

(But don't repeat too often to preserve density gradients)

No guarantee on quality but typically converges to near-equilateral-triangle mesh

[Lloyd 1982; Du and Gunzburger 2002]



Left: before; right: after

Figure from Du et al.

Optimization-based mesh smoothing

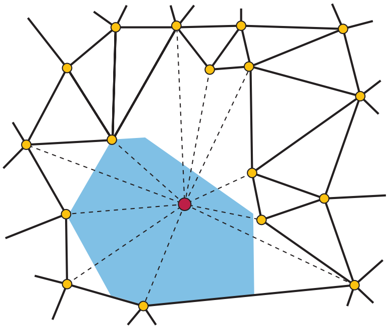
For each point (one at a time):

Find star-shaped polygon formed by its neighboring triangles

Choose new position in kernel of polygon, optimizing quality of its triangles

For many natural quality criteria, the problem of finding the optimal new position is LP-type

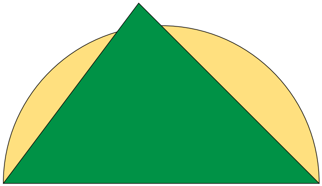
[Amenta et al. 1999]



Non-obtuse triangulation

Right angles are special

Non-obtuse ⇒ No vertex can be interior to the semicircle on the opposite side ⇒ Diameter circle of each edge is empty ⇒ Delaunay



Numerical properties of system of equation ("M-matrix")

Right angles are special in another way

We can get meshes with only $O(n)$ triangles and all angles $\leq 90^\circ$

We already saw that if we want all angles $\geq \varepsilon$,
$\#$ triangles depends on geometry not just on n

But if we try to get all angles $\leq 90^\circ - \varepsilon$, we also get all angles $\geq 2\varepsilon$

$\Rightarrow$ complexity cannot be linear

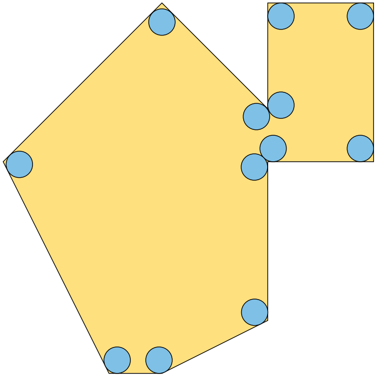
Main idea

If it's a non-obtuse triangulation, it will be a Delaunay triangulation

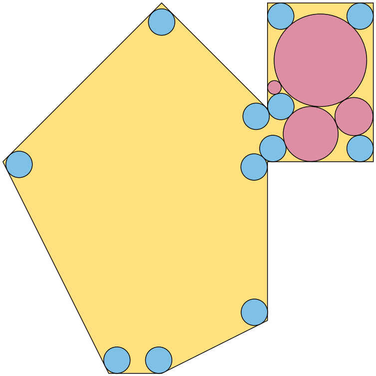
Therefore, there will be lots of empty circles

Add the circles first, then build the triangulation around them

Packing circles into a polygon



Protect vertices

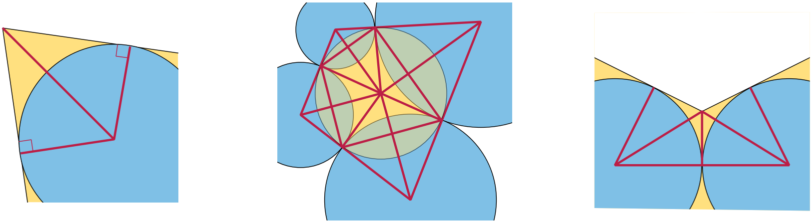


Split regions with > 4 sides

Non-obtuse triangulation

Add radii from centers to points of tangency

Messy case analysis: All $\leq$ 4-sided regions can be triangulated



Result: non-obtuse triangulation, $O(n)$ triangles [Bern et al. 1995]

References and image credits, I

Nina Amenta, Marshall Bern, and David Eppstein. Optimal point placement for mesh smoothing. *Journal of Algorithms*, 30(2):302–322, 1999. doi: 10.1006/jagm.1998.0984.

Marshall Bern, Scott Mitchell, and Jim Ruppert. Linear-size nonobtuse triangulation of polygons. *Discrete & Computational Geometry*, 14(4): 411–428, 1995. doi: 10.1007/BF02570715.

Marshall W. Bern, David Eppstein, and John R. Gilbert. Provably good mesh generation. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 231–241. IEEE Computer Society, 1990. doi: 10.1109/FSCS.1990.89542.

L. Paul Chew. Guaranteed-quality mesh generation for curved surfaces. In Chee Yap, editor, *Proceedings of the Ninth Annual Symposium on Computational GeometrySan Diego, CA, USA, May 19-21, 1993*, pages 274–280. ACM, 1993. doi: 10.1145/160985.161150.

References and image credits, II

Qiang Du and Max Gunzburger. Grid generation and optimization based on centroidal Voronoi tessellations. *Applied Mathematics and Computation*, 133(2–3):591–607, 2002. doi: 10.1016/S0096-3003(01)00260-0.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. doi: 10.1109/TIT.1982.1056489.

Jim Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. In Vijaya Ramachandran, editor, *Proceedings of the Fourth Annual ACM/SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas, USA*, pages 83–92, 1993.

Jonathan Richard Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 47(7):741–778, 2014. doi: 10.1016/j.comgeo.2014.02.005.