Computational Geometry Lab: BARYCENTRIC COORDINATES IN TETRAHEDRONS

John Burkardt
Information Technology Department
Virginia Tech
http://people.sc.fsu.edu/∼jburkardt/presentations/cg_lab_barycentric_tetrahedrons.pdf

December 23, 2010

1 Introduction to this Lab

In this lab we consider the *barycentric coordinate system* for a tetrahedron. This coordinate system will give us a common map that can be used to relate all tetrahedrons. This will give us a unified way of dealing with the computational tasks that must take into account the varying geometry of tetrahedrons.

We start our exploration by asking whether we can come up with an arithmetic formula that measures the position of a point relative to a plane. The formula we arrive at will actually return a *signed distance* which is positive for points on one side of the plane, and negative on the other side. The extra information that the sign of the distance gives us is enough to help us answer the next question, which is whether we can determine if a point is inside a tetrahedron. This turns out to be simple if we apply the signed distance formula in the right way.

The distance function turns out to have another crucial use. Given any face of the tetrahedron, we know that the tetrahedral point that is furthest from that face is the vertex opposite to the face. Therefore, any other point in the tetrahedron has a relative distance to the face that is strictly less than 1. For a given point, if we compute all four possible relative distances, we come up with our crucial barycentric coordinate system.

The barycentric coordinate system has a particularly simple interpretation when we consider the *reference tetrahedron*. If we select a set of points in the reference tetrahedron, it is easy to determine the corresponding points in any other tetrahedron. Thus, the reference tetrahedron is generally used to define things such as quadrature rules, whose values can then easily be adapted to the particular tetrahedron of interest.

The barycentric coordinate system will come up again in other labs, where we will have a chance to show some of its many special properties and uses.

2 Signed Distance From a Plane

In 3 dimensional space, a **plane** is usually thought of as "defined" by three points. We understand that a plane is the "flat sheet" that passes through those three points, and a little thought convinces us that we do have to be careful, when picking those three points, that they not lie on a single common line (or that any two of the points are the same!).

This geometric intuition is helpful in thinking about planes, but we prefer to be able to write some kind of algebraic formula so that the answer to whether a point belongs to a plane comes down to whether the formula has the right value at that point.

The algebraic definition of a plane goes as follows: a plane **PL** in 3 dimensional space may be defined by a point **Q** and a vector $\vec{n}$ having unit length. The plane **PL** = {Q,$\vec{n}$} then consists of all points **P** such that

$$(P - Q) \cdot \vec{n} = 0$$

that is, the difference vector **P-Q** is orthogonal to $\vec{n}$.

If necessary, it is possible to convert between the geometric definition of a plane in terms of three points, and this algebraic definition in terms of a point and a vector. For our purposes, however, the algebraic definition is just what we want.

Now that we have a formula for a plane, we may want to know whether we can measure distances between an arbitrary point and a given plane. The answer turns out to be ready to drop into our laps.

Given a plane **PL** = {Q,$\vec{n}$} and a point **P**, the **signed distance** from **P** to **PL** is given by

$$\text{SignedDistance}(P, PL) = (P - Q) \cdot \vec{n}.$$

Notice that this implies that we can define a plane as the set of points for which the signed distance function is 0! Moreover, the signed distance function essentially parameterizes the entire space as a family of planes, indexed by their signed distance from the fundamental plane **PL**.

Now the signed distance function is *almost* a distance function; as its name implies, it can return a negative value for distance, however. Points on one side of the plane will have a positive distance, and points on the other will have a negative distance. So if we want a distance in the ordinary sense, we simply take the absolute value. It turns out, however, that the information contained in the sign of the signed distance function will turn out to be useful as well.

3 Program #1: Point/Plane Orientation

Write a program which

- reads the definition of a plane **PL**;
- computes the unit normal vector to the plane, $\vec{n}$;
- reads the definition of a point **P**;
- computes $D = \text{SignedDistance}(P, PL)$;
- prints **PL**,$\vec{n}$, **P** and **D**.

To test your program, use the plane **PL**={ {1,2,3}, {4,1,2},{2,4,4}}.

As test points **P**, use {3,2,5}, {$\frac{7}{3}$,$\frac{7}{3}$,3}, and {1,3,-5}.

4 Does a Point Lie Inside a Tetrahedron?

An important task in computational geometry is the determination of whether a point **P** is contained inside some geometric shape. The eye can answer such a question easily, but we must come up with arithmetic formulas instead. So let us consider whether we can tell that a point is inside a tetrahedron or not.

Now the tetrahedron is bounded by four planes, and the points of the tetrahedron are exactly all those points which are on the "correct" side of all four of those planes. Thus, it seems like we might be able to use our signed distance function to help us here.

Unfortunately, we don't know, for any of these planes, which is the side that will give us positive distances, and which side gives negative. Let's assume we're very lucky, and that, by chance, the signed distance function will always return a *positive* distance between a point in the tetrahedron and any of the four planes.

In that case, to determine whether a point is inside the tetrahedron, we simply compute its signed distance from each of the four faces (that is, the planes that include the faces). If all four distances are positive (or at least nonnegative), then the point lies on the correct side of each plane, and hence is in the tetrahedron. If any distance is strictly negative, the point is not in the tetrahedron.

Of course, we can't rely on being lucky enough that the positive region of the signed distance function will always happen to lie on the side of the plane we are interested in. But there's an easy way to fix this. Instead of demanding that the signed distance be positive, we can simply require that it have the same sign as another point that we are sure is in the tetrahedron. Since the plane we are checking involves three vertices, the fourth vertex can be used as this reference point.

Say we're checking on the point **P** in tetrahedron **T={a,b,c,d}**. We begin by computing the signed distance from **P** to the face **{b,c,d}**. Instead of demanding that this value be positive, we simply compare it with the signed distance of the vertex **a** to the same face. We don't need to worry about the sign of the distance, just that it's the same for both points.

In a similar way, we compare the signed distances of **P** and vertex **b** to the face **{a,c,d}**, and so on. If in every case, the sign of the signed distance of **P** and the vertex match, then **P** is inside the tetrahedron.

Thus, we have an algebraic formula that tells us whether a point is inside a tetrahedron.

5 Program #2: Does a Point Lie Inside a Tetrahedron?

Write a program which

- reads the definition of a tetrahedron**T={a,b,c,d}**;
- reads a point **P**;
- computes and prints the signed distances from **P** to each of the four faces **{b,c,d}**, **{a,c,d}**, **{a,b,d}** and **{a,b,c}**;
- computes and prints the signed distances from **a** to **{b,c,d}**, **b** to **{a,c,d}**, **c** to **{a,b,d}** and **d** to **{a,b,c}**;
- prints a message that **P** is inside or outside the tetrahedron.

Test your program on tetrahedron Tet3 defined by:

```
{ { 1.0,  2.0,  3.0 },
  { 2.0,  2.0,  3.0 },
  { 1.0,  3.0,  3.0 },
  { 1.0,  2.0,  9.0 } }
```

For your points **P** try each of these:

```
{ 1.25, 2.25, 4.50 }
{ 1.00, 2.50, 4.00 }
{ 1.00, 3.00, 3.00 }
{ 1.00, 2.00, 7.80 }
{ 0.80, 2.40, 6.60 }
```

6 Barycentric Coordinates

Instead of comparing the signed distance of a point **P** and a vertex to the opposite face, suppose we take the *ratio* of these two signed distances. The ratio will be positive exactly when the two signed distances agree. Moreover, since the vertex is the point in the tetrahedron that is as far as possible from the opposite face, the ratio will be 0 for points on the face, 1 for the vertex, and strictly between 0 and 1 for points that are strictly inside the tetrahedron.

This revision of our measurement system has a lot to recommend it. In fact, we will use it enough that we want to come up with a system for labeling all four possible measurements. We will use the symbol ξ to represent any of these ratios of signed distances, and use a subscript to indicate the vertex used for comparison. Our set of ratios is:

$$\begin{aligned}\xi_a =& \frac{\text{SignedDistance}(P, \{b, c, d\})}{\text{SignedDistance}(a, \{b, c, d\})}\\ \xi_b =& \frac{\text{SignedDistance}(P, \{a, c, d\})}{\text{SignedDistance}(b, \{a, c, d\})}\\ \xi_c =& \frac{\text{SignedDistance}(P, \{a, b, d\})}{\text{SignedDistance}(c, \{a, b, d\})}\\ \xi_d =& \frac{\text{SignedDistance}(P, \{a, b, c\})}{\text{SignedDistance}(d, \{a, b, c\})}\end{aligned}$$

These ratios are known as the *barycentric coordinates* of a point **P** with respect to a tetrahedron **T**. The coordinates can be computed for any point, whether it is inside or outside the tetrahedron. We know that all four of these values must be nonnegative if the point is to be inside the tetrahedron. However, another surprising fact is that the barycentric coordinates satisfy the following condition:

$$\xi_a + \xi_b + \xi_c + \xi_d = 1$$

7 Program #3: Barycentric Coordinates

Write a program which:

- reads the definition of a tetrahedron **T={a,b,c,d}**;
- reads a point **P**;
- computes and prints $(\xi_a, \xi_b, \xi_c, \xi_d)$.

Test your program on tetrahedron Tet3 defined by:

```
{ { 1.0,  2.0,  3.0 },
  { 2.0,  2.0,  3.0 },
  { 1.0,  3.0,  3.0 },
  { 1.0,  2.0,  9.0 } }
```

For your points **P** try each of these:

```
{ 1.25, 2.25, 4.50 }
{ 1.00, 2.50, 4.00 }
{ 1.00, 3.00, 3.00 }
{ 1.00, 2.00, 7.80 }
{ 0.80, 2.40, 6.60 }
```

Compare these results with your “inside/outside” program results.