Computers Dream of Triangles

John Burkardt
Department of Scientific Computing
Florida State University

.....

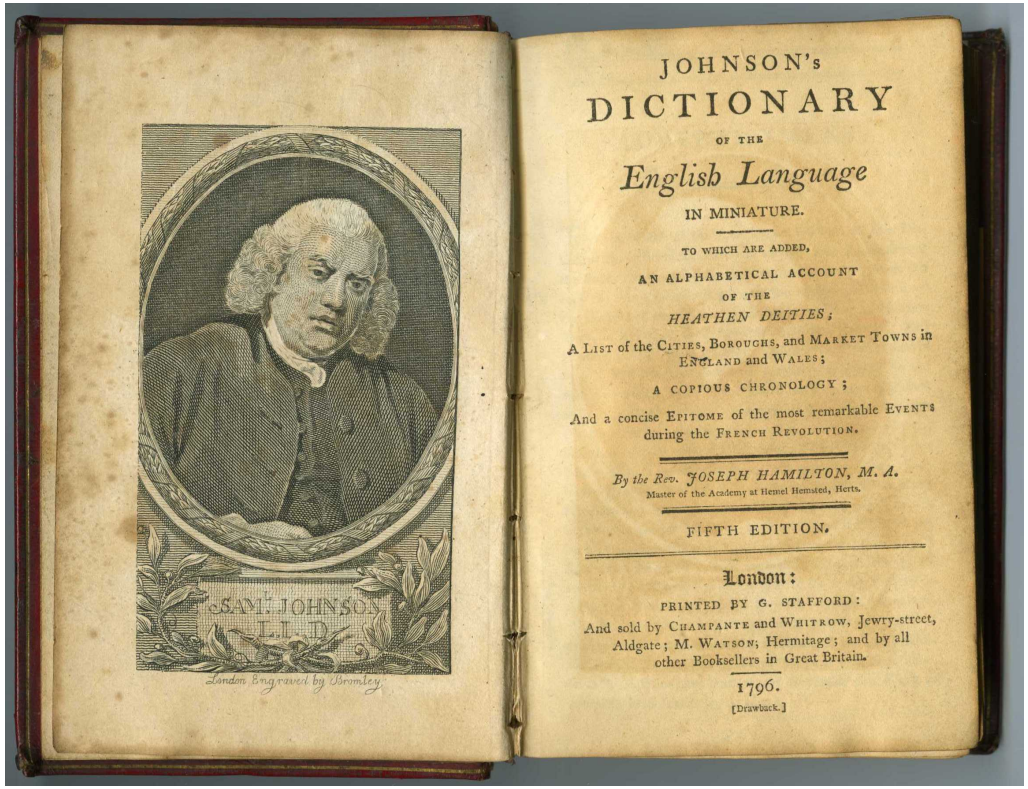
3:00-4:00, 22 January 2016
Graduate Student Seminar

.....

http://people.sc.fsu.edu/~jburkardt/presentations/...
mesh_2016_fsu.pdf

The Worst Definition

"**Network**: *Any thing reticulated or decussated, at equal distances, with interstices between the intersections.*"
– Samuel 'Dictionary' Johnson.



What is a Mesh?

A mesh can be a physical network (a net, a fence, a screen).

A mesh can be a mathematical object, a collection of nodes, representing geometric points, and pairs of nodes, representing geometric lines, which forms a discrete model of a physical shape or surface.

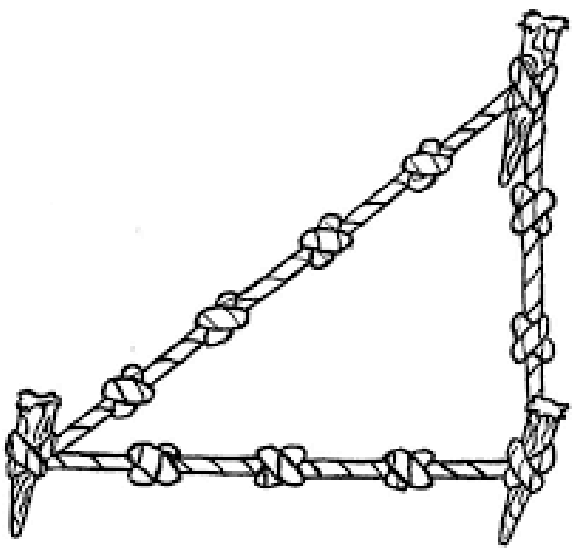
It's easy to "explain" a mathematical mesh to a computer.

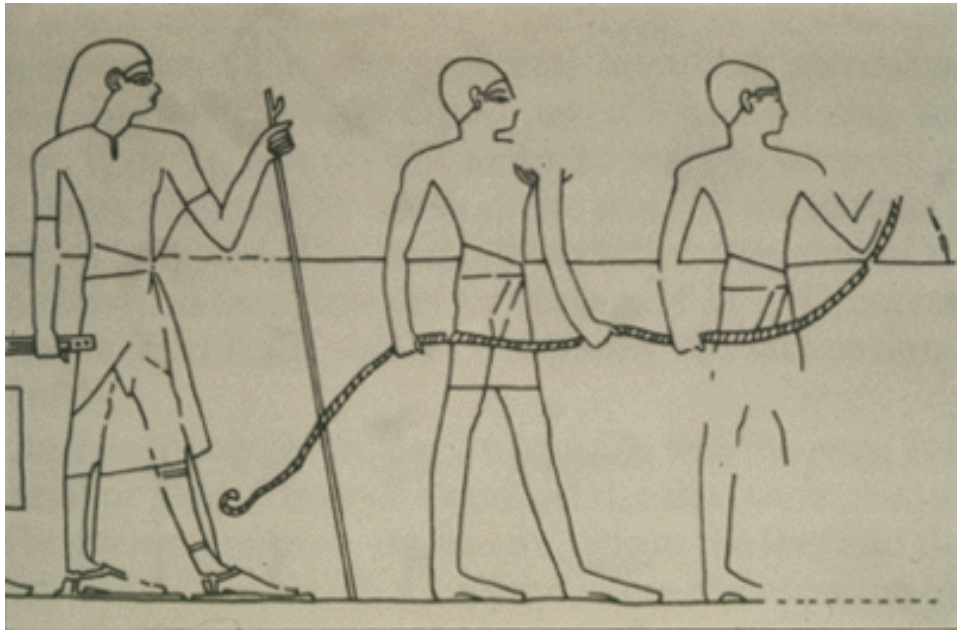
Although the computer can't see the physical shape, it can "see" the computational mesh, and do interesting calculations about it.

In this way, a mesh allows a computer to "dream" of geometry the way that the IEEE format allows it to dream of numbers.

It All Began with Taxes

The fertile land along the Nile river was broken up into strips, and the owner of each strip was taxed according to its area. The annual floods meant that the boundaries between strips frequently had to be redrawn. This was done using a 3-4-5 triangle of rope, which guaranteed a right angle and a good survey. The surveyors were known as "rope stretchers".



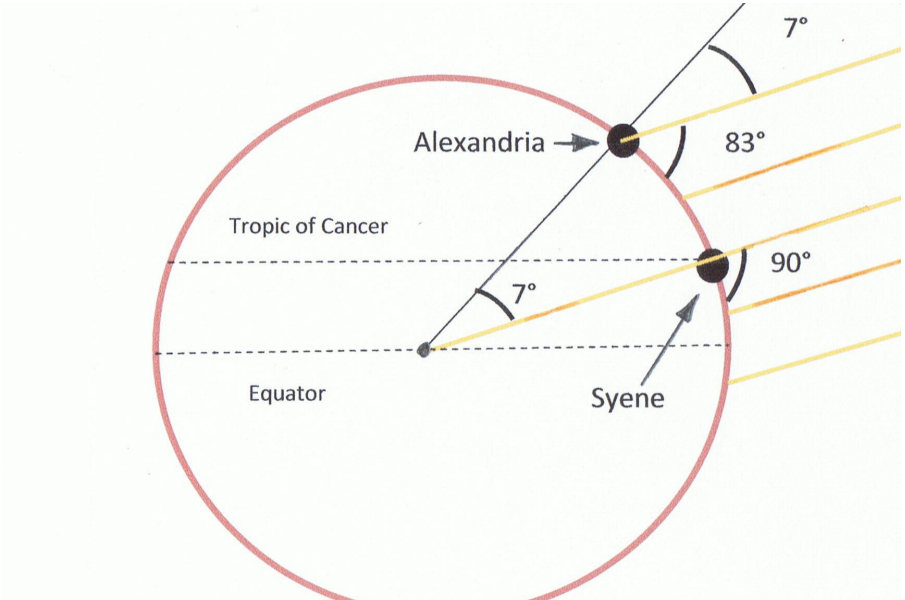


Measuring the Earth

The Greeks named it (*geo-metry* = *earth* + *measurement*), and focused on triangles (*tri-gono-metry* = *three* + *side* + *measurement*).

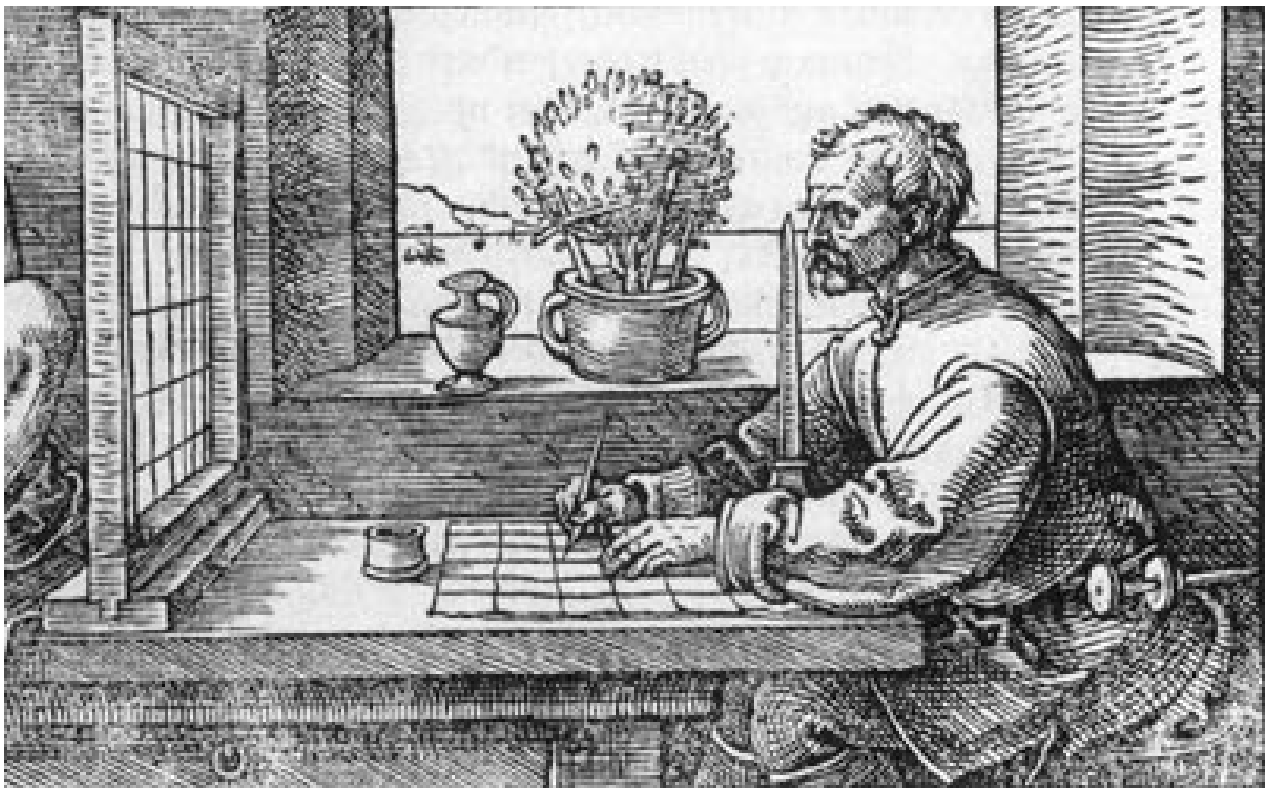
They realized a triangle is better than a quadrilateral for surveying because it is rigid, and there are relations between sides and angles.

Eratosthenes used this knowledge to estimate the size of the earth, the tilt of the Earth's axis, and the distance to the sun.



A Mesh Organizes and Simplifies a Complicated View

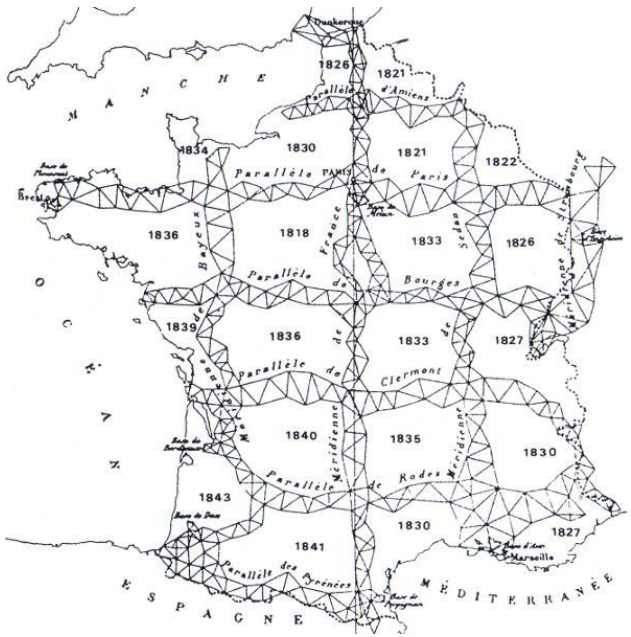
Artists discovered that they could use a mesh to divide up a prospect that they were observing, to divide up a sheet on which they were drawing, and to transfer small items of information from one mesh box to the other.



A Mesh Can Organize and Measure Curved Surfaces

French revolutionaries demanded new scientific units, and defined the meter as 1/40,000,000 of the earth's circumference. Unfortunately, no one had a reliable estimate of this.

Surveyors made an extremely accurate sequence of triangles along a line of longitude, to answer this question. They also established the baseline for a set of triangles used to map all of France.

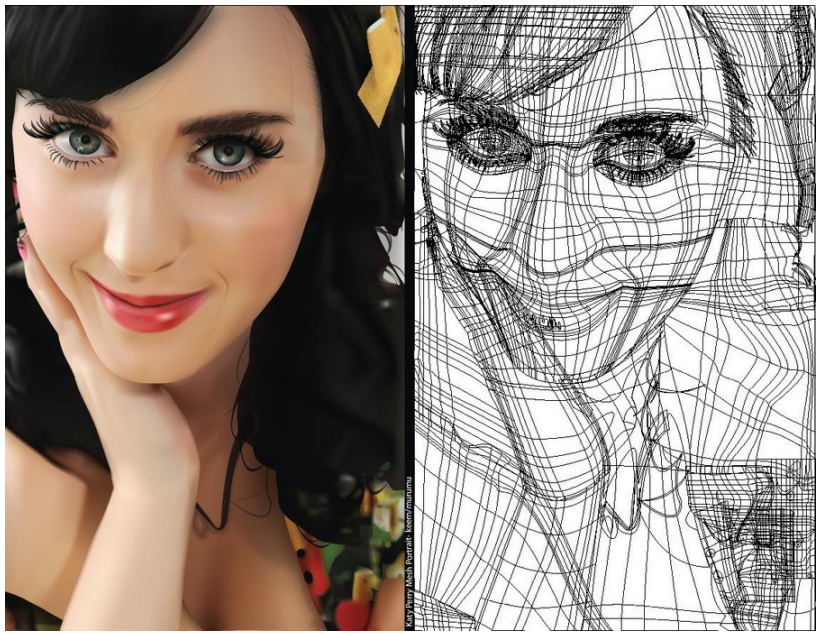


Computers Model Reality with Meshes

Meshes capture geometry like nets capture fish.

The little ones get away (whether fish, or geometry objects) but we can refine the mesh if desired.

While physical reality can be messy, and mathematical models can be too idealized, a mesh is a practical, adaptable, discrete computational tool.



It was Easy to Teach Arithmetic to Computers

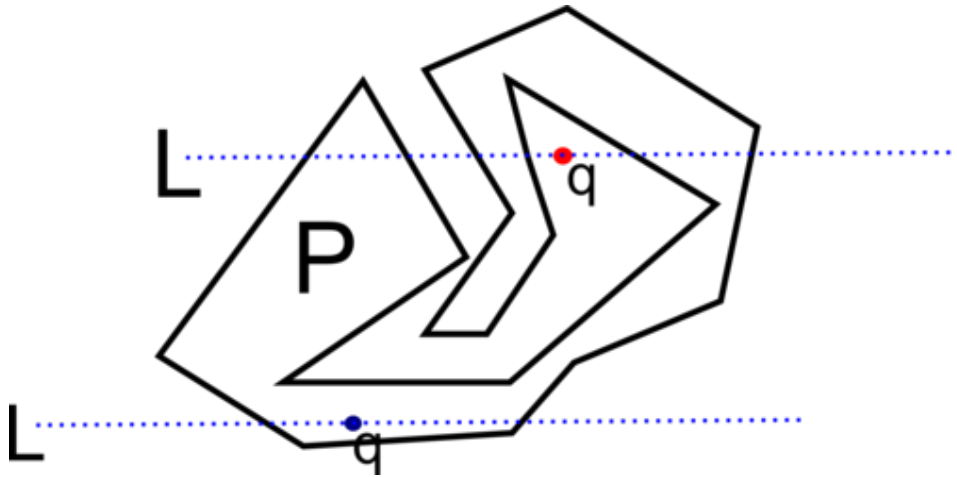
Now that we want computers to do all our work, we have to figure out how to convert our physical problems to mathematical models, which can be represented and manipulated on the computer.

Teaching a computer about natural numbers is easy. The IEEE format allows us to count 0, 2, 3, 4, ... (except that now ... really means keep going til maximum value and then wrap around!), and programming languages let us type "17", meaning ... seventeen.

Similarly, IEEE real numbers seem to behave like mathematical real numbers, although if we look closely, we can see that there is a maximum real number, a minimum nonzero real number, that A * 1/A is not 1, and that A+1=A is possible.

But what happens when we deal with sets of objects in $\mathbb{R}^2$, that is, simple planar geometric shapes?

It's Not So Easy to Teach Geometry to Computers!



Interesting geometric questions need at least two dimensions:

- perimeters,
- areas,
- centers of mass,
- whether a point is inside or outside a shape,
- the distance between a point and a shape
- can I get through a maze?
- how many objects can fit inside this shape?
- do these two shapes fit together?

What If You Only Have an Outline?

A simple model of a shape is the curve formed by its outline.

Mathematically, this is a continuous (and perhaps differentiable) closed curve.

As long as the curve isn't too weird, we can make a discrete model of it as a sequence of straight lines.

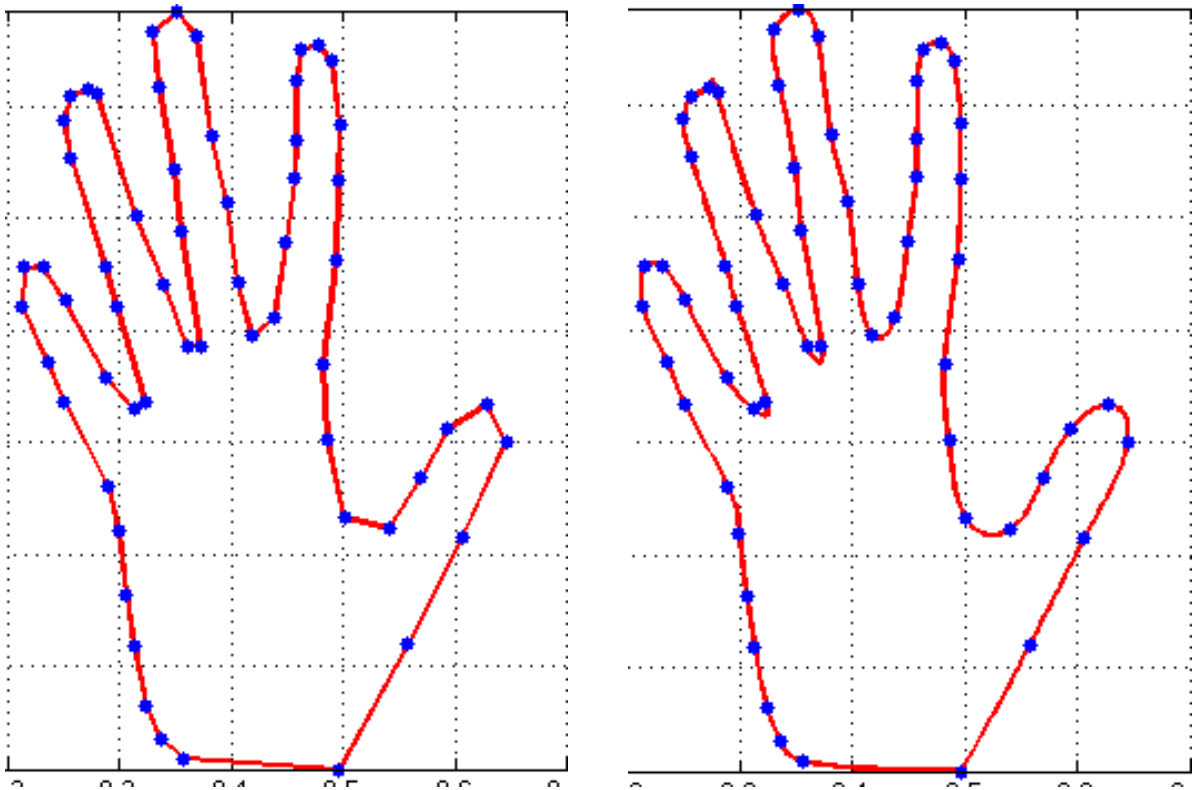
Then the computer can use the sequence of "turning points" as an approximate version of the original curve.

We Can Automate the Measurements

```
%
%  Get the screensize.
%
    screen = get ( 0, 'ScreenSize' );
%
%  Use the whole screen as a figure.
%
  figure ( 'Position', screen );
%
%  Create a graphical coordinate system on the figure.
%
  axes ( 'Position', [ 0, 0, 1, 1 ] );
%
%  The user clicks sample points
%  and terminates with RETURN.
%
  [ x, y ] = ginput ( );
```

http://people.sc.fsu.edu/~jburkardt/m_src/hand_data/hand_acquire.m

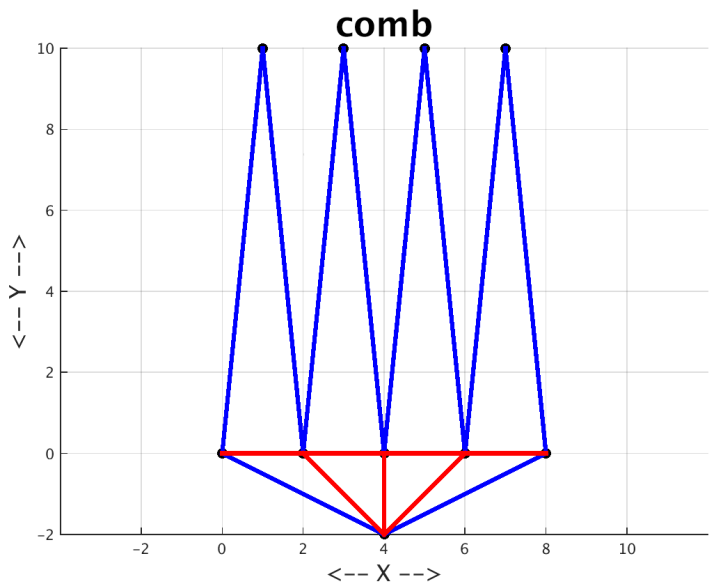
From Points We Have a Curve

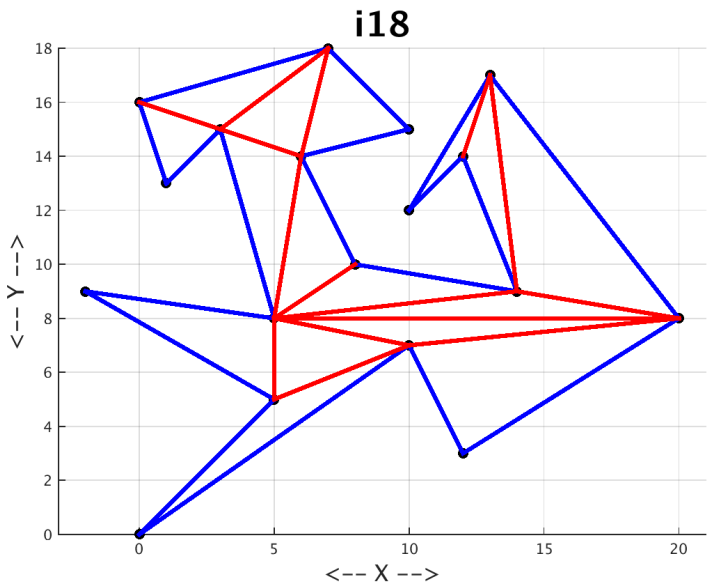


We can connect the points with straight lines (a piecewise linear model) or use a fancy technique called splines for a smoothe curve.

http://people.sc.fsu.edu/~jburkardt/m_src/hand_data/hand_spline.m

A Polygon is a Collection of Triangles





Any (reasonable) polygon can be dissected into triangles, because such a polygon has at least one "ear", two successive edges that form a triangle entirely inside the polygon.

If we slice off this ear, the remaining shape is a polygon with N-1 sides. Keep slicing til we get everything.

The Area of a Triangle

Once we have reduced a polygon to triangles, some questions are easy to answer.

The cross-product of two vectors is the area of the parallelogram between them.

A triangle with corners (A,B,C) can be regarded as two vectors (B-A) and (C-A).

The area is 1/2 the determinant of this 2x2 matrix:

$$\begin{aligned}\text{area}(A,B,C) &= 1/2 \begin{vmatrix} B_x - A_x & B_y - A_y \\ C_x - A_x & C_y - A_y \end{vmatrix} \\ &= 1/2((B_x - A_x)(C_y - A_y) - (B_y - A_y)(C_x - A_x))\end{aligned}$$

Code for Triangle Area

```
1  function area = triangle_area ( ax, ay, bx, by, cx, cy
       )
2
3    area = 0.5 * ...
4      ( ( bx - ax ) * ( cy - ay ) ...
5      - ( cx - ax ) * ( by - ay ) );
6
7    return
8  end
```

Listing 1: Area of triangle

http://people.sc.fsu.edu/~jburkardt/m_src/triangle_properties/triangle_area.m

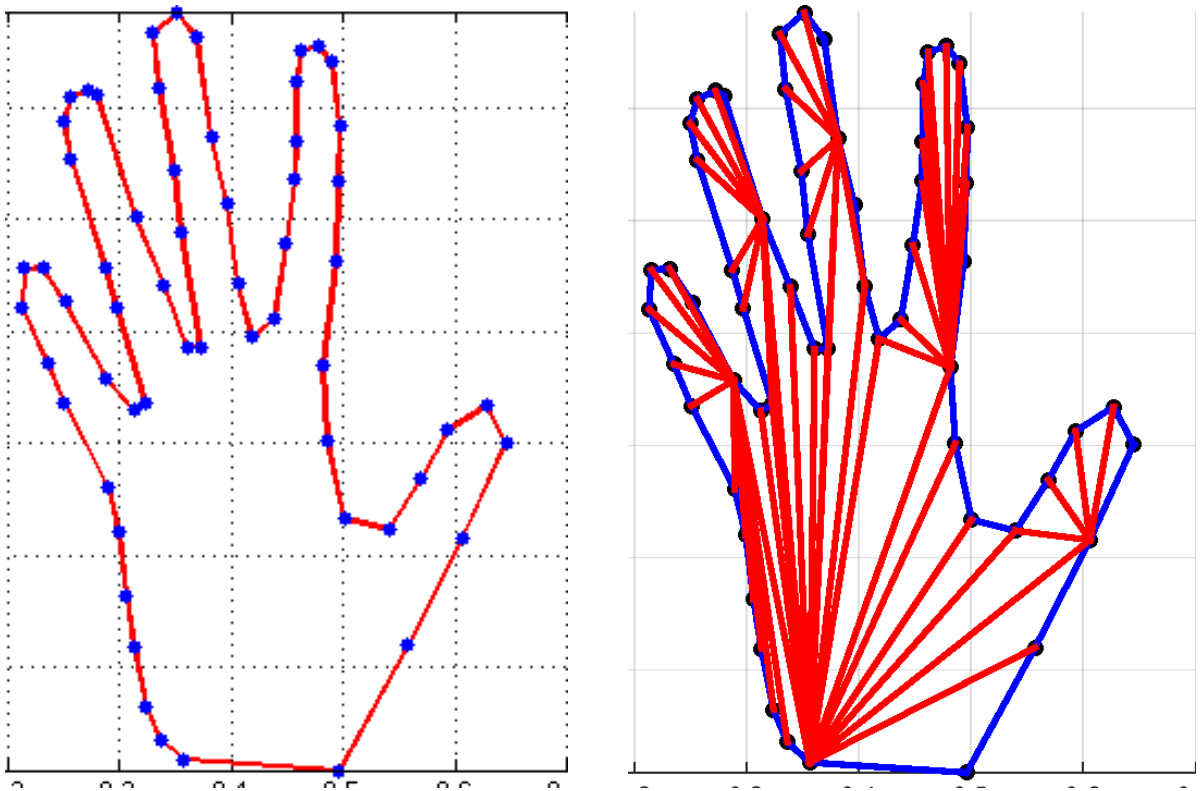
The Area Formula Can Return a Negative Value!

It is possible to get a **negative area**. Indeed, if A = (0,0), B = (1,4), C = (3,2), we get area (A,B,C) = -5. Interestingly enough, area (A,C,B) = +5.

The minus sign is telling us something very useful: the triangle (A,B,C) has its vertices listed in clockwise order, but (A,C,B) lists them in counterclockwise order, The sign of the area is a warning about the orientation of the triangle.

As long as we promise to list triangle vertices in counterclockwise order, we will have no problems with the area formula. But it turns out that this bit of knowledge can be used to determine other information.

The Area of My Hand



If I work with the straight line outline, I can triangulate my hand.

By summing triangle areas, I get the **area of my hand**.

http://people.sc.fsu.edu/~jburkardt/m_src/polygon_triangulate/polygon_triangulate.m

Filling in the Holes

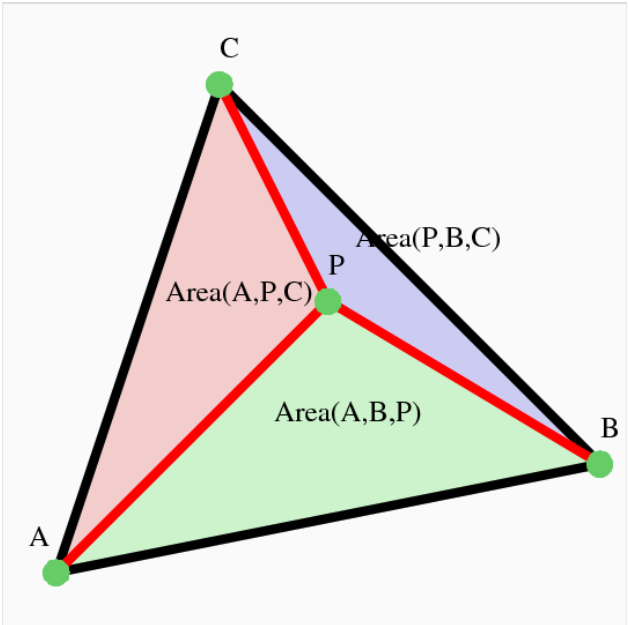
Many engineering problems start with a geometric region, and try to determine the value of heat, pressure, wind velocity, mechanical stress, or solvent concentration, at every point in the region.

For our discrete triangular geometry, the natural way to do this is to determine these values at the corners of each triangle, and then to smoothly interpolate values inbetween.

Thus, we need to be able to do two tasks:

- Determine if a point P is in triangle T
- Estimate quantities at P from values at the corners of T

Does a Triangle Contain a Point?



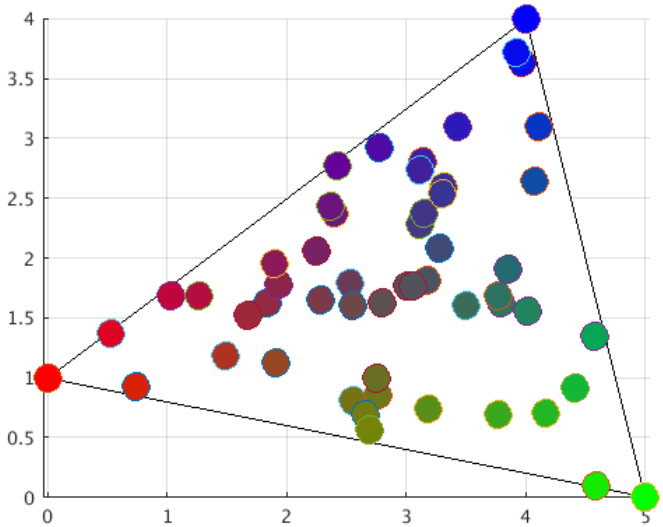
Given triangle T with corners (A,B,C) and a point P...

P is inside T if all three areas are positive:

- Area(P,B,C)
- Area(A,P,C)
- Area(A,B,P)

Thus we can determine if P lies inside any triangulated polygon.

Interpolation in a Triangle



If the corners (A,B,C) have values (F(A),F(B),F(C)). we can assign a value F(P):

F(P) = Area(P,B,C) F(A)+Area(A,P,C) F(B)+Area(A,B,P) F(C) / Area(A,B,C).

Thus we can interpolate data across any triangulated polygon.

http://people.sc.fsu.edu/~jburkardt/m_src/triangle_interpolate/triangle_interpolate.m

Internal Sample Points

Triangulating the hand completely covers the internal area, but it does so with triangles of many different sizes and shapes.

There might be reasons that we want a pattern of triangles that covers the region more evenly in shape and size. We can do this by augmenting our boundary points by internal sample points.

But can we choose these efficiently? I am willing to think, but only once!

MESH2D: Region Defined by Vertices

mesh2d is a computer program for which the user only has to describe an outline of the region of interest, that is, a counterclockwise list of points.

We will start by asking for a simple mesh of a simple region, and then push the code little by little to harder tasks.

---

http://www.mathworks.com/matlabcentral/fileexchange/25555-mesh2d-automatic-mesh-generation

MESH2D: Usage

```
[ p, t ] = mesh2d ( vertices, edge, hdata, options );
```

where:

- *vertices*, a **V** by 2 list of boundary vertex coordinates;
- *edge*, (optional input), lists pairs of vertex indices that form the boundary;
- *hdata*, (optional input), a structure containing element size information;
- *options*, (optional input), allows the user to modify the default behavior of the solver .
- *p*, the coordinates of nodes generated by the program;
- *t*, the triangulation of the nodes.

MESH2D: Simple ELL Mesh

As examples of mesh2d usage, we can start with variations of the L-shaped problem:

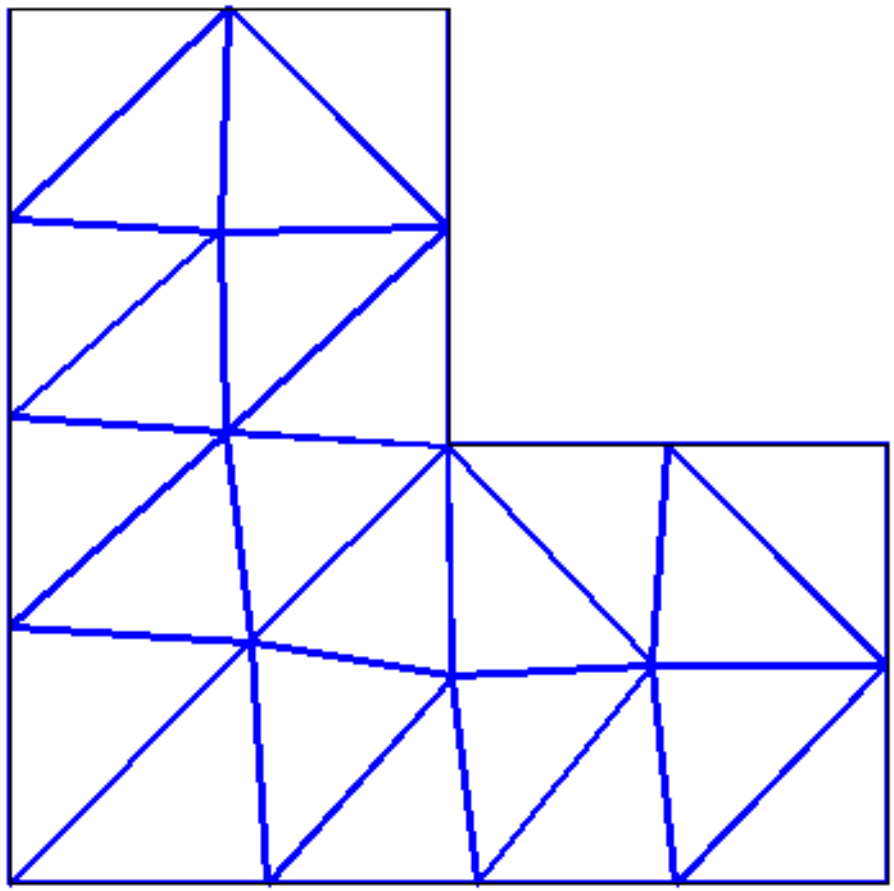
```
v = [ 0.0, 0.0; ...
      2.0, 0.0; ...
      2.0, 1.0; ...
      1.0, 1.0; ...
      1.0, 2.0; ...
      0.0, 2.0 ];

[ p, t ] = mesh2d ( v );
```

---

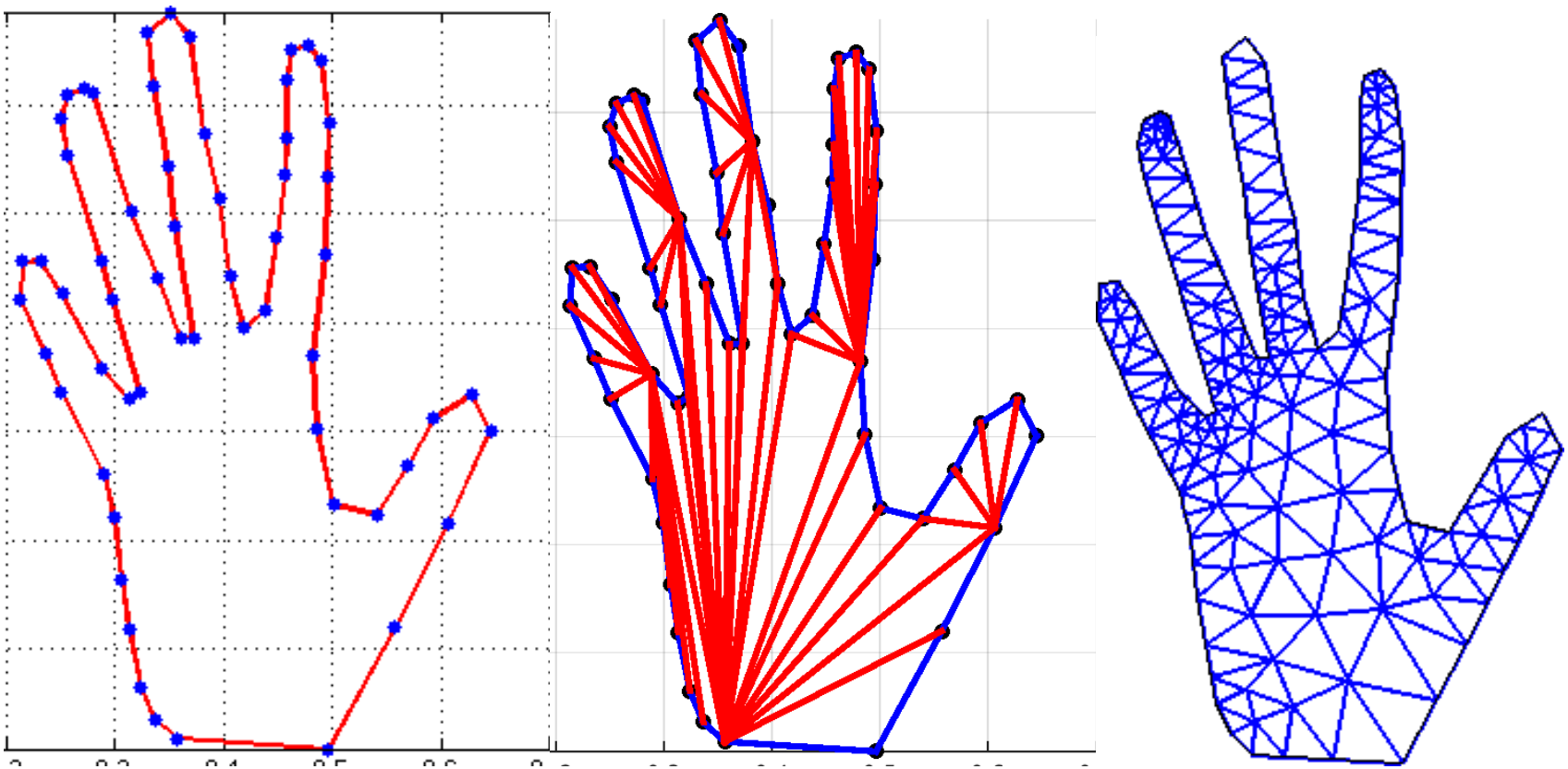
http://people.sc.fsu.edu/~jburkardt/m_src/mesh2d/ell_demo.m

MESH2D: Simple ELL Mesh



MESH2D includes some interior points as it builds the mesh.

Apply MESH2D to the Hand Data



For the hand data, MESH2D again inserts interior points. The resulting (blue) mesh is smoother than the (red) mesh using boundary points.

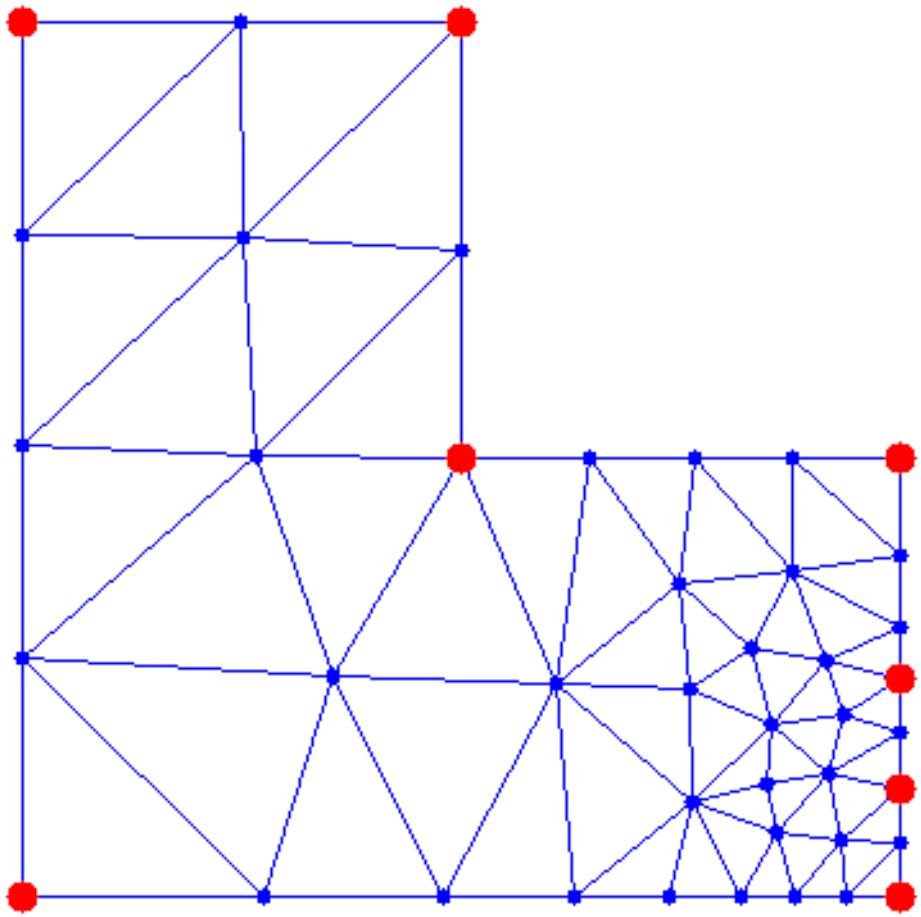
MESH2D: Two Short Boundary Segments

Suppose we add two extra boundary vertices:

```
v = [ 0.0, 0.0; ...
      2.0, 0.0; ...
      2.0, 0.25;
      2.0, 0.5; ...
      2.0, 1.0; ...
      1.0, 1.0; ...
      1.0, 2.0; ...
      0.0, 2.0 ];

[ p, t ] = mesh2d ( v );
```

MESH2D: Two Short Boundary Segments



MESH2D: Set Maximum Element Size

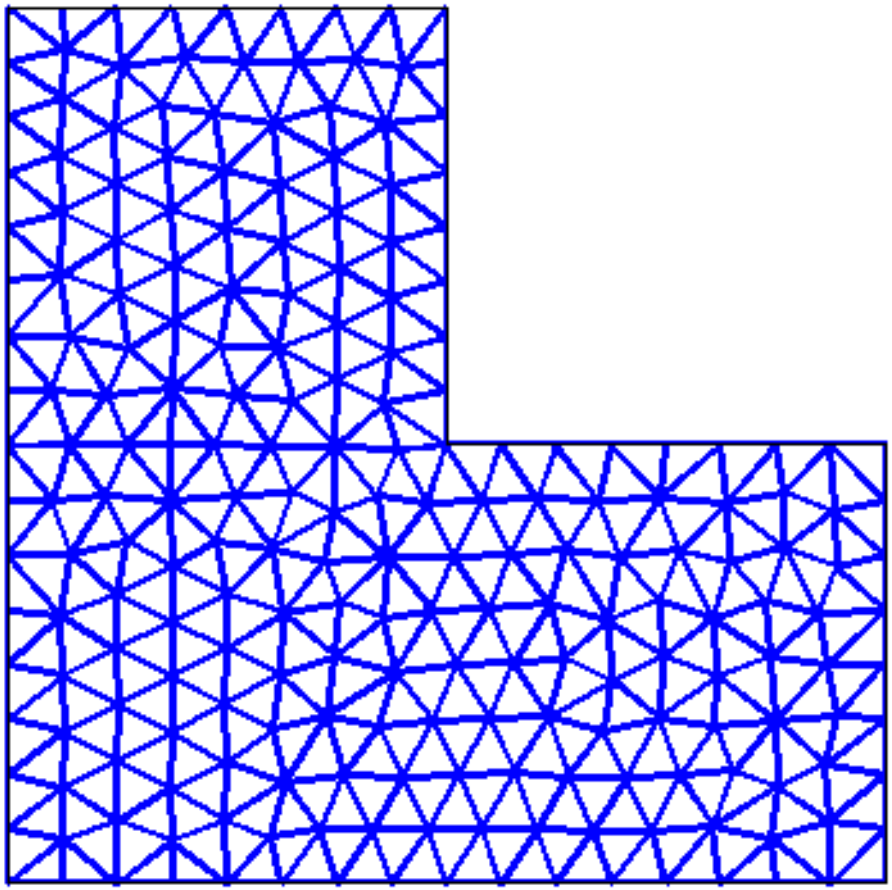
Go back to the original problem, but specify a maximum element size:

```
v = [ 0.0, 0.0; ...
      2.0, 0.0; ...
      2.0, 1.0; ...
      1.0, 1.0; ...
      1.0, 2.0; ...
      0.0, 2.0 ];

hdata = [];
hdata.hmax = 0.1;

[ p, t ] = mesh2d ( v, [], hdata );
```

MESH2D: Set Maximum Element Size



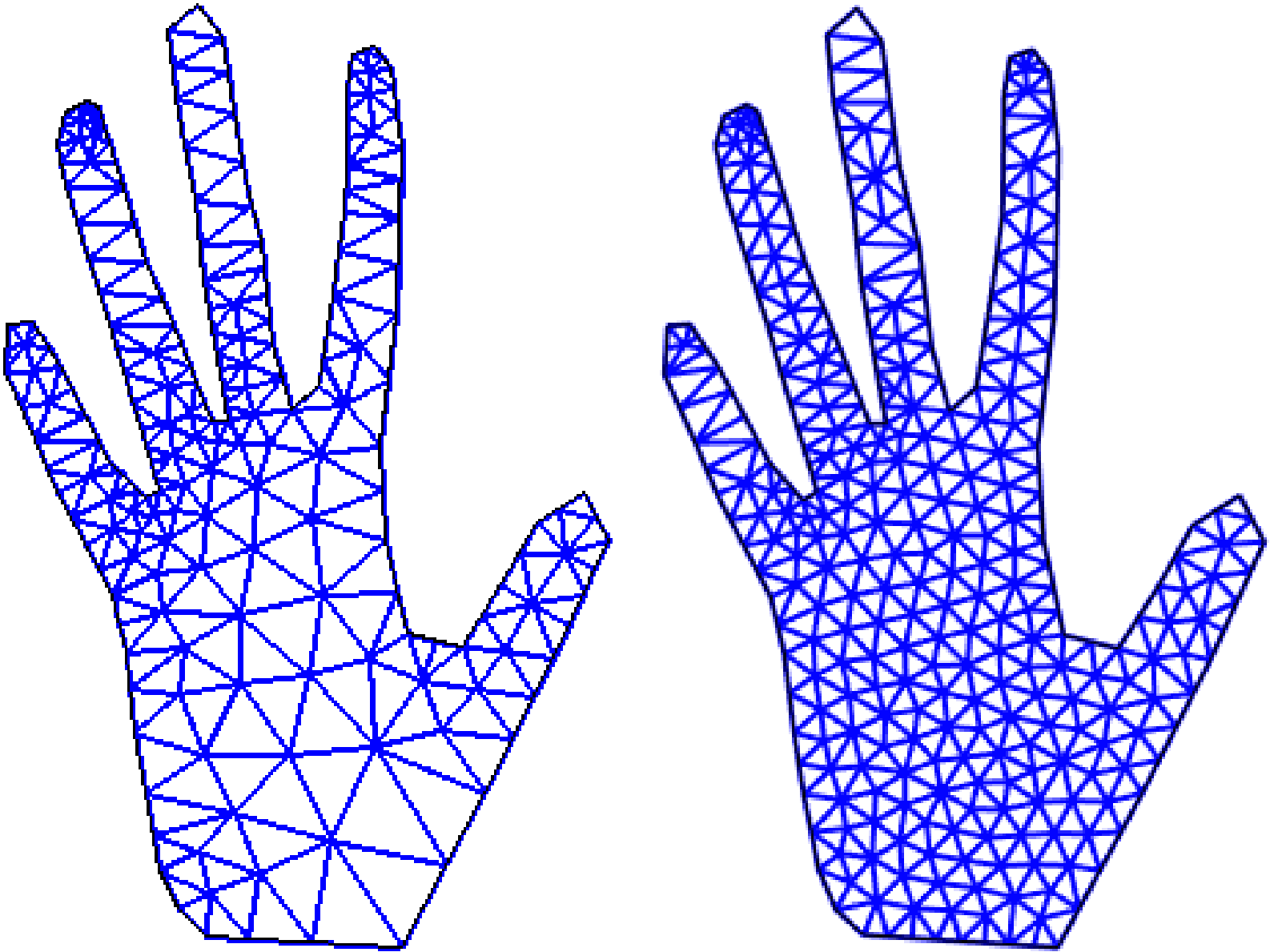
Set Element Size for the Hand

For the hand problem, we had to use many points at curvey places and not many on straight sections. This meant that the size of the triangles varied in an accidental way.

If we specify the triangle size, then we can draw a nice mesh where the size of the triangles does not depend on the accident of how we drew the boundary.

Apply MESH2D to the Hand Data

The mesh on the right uses a maximum triangle size of 0.025:



MESH2D: Use a Density Function

Go back to the original problem, but specify a density function so elements are small near the reentrant corner:

```
v = [ 0.0, 0.0; ...
      2.0, 0.0; ...
      2.0, 1.0; ...
      1.0, 1.0; ...
      1.0, 2.0; ...
      0.0, 2.0 ];

hdata = [];
hdata.fun = @hfun;

[ p, t ] = mesh2d ( v, [], hdata );
```

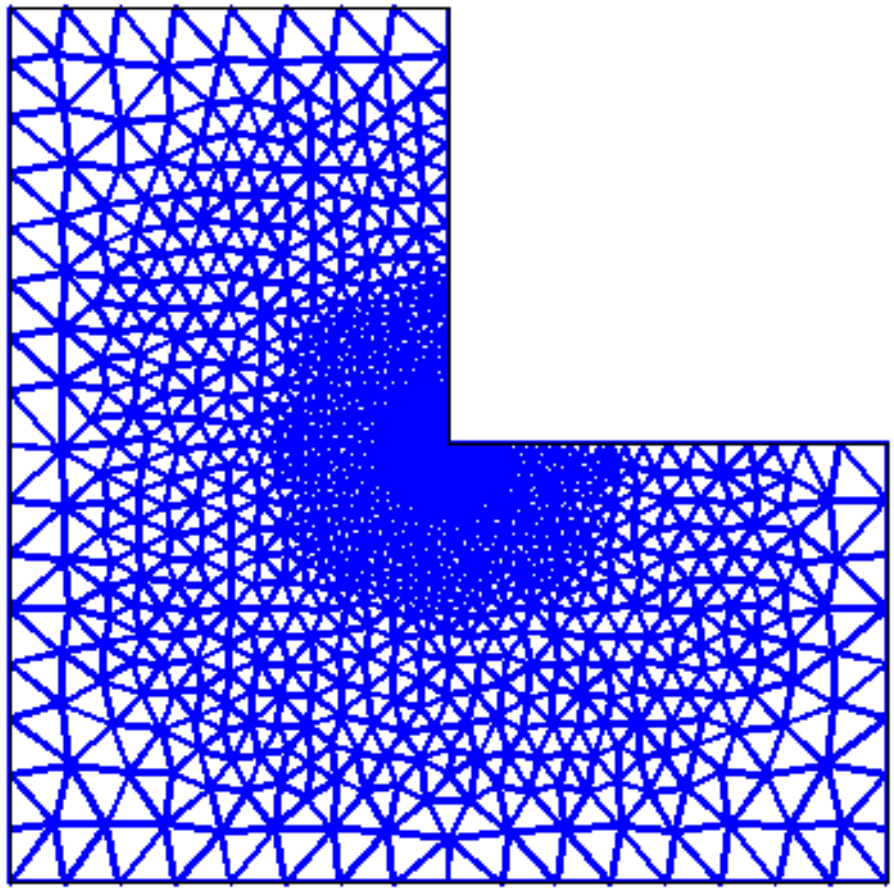
MESH2D: Use a Density Function

```
function h = hfun ( x, y )

%
%  Minimum size is 0.01, increasing as we move away
%  from ( 1.0, 1.0 ).
%
  h = 0.01 + 0.1 * sqrt ( ( x-1.0 ).^2  + ( y-1.0 ).^2 );

  return
end
```

MESH2D: Use a Density Function



A Density Function for the Hand

For the L-shaped region, it's easy to write down a formula that will make h small near (1.0,1.0).

For the hand data, it might also make sense to have a mesh in which the triangles are small near the edges, and large in the palm. But we probably can't imagine how to write a formula for h in this case.

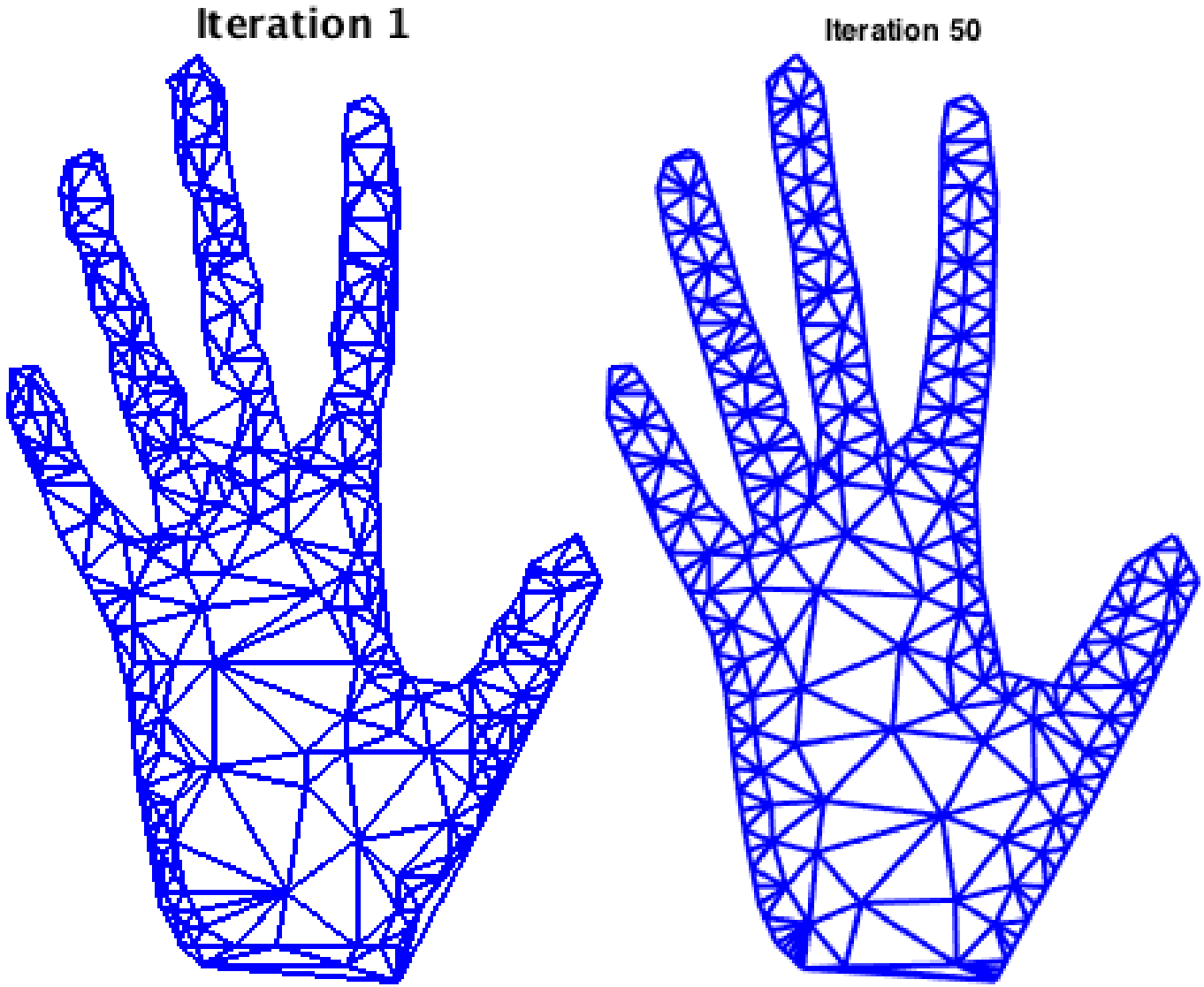
But remember, the boundary is simply a collection of straight line segments. For any point p, and any line segment l that is part of the boundary, we can figure out the distance from p to l. Then the distance from p to the boundary is simply the mininum of all these values.

Even though we can't write a formula in advance, it's possible to compute a formula from the data when it is given.

For convenience, this example will use **distmesh**, a related meshing program.

Using a Density

distmesh starts with a bad mesh, and improves it. Here are estimates 1 and 50:



$2\frac{1}{2}$D:

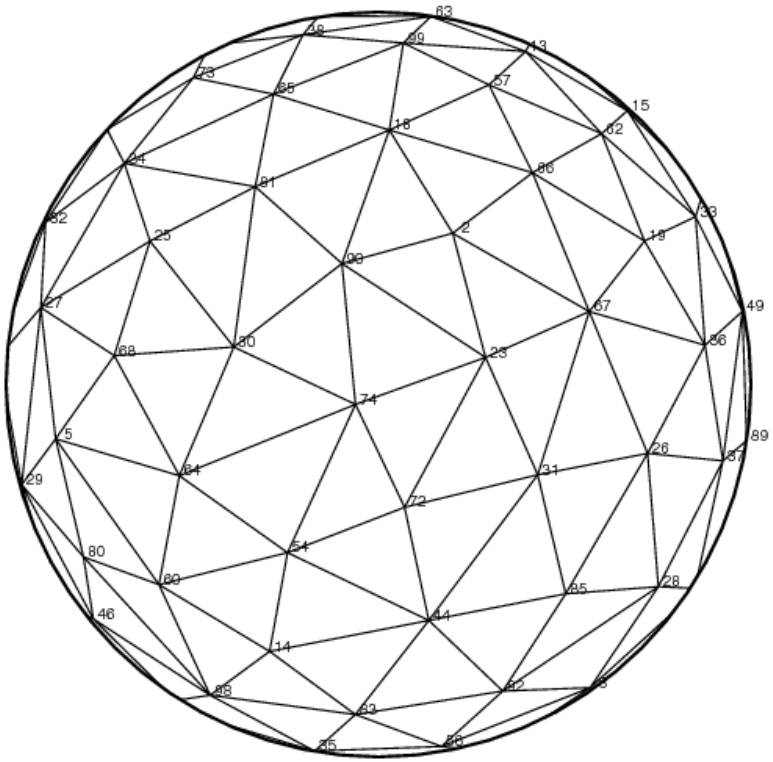
Since we don't have X-ray vision, we can't see inside objects.

Thus, if we are interested in creating a model of a lamp, or a human face, or a whale, we may be satisfied with just the surface, or skin.

Such a model "lives" in a 3D space, but can often be thought of as a 2D surface.

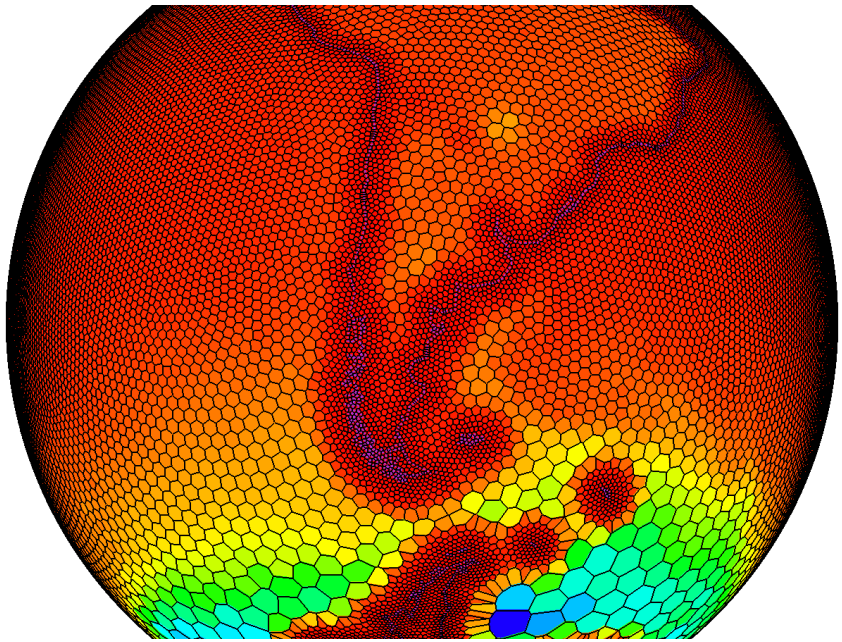
So we may be able to use a mesh of triangles, if we are willing to allow the points to have (x,y,z) coordinates. And we may also have some more complicated connections because a surface can have holes and handles.

$2\frac{1}{2}$D: Grids on a Sphere



If you are generating a model of a surface from tabulated data or formulas, it may be possible to lay out a smoothly varying grid of points. The best way to organize these points into triangles uses the Delaunay method.

$2\frac{1}{2}$D: Grids on a Sphere



Although I have emphasized triangular elements, here is an example of a mesh using hexagons. The meshes of the land and sea used separate tabulated information.

Along the coast, the mesh was made small (using a mesh density) so the land-sea transition is handled smoothly and accurately.

$2\frac{1}{2}$D: Scanner Data

How do you mesh an object that isn't a sphere or the earth?

Simple scanning devices are available which allow you to analyze an object and save the data.

A flatbed scanner registers the height z of the object at a each point (x, y) in a regular grid.

Only the front of the object is measured well; the sides are poorly sampled, and the back not at all.

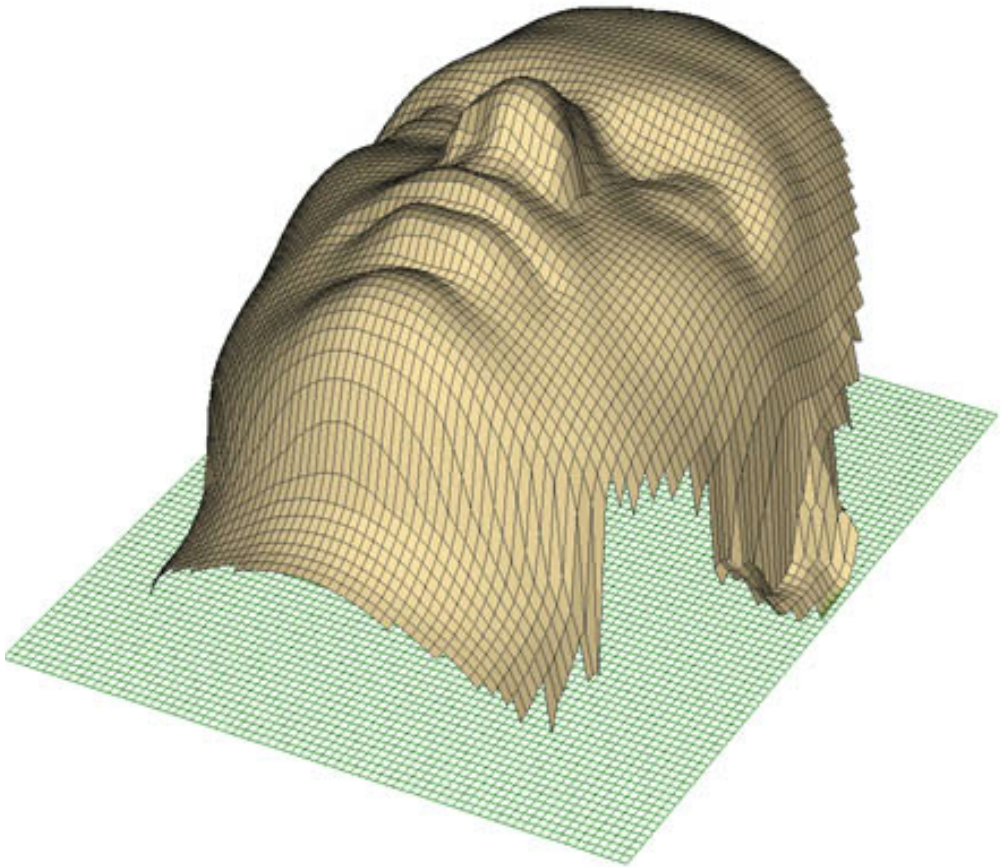
Moreoever, you also get a zero z value for every place not covered by the object.

Enough complaining, let's see what you can get!

$2\frac{1}{2}$D: Scanned Surface

A recent graduate from our department, Detalina Stoyanova, used a flatbed scanner on a collection of bones. Bones are hard to compare, but meshes of bones make a comparison much easier.

$2\frac{1}{2}$D: Scanned Surface



The flatbed scanner image is only very good on the portion of the object directly facing it.

$2\frac{1}{2}$D: Scanner Data

The flatbed scanner models the object as a simple function

$$z = f(x, y)$$

But for many applications, you need to see all sides of the object. A wrap-around scanner registers information at a collection of points (x, y, z), sometimes called a point cloud. It takes some significant work to try to organize these points into a surface, especially if there are holes and handles and sharp edges.

Moreover, the data gathering process can be defective, and there may be places where the scanner did not detect the object properly.

Doing a wrap-around scan

Dennis Slice's group operates a scanner that rotates around the subject. This gives an enormous amount of point data for almost an entire surface.

Doing a wrap-around scan

My hair and glasses gave the scanner real problems. You may not realize that you are looking at a mesh, because it is extremely fine (300,000 points) and we are not expecting the color information.

$2\frac{1}{2}$D: Graphics

Meshlab is an open source, portable, and extensible system for the processing and editing of unstructured triangular meshes in 3D.

Meshlab is aimed to help the processing of the typical not-so-small unstructured models arising in 3D scanning, providing a set of tools for editing, cleaning, healing, inspecting, rendering and converting this kind of mesh.

Meshlab can visualize your mesh, but not your finite element solution. If you want contours of scalars, or vector flow fields, you need to consider working in MATLAB, or try sophisticated graphics package such as ParaView or Visit.

---

http://meshlab.sourceforge.net

True 3D

As long as we are only looking at surfaces, our work is really two dimensional. That's fine if we're doing graphics or studying continents on the surface of the earth.

But sometimes we need to look inside, because we've got data from a medical scanner, or we are studying the heat distribution throughout the interior of a metal object or we want to understand how a solid object can crack and break.

To model true 3D objects, we must replace triangles by tetrahedrons.

True 3D

In 2D, we could create a model if we had an outline (of my hand for instance.)

In 3D, we would normally start with the surface or "skin" of the object, and plan to fill it in.

If we let a program do the filling, it will place sample points inside the surface, and then try to organize them into tetrahedrons (groups of four) that are not too flat or pointy.

As a side effect, we will also get a meshing of the surface, corresponding to tetrahedron faces that happen to touch the outside.

3D: DISTMESH

The **distmesh** program can work on 3D problems almost as easily as in 2D.

It fills in the region with a regular grid of points.

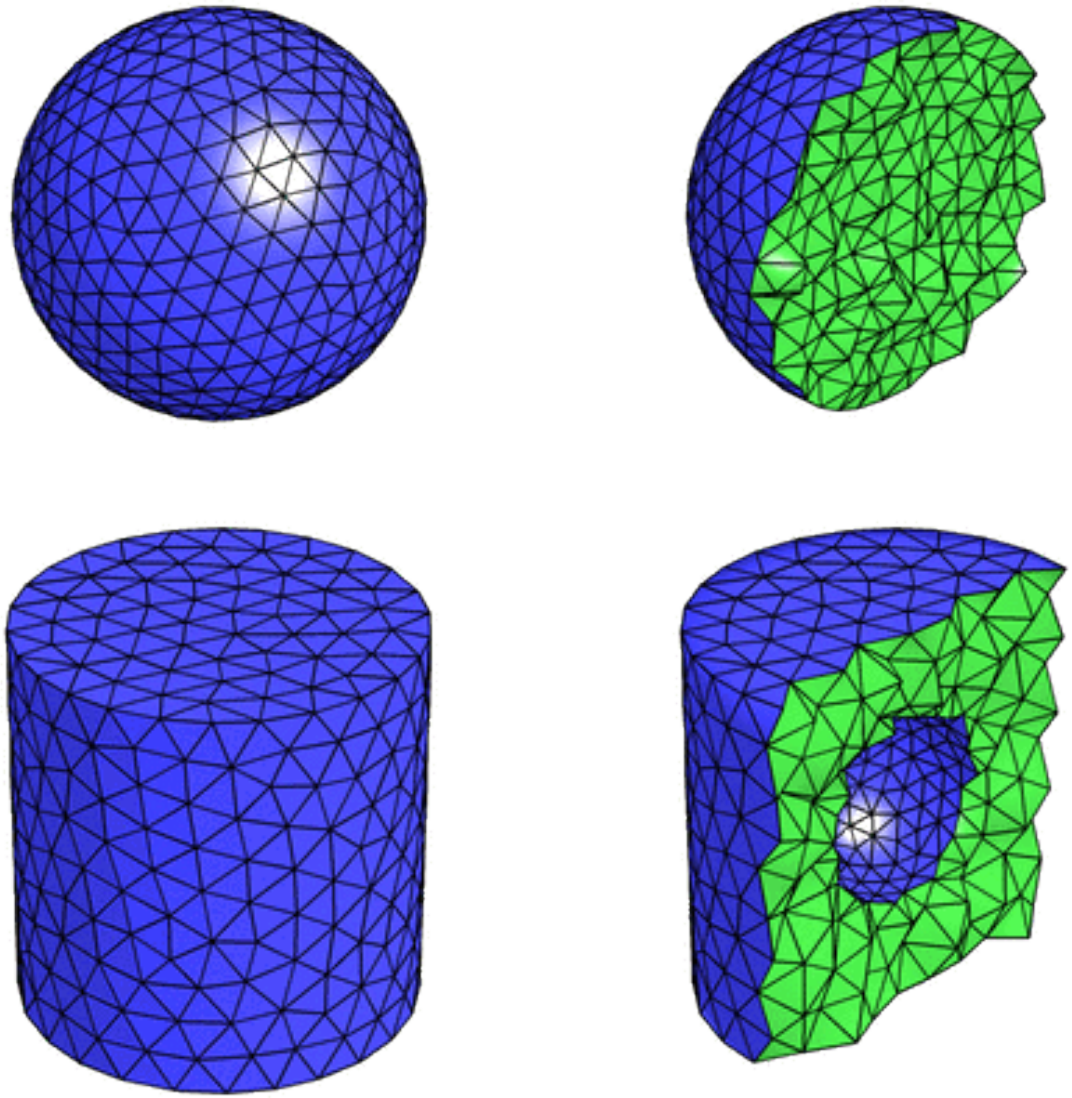
Then it inserts a sort of spring between every pair of points that are neighbors.

If a varying density is desired, then springs of differing stiffness can be used.

Then the program "imagines" that the spring force causes all the points to adjust their positions within the surface.

By taking small steps in this process, the program can find an arrangement of points that is in equilibrium. This arrangement is used to construct the mesh.

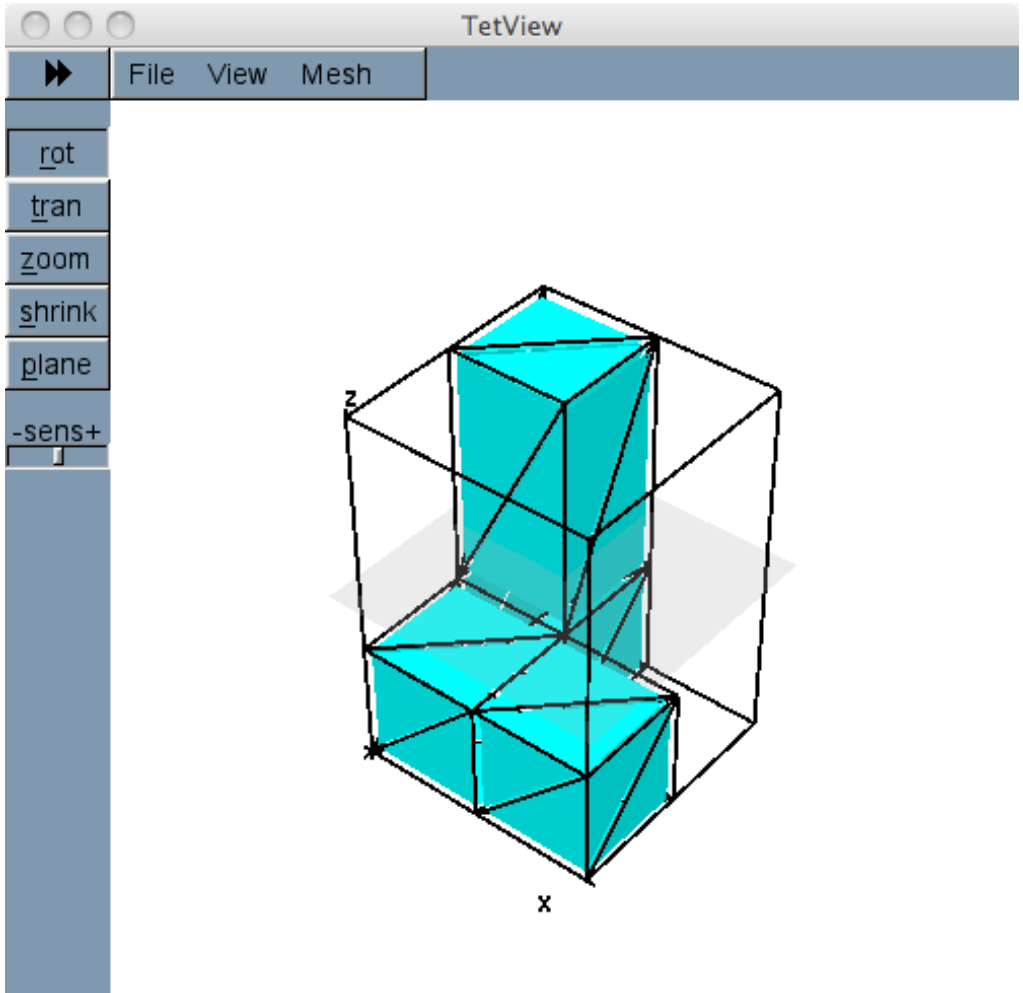
3D: DISTMESH



3D: TETGEN Can Mesh the Data

tetgen is another 3D meshing program. Given the surface below, it creates a mesh of 24 tetrahedrons and 66 triangular faces.

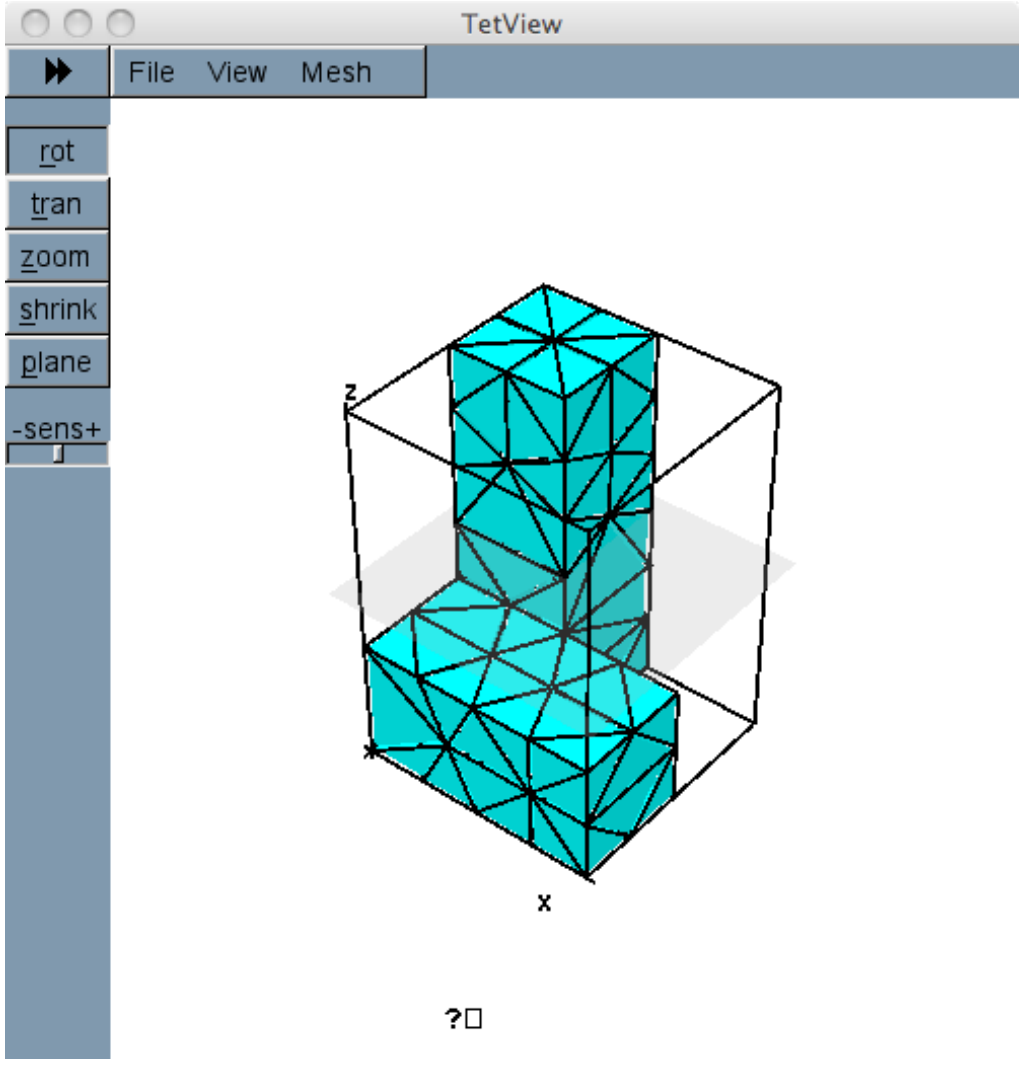
```
tetgen -pq ellell.ply
```



3D: TETGEN Can Refine the Mesh

tetgen can refine the mesh to 132 tetrahedrons:

```
tetgen -ra0.10 ellell.1
```



Mathematics Versus Computing

One of the first times I realized that there was more to math than arithmetic, I read "Every shape, no matter how complicated, can be expressed in terms of equations. Even the profile of your face forms a curve which can be represented as a formula."

Mathematicians love saying things like that and getting that "so there!" look on their face.

One of the virtues of computational science is that it takes mathematical miracles and figures out how to make a practical, if approximate, reality out of them.

Thus geometric meshing finally answered the question I had when I was in high school!