Computational Geometry Lab: MAPPING TRIANGLES

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

August 28, 2018

1 Introduction to this Lab

In this lab we seek a method of *mapping* one triangle to another, that is, establishing a correspondence between their points. Our path to this mapping will begin by choosing a special **reference triangle** of a standard shape and position. We will then consider the specialized problem of computing a mapping from this reference triangle to some other triangle.

We will identify this mapping as an **affine function**, which is a simple modification of a linear function. When we write the affine function in matrix form, it will become obvious how to compute the inverse map, from the other triangle back to the reference triangle.

But now this implies that we can create a similar map between **any** two triangles, as long as we are willing to use the reference triangle as an intermediary point, using the composition of maps.

These mappings are useful tools. As an example, we show how a quadrature rule can be defined just once, on a reference triangle, but then used on any triangle by an appropriate use of the mapping function.

The simple affine map discussed here is very useful in the finite element method, for constructing a mesh that triangulates a region, for applying quadrature rules on each triangle in the mesh, and even for using the inverse mapping so that an integral over a triangle can be computed over the reference triangle instead.

In finite elements, it is possible to design more sophisticated mappings of the reference triangle so that the image of the triangle actually has curved sides which can better match the shape of a complicated region. Such mappings are nonlinear, and beyond the scope of this lab.

2 The Reference Triangle

We will find it convenient to agree on a single standard triangle called the *reference triangle*. The actual definition of this triangle is simply **Tref**={ {1,0}, {0,1}, {0,0} }.

For technical reasons, we will find it convenient in this lab to list the vertices in exactly this order. We would normally use x and y to denote coordinate directions. However, we are going to be considering maps between the reference triangle and other triangles. Let's allow the other triangles to use the (x, y) coordinate system, while for the reference triangle we will denote the coordinate directions as r and s.

Consider figure 1, in which we have displayed an image of the reference triangle. This triangle can be defined as the intersection of three half planes:

$$\begin{aligned} 0 \leq & r \\ 0 \leq & s \\ r + s \leq & 1 \end{aligned}$$

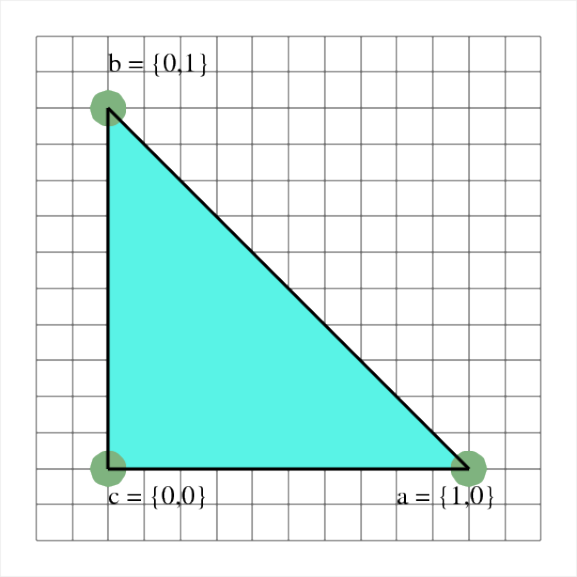

Figure 1: The Reference Triangle

Suppose that we define a new variable $t = 1 - r - s$. Then our reference triangle definition has the nicely symmetric form:

$$\begin{aligned} 0 \leq & r \\ 0 \leq & s \\ 0 \leq & t \end{aligned}$$

Our quantity t is not really an independent coordinate, but it does allow us to simplify some of our discussions. It is also related to the barycentric coordinate system which is discussed in another lab.

3 A Map From the Reference Triangle to Another Triangle

Now let's consider the problem of coming up with a map $\phi_{Tref,T}(r,s)$ from the reference triangle to a triangle **T**. The map takes a typical point (r,s) of the reference triangle and maps it to the point $\phi_{Tref,T}(r,s) = (x,y)$

The simplest map that comes to mind would be some kind of linear map. This can't be quite right, though, since every linear map would send the vertex {0,0} of the reference triangle to {0,0}, which is probably not even a point in the triangle **T**, let alone a vertex of it.

But this objection suggests a partial solution. If we want {0,0}, the third vertex of **Tref** to map to vertex c of **T**, we can take any linear mapping, and then simply add c to the result. Writing $\vec{r}$ for (r,s) and $\vec{x}$ for (x,y), we can think of our desired map as having the form:

$$\phi_{Tref,T}(\vec{r}) = A \cdot \vec{r} + \vec{c} = \vec{x}$$

where we don't know A yet, but $\vec{c}$ is definitely the third vertex of **T**!

To see what's going on with A, let's see what happens when $\vec{r} = (1,0)$, which we want to map to vertex a of **T**:

$$\phi_{Tref,T}(1,0) = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix} = \begin{pmatrix} a_x \\ a_y \end{pmatrix}$$

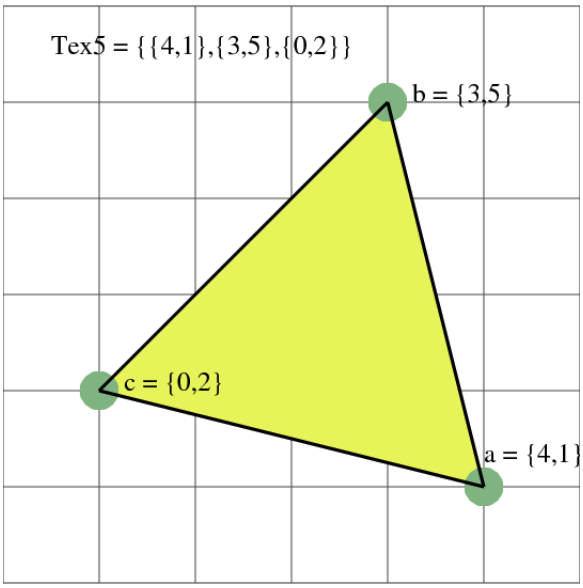

Figure 2: Example Triangle Tex5

This immediately yields the results that

$$\begin{aligned} A_{11} =& a_x - c_x \\ A_{21} =& a_y - c_y. \end{aligned}$$

The remaining two entries of A can be determined by working out the desired result that $\vec{r} = (0, 1)$ maps to vertex b of **T**. We can now write out our mapping in full:

$$\phi_{Tref,T}(\vec{r}) = \begin{pmatrix} a_x - c_x & b_x - c_x \\ a_y - c_y & b_y - c_y \end{pmatrix} \cdot \begin{pmatrix} r \\ s \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix} = \vec{x}$$

Now it is common to think of this as a linear map, but as mentioned earlier, a linear map sends $\vec{0}$ to $\vec{0}$ and that does not happen with this map. We also think of the function $y = a * x + b$ as being a linear function, but that's also technically wrong. In both cases, we are dealing with what is called an **affine map**, which can be regarded as the sum of a linear function and a constant function.

4 Program #1: Mapping from the Reference Triangle

Write a program which

- reads the definition of a triangle **T={a,b,c}**;
- computes the 21 points in **Tref** of the form $(r, s) = (\frac{i}{5}, \frac{j}{5})$;
- computes the images (x, y) of these points in **T** under the map $\phi_{Tref,T}$;
- plots the image points (x, y).

For **T**, use the triangle **Tex5** defined by {{4,1},{3,5},{0,2}}.

5 The Inverse Map

One way you will realize that the function $y = a * x + b$ is affine, and not linear, is when you try to invert the map. The first step is to subtract the constant function; the remaining linear function is then easy to invert: $x = \frac{y-b}{a}$. Similarly, when we are dealing with an affine map in multiple dimensions, we can use the matrix notation:

$$\phi_{Tref,T}(\vec{r}) = A \cdot \vec{r} + \vec{c} = \vec{x}$$

and so the inverse map is easily computed, once we subtract the constant term.

$$\phi^{-1}_{Tref,T}(\vec{x}) = A^{-1} \cdot (\vec{x} - \vec{c}) = \vec{r}$$

A linear function in a space of N dimensions is completely defined by its action on N distinct points, whereas an affine function requires $N + 1$ points. In both cases, we also must assume that the points are in "general position".

For our affine maps in 2D, this simply means that the three vertices of the triangle can't lie on a common line; in other words, the triangle must have nonnegative area. And so we have the result that in the plane, any affine map can be completely characterized by its values at the vertices of a (nondegenerate) triangle.

In order to compute the inverse mapping, we first compute the determinant of A:

$$\det A = (a_x - c_x)(b_y - c_y) - (a_y - c_y)(b_x - c_x)$$

This quantity is nonzero exactly when the area of **T** is nonzero. *(Why?)* The inverse of A can then be written as

$$A^{-1} = \frac{1}{\det A} \begin{pmatrix} (b_y - c_y) & -(b_x - c_x) \\ -(a_y - c_y) & (a_x - c_x) \end{pmatrix}$$

Thus, we have all the details necessary to set up the inverse function that takes any point $\vec{x}$ in **T** and produces the corresponding point $\vec{r}$ in the reference triangle.

We now have the formula for an affine map from **Tref** to an arbitrary triangle **T**, and, assuming **T** is nondegenerate, the formula for the inverse map as well.

6 Program #2: The Inverse Map

Write a program which

- reads the definition of a triangle **T={a,b,c}**;
- computes the 21 points in **Tref** of the form $(r, s) = (\frac{i}{5}, \frac{j}{5})$;
- computes the images (x, y) of these points in **T** under the map $\phi_{Tref,T}$;
- computes the inverse images (r_2, s_2) of these points in **Tref** under the map $\phi^{-1}_{Tref,T}$;
- prints (r, s), (x, y) and (r_2, s_2)

For **T**, use the triangle **Tex5** defined by {{4,1},{3,5},{0,2}}.

7 Mapping Between General Triangles

Our original goal was to establish maps between any pair of nondegenerate triangles **Ta** and **Tb**. What we have done so far assumes that one of the triangles is the reference triangle **Tref**. But this really gets us what we need. To define a mapping $\phi_{Ta,Tb} : Ta \to Tb$, we will simply compose the mappings from **Ta** to **Tref**

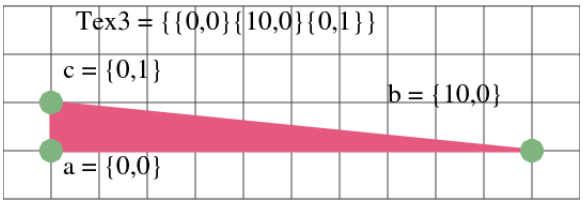

Figure 3: Example Triangle Tex3

and from **Tref** to **Tb** (read the composition formula carefully, because the order may not be quite what you expect):

$$\phi_{Ta,Tb} = \phi_{Tref,Tb} \circ \phi_{Ta,Tref}$$

The form of the inverse map should be obvious.

Suppose we actually want to work out the matrices involved in this mapping. In particular, let's suppose that our individual mappings have the forms:

$$\begin{aligned}\phi_{Tref,Ta}(\vec{r}) =& A_a \cdot \vec{r} + \vec{c}_a = \vec{x}_a \\ \phi_{Tref,Tb}(\vec{r}) =& A_b \cdot \vec{r} + \vec{c}_b = \vec{x}_b\end{aligned}$$

Then the matrix form of the map $\phi_{Ta,Tb}$ which takes a point $\vec{x}_a$ in **Ta** to the point $\vec{x}_b$ in **Tb** is

$$\phi_{Ta,Tb}(\vec{x}_a) = A_b \cdot \left(A_a^{-1}(\vec{x}_a - \vec{c}_a)\right) + \vec{c}_b = \vec{x}_b$$

Since we've already worked out the formulas for the transformation matrix and its inverse when the reference triangle is involved, we simply need to put the pieces together in the right way to construct the mapping between **Ta** and **Tb**. So we have now accomplished our goal of establishing a mapping between arbitrary triangles.

Note that this somewhat complicated formula really represents another affine map, and could be rewritten in the form

$$\phi_{Ta,Tb}(\vec{x}_a) = B \cdot \vec{x}_a + \vec{d} = \vec{x}_b$$

for some matrix B and vector $\vec{d}$. However, this information would only be worth computing in this form if we were going to do many transformations between **Ta** and **Tb**.

8 Program #3: Mapping Between General Triangles

Write a program which

- reads the definition of triangles **Ta** and **Tb**;
- computes 10 random points $\vec{x}_a$ in **Ta**;
- computes the images $\vec{x}_b$ of these points in **Tb**;
- computes the inverse images $\vec{x*}_a$ of these points in **Ta**;
- prints $\vec{x}_a$, $\vec{x}_b$ and $\vec{x*}_a$

For **Ta**, use the triangle **Tex5** defined by {{4,1},{3,5},{0,2}} and for **Tb**, use the triangle **Tex3** defined by {{0,0},{10,0},{0,1}}.

9 Mapping Quadrature Rules

A quadrature rule for a region $\mathcal{R}$ is a set of n weights w_i and abscissas $\vec{x}_i$ designed to approximate integrals of functions over the region. That is, the quadrature rule approximates integrals by a weighted sum of function values:

$$\int_{\mathcal{R}} f(\vec{x})\, d\vec{x} \approx \sum_{i=1}^{n} w_i f(\vec{x}_i)$$

Obviously, a given quadrature rule can only work correctly for a particular triangle. If we have hundreds of triangles in a mesh, we don't want to have to work hard figuring out a new quadrature rule for each one. There must be some simple way to adjust a particular rule to the peculiar geometry of each triangle in our mesh.

Luckily, the approximation properties of a quadrature rule are preserved under affine maps. Our entire triangulation can be regarded as a series of affine mappings of the reference triangle. That means that we can determine a quadrature rule that accurately approximates integrals on the reference triangle, then map the abscissas from the reference triangle to the triangle we're interested in, appropriately modify the weights, and thereby construct an accurate rule for the triangle that actually interests us.

(We could also imagine an inverse process, in which we map the integration problem over the given triangle *back* to the reference triangle. In that case, we can use the quadrature rule exactly as stated, though on a transformed integrand. This may be more than you want to think about, but it is a technique that is often used in finite element computations.)

The mapping we have constructed tells us how to transform the abscissas, of course, but what is the issue with the weights? It's actually very simple. Suppose we were to integrate the function $f(\vec{x}) = 1$ over $\mathcal{R}$. Then the exact integral is equal to area($\mathcal{R}$). If the quadrature rule is to have any accuracy at all, then it follows that the weights of any quadrature rule for $\mathcal{R}$ must always sum to area($\mathcal{R}$).

This means that the weights for a quadrature rule over the reference triangle must sum to $\frac{1}{2}$. Now, when we transform the quadrature rule to the new triangle $\mathbf{T}$, the weights must sum to area(T). Because we are using an affine map, all the weights change in exactly the same proportion; in other words, each reference weight is multiplied by $2 * \text{area}(T)$ to become the weight for the quadrature rule over $\mathbf{T}$.

(We mentioned in the introduction that there are higher order mappings of the triangle. In that case, the nonlinearity of the mapping means that the weights of a transformed quadrature rule will be transformed by differing amounts, depending on the Jacobian of the transformation at the corresponding abscissa. So be sure to appreciate the simplicity of the affine mapping case!)

10 Program #4: Mapping a Quadrature Rule

The table gives examples of quadrature rules for the unit triangle, with the order N, precision P, weights W, and abscissas (X, Y).

Write a program which:

- reads a triangle $\mathbf{T}$;
- reads the weights and abscissas of a quadrature rule for **Tref**;
- transforms the quadrature rule to the triangle $\mathbf{T}$;
- uses the quadrature rule to estimate the integral of some function $f(\vec{x})$ over $\mathbf{T}$.

For your triangle, use the triangle **Tex5**. For your quadrature rule, try the four point rule, which is precise for polynomials up to total degree 3. For your function $f(\vec{x})$ try $f(x, y) = 1$, $f(x, y) = xy^2$ and $f(x, y) = e^{x+y}$.

Table 1: Some Quadrature Rules for the Reference Triangle.

N	P	W	X	Y
1	1	0.500000	0.333333	0.333333
3	2	0.166666	0.500000	0.000000
		0.166666	0.500000	0.500000
		0.166666	0.000000	0.500000
4	3	-0.281250	0.333333	0.333333
		0.260416	0.600000	0.200000
		0.260416	0.200000	0.600000
		0.260416	0.200000	0.200000
6	4	0.054975	0.816847	0.091576
		0.054975	0.091576	0.816847
		0.054975	0.091576	0.091576
		0.111690	0.108103	0.445948
		0.111690	0.445948	0.108103
		0.111690	0.445948	0.445948
7	5	0.112500	0.333333	0.333333
		0.062969	0.797427	0.101287
		0.062969	0.101287	0.797427
		0.062969	0.101287	0.101287
		0.066197	0.059716	0.470142
		0.066197	0.470142	0.059716
		0.066197	0.470142	0.470142